VALID GUIDE TO OPERATIONS

Manual Number: MN210 Rev A

10 March 1986

Valid Logic Systems, Incorporated
2820 Orchard Parkway
San Jose, California 95134
(408) 945-9400 Telex 371 9004

TABLE OF CONTENTS

SECTION 1
OVERVIEW

This manual is the operations guide for the Logic Designer AT. The manual contains procedures for the following groups of tasks:

- Day-to-Day Operations

  System startup and shutdown
  Moving between DOS and UNIX
  Operating in super-user mode
  Creating user accounts
  Defining passwords
  Setting system time/date
  File transfers between UNIX and DOS

- System Maintenance

  Backup procedures for UNIX and DOS
  Restoring files
  System configuration and device names
  Adding a printer or plotter
  Moving procedures

- Network Maintenance

  Ethernet application utilities

- Troubleshooting

  Troubleshooting/diagnostics guide

- Software Installation Instructions

  Installing DOS resident software
  Configuring UNIX
  Installing UNIX and GED
  Installing other SCALD software

Installing ethernet software
Preparing the system for use

The manual also contains the following appendices:

- Appendix A: System Messages
- Appendix B: Devices
- Appendix C: Ethernet Administration

These manuals are referenced by the Operations Guide:

- *UNIX System V: Volume I (User Guide with tutorials)*
- *UNIX System V: Volume II (User Reference Manual)*
- *UNIX System V: Volume III (Administrator Reference Manual)*
- *UNIX System V: Volume IV (Programmer Reference Manual)*
- *UNIX System V: Volume V (Administration Guide)*
- *UNIX System V: Volume VI (Support Tools Guide)*
- *Guide to Operations, IBM Personal Computer AT*
- IBM *Disk Operating System*
- *Hardware Installation Guide for the Logic Designer AT*

In this manual, UNIX commands appear in **bold** type and are usually followed by a number and/or letter in parentheses. The number indicates the "man page" group containing a detailed description of the command. Man pages are contained in *UNIX System V, Volume II (User Reference Manual).*

This manual assumes a basic working knowledge of UNIX. If you are not familiar with UNIX, use the tutorials in the manual *UNIX System V: Volume I.*

If your machine has not been installed for you, refer to the *Hardware Installation Guide for the Logic Designer AT.*

For a step-by-step introduction to Valid's design tools, see the following tutorials:

- *Tutorial I, Introduction to the Logic Designer AT*
- *Tutorial II, Advanced Topics*

Tutorial I introduces the basic operations of the Graphics Editor, the Compiler, and the Packager, as well as basic information on libraries. Tutorial II introduces the ValidSIM and ValidTIME analysis tools and hierarchical design methodology.

SECTION 2
DAY-TO-DAY OPERATIONS

This section covers routine system administration tasks.

2.1 SYSTEM STARTUP AND SHUTDOWN

Once UNIX System V has been installed, the day-to-day start-up of UNIX is simple.

1. Power up your system and start up DOS as usual.

2. At the DOS prompt, type:

   unix

   The following messages appear (the list of devices varies depending on the individual system):

   ```
   Level 1 Test: PASS
   Level 2 Test: PASS
   Configuration: 32.16 2MB dos console
                  dsk/0 flpa flpb lp lp1
   ```

   A file system consistency check (**fsck**(1M)) is now done on the root file system (/dev/dsk/0s0). This is a sample display; the exact numbers for yours will be different.

   ```
   $$ :fsck /dev/dsk/0s0

   /dev/dsk/0s0
   File System: Volume:

   ** Phase 1 - Check Blocks and Sizes
   ** Phase 2 - Check Pathnames
   ** Phase 3 - Check Connectivity
   ** Phase 4 - Check Reference Counts
   ```

```
** Phase 5 - Check Free List
1051 files 13430 blocks 2966 free
```

This display illustrates a file system check with no problems. See the *UNIX System V Release 2.0 Administrator Guide* for more information on the file system check utility, **fsck**.

3. UNIX is now booted from DOS. The system displays the amount of real and available memory in your system. This differs for different systems.

```
$$ /unix -Q -u 1000000

System date is: Wed Dec 18  11:24:33  PST 1985

cron started
errdemon started

Console Login:
```

You can now log on to the system. Note that the system date displayed can be changed if it is incorrect. See Section 2.6 for more information.

To make UNIX come up in single user mode by default, modify the file /etc/inittab, changing the first line from "is:2:initdefault" to "is:s:initdefault". However, there should be no reason to operate in single user mode. See the document "Single User and Multiuser" in the *UNIX System V Release 2.0 Operator Guide*, **init**(1M), and **inittab**(4) for further information on the **init** states.

Section 2.4 tells how to create user accounts. If security is important for your site, root should have a password, and all users should have passwords too. See Section 2.5 for more information on creating passwords.

SHUTDOWN PROCEDURES

Follow these procedures whenever you switch off the system:

1. Log on as root.
2. Enter the UNIX command "shutdown". Answer questions from the shutdown procedure.
3. The message "INIT: Single user mode" appears. Type "dos" to enter DOS.
4. Switch off the IBM PC/AT.
5. Switch off the monitor. (If you have a GX monitor, switch off the monitor control unit.)

2.2 MOVING BETWEEN DOS AND UNIX

If you are in DOS and want to enter UNIX, enter the command "unix". Note that if UNIX has already been booted during the current terminal session, the system will not reboot. You will simply return to your UNIX session right where you left off.

If you are in UNIX and want to switch to DOS, type "dos".

2.3 OPERATING IN SUPER-USER MODE

Installations that run UNIX operating systems usually distinguish between regular users and one or more "super-users". Super-users have special access privileges; for example, some files can be edited and some programs can be run only by a super-user. Super-users generally have responsibility for system administration and maintenance.

Logging in successfully with the "root" login name and password confers super-user privileges, so the super-user is often referred to simply as "root". You can also log in as

super-user after logging in as a regular user by entering the command **su** and then the super-user password when prompted. When you exit from super-user in this case, you return to your session as a regular user. In single-user mode you are root automatically.

2.4 CREATING USER ACCOUNTS

Use the **mkusr**(1M) command to create user accounts. Each user should have his own account. The **mkusr** command creates a new user on the system by prompting for information about the user and then updating the /etc/passwd and /etc/group files. **Mkusr** then builds the new user's home directory and sets up default files needed by Valid software tools and the UNIX shell.

You must be super-user to use the **mkusr** command.

```
# mkusr
```

Mkusr prompts you for the login name of the new user, the group name (default is "user"), the new user's full name, and the file system for the user's home directory (default is /u0). Note that you can enter the file system name with or without the "/". **Mkusr** then displays the information entered as well as the new userid and groupid numbers. The screen displays:

```
Is this new user information correct [y,n]?
```

If you type "n", you must re-enter all the user information. If you type "y", **mkusr** updates the password and group files and creates the new user's home directory and default files.

2.5 DEFINING PASSWORDS

If root has no password, anybody can successfully log in as root. This is dangerous, since root has privileges and powers not available to any other user. If inexperienced or

malicious users pose a problem for your site, you should give root a password, keep it secret, and change it regularly.

Root assigns a password to him/herself in the same way any other user does -- with the **passwd**(1) command. Log in as root and type

```
# passwd
```

The system responds

```
Changing password for root
New password:
```

Type the password. The **passwd** program asks you to type the password a second time for verification purposes.

Other users can change passwords by following the same procedure, except that the user logs in as him/herself, types **passwd**, and the system echoes "Changing password for <user name>", where <user name> is the user's login name. Ordinary users' passwords must be more than six characters long, and must contain at least one non-alphabetic character.

The super-user can change anybody's password; ordinary users can change only their own. For more information, see the man pages on **passwd**(1).

2.6 SETTING SYSTEM TIME/DATE

Set the time and date by booting from the Diagnostics diskette and selecting "SETUP". A less permanent method for changing the time and date is to use the "TIME" or "DATE" commands in DOS, or the **date** command in UNIX. These commands do not change the hardware time-of-day clock, so the time will revert back when you reboot the machine.

2.7 FILE TRANSFERS BETWEEN UNIX AND DOS

To copy files from DOS to UNIX, use the **ducp(1)** command:

ducp [**-a**] *DOS_file UNIX_file*

Ducp copies a DOS file to a Unix file. Use the option **-a** to copy ASCII text files, so that DOS \r\n sequences are converted to ASCII newline characters for compatibility with UNIX. **Ducp** recognizes ^Z as the end of a DOS file. With no option, no character conversion is done.

Ducp requires that both filenames be specified explicitly. To specify DOS pathnames from within UNIX, use two backslashes wherever you would ordinarily use one within DOS; this is necessary because the backslash \ is a special character for the shell. For example,

```
INCORRECT: ducp \test\file /usr/file
CORRECT:   ducp \\test\\file /usr/file
```

To copy files from UNIX to DOS, use the **udcp(1)** command:

udcp [**-a**] *UNIX_file DOS_file*

Udcp copies a UNIX file to a DOS file. Use the **-a** option to copy ASCII text files; it converts ASCII newline characters to \r\n sequences for compatibility with DOS. With no option, no character conversion is done.

Udcp requires that both filenames be specified explicitly. To specify DOS pathnames from within UNIX, use two backslashes wherever you would ordinarily use one within DOS; this is necessary because the backslash \ is a special character for the shell. For example,

```
INCORRECT: udcp /usr/file \new\file
CORRECT:   udcp /usr/file \\new\\file
```

SECTION 3
SYSTEM MAINTENANCE

This section describes backup and restore procedures for UNIX files. See the IBM manual *Disk Operating System Reference* for DOS backup and restore procedures. This section also covers system configuration and device names, adding a printer or plotter, and moving procedures.

3.1 BACKUP PROCEDURES

There are three levels of backup: monthly, weekly, and daily. Monthly backs up all files, weekly backs up files modified since the last monthly backup, and daily backs up files modified since the latest weekly or monthly backup.

There are five scripts

```
/etc/backup
/etc/backup.sys
/etc/backup.usr
/etc/backup.u0
/etc/backup.u1
```

and one C program

```
/etc/mkflplists
```

that comprise the backup system. The script */etc/backup.sys* backs up all the files that are NOT in the */usr* or */u0* trees. The script */etc/backup.usr* backs up all the files in the */usr* tree and */etc/backup.u0* backs up all the files in the */u0* tree. The script */etc/backup.u1* backs up all the files in the */u1* tree. The script */etc/backup* asks if you want to do each of the other backups.

You should be in super-user mode to back up files. This allows you to access all files. You can use a 1/4" tape drive or formatted floppy disks for backup. Be sure to have

enough disks before you start the backup. Each floppy holds about 1.2 Mbytes. To run a backup script, type the name of the script file after the super-user prompt, as in this example:

```
# /etc/backup
```

The script will prompt you for required entries. If you start a backup operation and want to stop, type ctrl-c to break out. If you do this, you must begin the backup all over again; you cannot pick up where you left off.

The scripts use **cpio**(1) to do the actual backup. Restoring is also done with the **cpio** command. (Note that the **-c** option is used when the backup is made.) Also note that **cpio** is unrooted from / for the sys backup, unrooted from */usr* for the usr backup, unrooted from */u0* for the u0 backup, and unrooted from */u1* for the u1 backup.

That is, path names in a *sys* backup would look like this

```
etc/backup
bin/sh
tmp/removeme
```

for paths

```
/etc/backup
/bin/sh
/tmp/removeme
```

Path names in a *usr* backup would look like this

```
lib/hpf
bin/awk
```

for paths

```
/usr/lib/hpf
/usr/bin/awk
```

And path names in a *u0* backup would look like this

user/mkfile
user/.profile
pdh/bigfile

for paths

/u0/user/mkfile
/u0/user/.profile
/u0/pdh/bigfile

The scripts record when the last backup was made by creating files in the */etc* directory. IT IS VERY IMPORTANT that nobody touches these files in any way that modifies the dates on them. These files are:

/etc/monthlydonesys
/etc/weeklydonesys
/etc/dailydonesys

/etc/monthlydoneusr
/etc/weeklydoneusr
/etc/dailydoneusr

/etc/monthlydoneu0
/etc/weeklydoneu0
/etc/dailydoneu0

/etc/monthlydoneu1
/etc/weeklydoneu1
/etc/dailydoneu1

Don't touch'um!

The backup scripts use the C program *mkflplists* to create lists of files contained on each floppy. As the backup proceeds, these lists are recorded in the following files:

/etc/monthlylistsys
/etc/weeklylistsys
/etc/dailylistsys

/etc/monthlylistusr
/etc/weeklylistusr
/etc/dailylistusr

/etc/monthlylistu0
/etc/weeklylistu0
/etc/dailylistu0

/etc/monthlylistu1
/etc/weeklylistu1
/etc/dailylistu1

In each of these files, a header of the form "list *n*" prefaces the list of the files in list number *n*. These lists correspond to the floppies used for backup. That is, list 1 shows the files on the first floppy, list 2 shows those on the second, and so forth. Here is an example of */etc/monthlylistu0*:

list 1

./pdh/foo
./pdh/bigfile

list 2

./pdh/verybigfile

list 3

./pdh/morefiles
./pdh/blahblah

NOTE: In the rare case where *./pdh/verybigfile* spans more than one floppy, the files in list 3 will not be on floppy 3; they might be on floppy 4 or even floppy 5.

3.2 RESTORING FILES

This section describes procedures for restoring UNIX files which were backed up with the backup scripts described in "Backup Procedures." Use **cpio** to restore files. Some examples appear below. See the man pages for **cpio** if you need more detailed information. (Remember that backup is done with the **-c** option of **cpio**.) When restoring files, use the lists from backup (e.g., /etc/monthlylistsys) to determine which floppy disk(s) you will need. (See the "Backup Procedures" section for the list file names.) If the files are on that floppy, the screen displays the names of the files as they are restored (if you use the "verbose" option, **-v**). If you simply get a prompt, then the files were not restored. If you want to list the names of the files on a floppy, use the following command:

cpio -iBct < /dev/rflpa

If you want to stop a restore operation, press ctrl-c to break out.

This example restores the file */u0/pdh/test1*:

cd /u0

(insert floppy)

cpio -iBduvcm pdh/test1 < /dev/rflpa

Note that because the backup was made from */u0* you don't include */u0/* in the pathname of the file to be extracted. Be sure that the file name format is an exact match. The file name should be specified exactly as it appears when you list the names of files on the backup floppy.

This example restores all the files in a directory tree */u0/pdh/bar*:

cd /u0

(insert floppy)

cpio -iBduvcm "pdh/bar/*" < /dev/rflpa

Note: You must put quotes (") around the argument *pdh/bar/** to prevent the shell from trying to evaluate the "*".

This example restores a list of files contained in the file */tmp/toextract* (you might more typically be using */etc/dailylistu0*):

cd /u0

(insert floppy)

cpio -iBduvcm 'cat /tmp/toextract' < /dev/rflpa

Note that the size of the file */tmp/toextract* is limited to 5120 characters. Also, the file names in /tmp/toextract can have meta-characters in them. The backquote character (') must be used as shown in the example.

Note on the meta-character "*":

In **cpio**, unlike the shell, "*" matches the slash character ("/"). If you use a "*" in your path specification you will generally get all sub-directories of that path.

3.3 SYSTEM CONFIGURATION

UNIX does all of its input and output through device files. These are files designating I/O devices such as disks, terminals, and printers.

Devices, by convention, live in the directory */dev*. The following document is an annotated listing of this directory (similar to those created by **ls -l**), explaining what the devices are and how they are used. See Appendix B for more information on devices.

DISKS

```
/dev/dsk:
drwxrwxr-x  2 root    sys        256 Oct  8 14:17 dsk/
br--------  1 root    sys     0,  0 Jul 21  1983 0s0
br--------  1 root    sys     0,  1 Jul 21  1983 0s1
br--------  1 root    sys     0,  2 Jul 21  1983 0s2
br--------  1 root    sys     0,  3 Jul 21  1983 0s3
br--------  1 root    sys     0,  4 Jul 21  1983 0s4
br--------  1 root    sys     0,  5 Jul 21  1983 0s5
br--------  1 root    sys     0,  6 Jul 21  1983 0s6
br--------  1 root    sys     0,  8 Jul 21  1983 1s0
br--------  1 root    sys     0,  9 Jul 21  1983 1s1
br--------  1 root    sys     0, 10 Jul 21  1983 1s2
br--------  1 root    sys     0, 11 Jul 21  1983 1s3
br--------  1 root    sys     0, 12 Jul 21  1983 1s4
br--------  1 root    sys     0, 13 Jul 21  1983 1s5
br--------  1 root    sys     0, 14 Jul 21  1983 1s6
```

```
/dev/rdsk:
drwxrwxr-x  2 root    sys          288 Oct  8 14:17 rdsk/
cr--------  1 root    sys        4,  0 Jul 21  1983 0s0
cr--------  1 root    sys        4,  1 Jul 21  1983 0s1
cr--------  1 root    sys        4,  2 Jul 21  1983 0s2
cr--------  1 root    sys        4,  3 Jul 21  1983 0s3
cr--------  1 root    sys        4,  4 Jul 21  1983 0s4
cr--------  1 root    sys        4,  5 Jul 21  1983 0s5
cr--------  1 root    sys        4,  6 Jul 21  1983 0s6
cr--------  1 root    sys        4,  7 Jul 21  1983 0s7
cr--------  1 root    sys        4,  8 Oct 24 12:58 1s0
cr--------  1 root    sys        4,  9 Jul 21  1983 1s1
cr--------  1 root    sys        4, 10 Jul 21  1983 1s2
cr--------  1 root    sys        4, 11 Jul 21  1983 1s3
cr--------  1 root    sys        4, 12 Jul 21  1983 1s4
cr--------  1 root    sys        4, 13 Jul 21  1983 1s5
cr--------  1 root    sys        4, 14 Jul 21  1983 1s6
cr--------  1 root    sys        4, 15 Jul 21  1983 1s7
```

Disks have two standard uses:

1. They are where "file systems" are mounted. UNIX keeps track of files, directories, and devices through file systems.

2. They are often used to keep large collections of unorganized or user-organized data such as databases.

The file names of disks are kept in */dev/dsk* and */dev/rdsk*. The first character of the name is the unit number, e.g., 0s0 refers to disk 0. The last character refers to the partition number. Disks are normally partitioned into areas which have specific uses. For example, the root file system might be mounted on */dev/dsk/0s0*.

The difference between dsk and rdsk is as follows: UNIX classifies devices into block and character devices. A block device is suitable for mounting file systems and has a very

stylized I/O protocol; a character device has a much more free-form interface (usually idiosyncratic to the particular device). Character versions of disk devices exist primarily for increased speed; programs such as **dump**(1M) and **fsck** use the character version of the file to read and write large amounts of data at one time.

FLOPPY

```
brw-rw-rw-  1 root  sys     0,112 Nov  1 15:09 flpa
brw-rw-rw-  1 root  sys     0,104 May  2  1985 flpb
crw-rw-rw-  1 root  sys     4,112 Nov 22 11:55 rflpa
crw-rw-rw-  1 root  sys     4,104 May  2  1985 rflpb
```

The floppy drive is accessed through the *[r]flpn* files. (These files may have the following names, depending on your system: *flpa*, *flpb*, *rflpa*, *rflpb*.) The **b** version of the file doesn't work; use only the **a** version. As with disks, there is a raw (character) version of the device which is usually faster. You can use the floppy to take backups (via the **backup**(1M) program) and transfer files (via **cpio**(1)). The **tar**(1) program can also be used to make floppies.

TAPE

```
crw-rw-rw   1 root  sys     18,1 Feb  1 1985 rmt/0m
crw-rw-rw   1 root  sys     18,0 Feb  1 1985 rmt/0mn
```

The system provides support for a 1/4 inch Techmar tape drive. The *0m* device rewinds automatically when the device is closed; the *0mn* device does not.

TERMINALS

```
crw--w--w-  3 root  sys    0,  0 Dec  5 16:42 console
crw--w--w-  3 root  sys    0,  0 Dec  5 16:42 syscon
crw--w--w-  3 root  sys    0,  0 Dec  5 16:42 systty
crw-rw-rw-  1 root  sys    2,  0 Dec  4 13:33 tty
crw--w--w-  1 root  sys    9,  0 Nov 26 13:14 tty0
crw--w--w-  1 root  sys    9,  1 May  2  1985 tty1
crw-rw-rw-  1 root  sys    9,128 May  2  1985 tty128
crw-rw-rw-  1 root  sys    9,129 May  2  1985 tty129
```

The console is the combination of keyboard and display that you normally use when entering data into the system. Tty0 and tty1 can be connected to serial devices such as terminals and printers. (Note: on EGA systems, one of these must be reserved for the use of the mouse). Tty is a virtual device: opening it opens "your terminal" no matter which terminal you are using. Syscon and systty are new in UNIX System V and can be used as virtual terminal interfaces, although this system doesn't use them that way. Tty128 and tty129 are like tty0 and tty1, except that opening them doesn't wait until carrier is established. (This is good for modems which are used for both call-out and receive.)

PRINTERS

```
crw-rw-rw-  1 root  sys   10,  0 May  2  1985 lp
crw-rw-rw-  1 root  sys   10,  1 May  2  1985 lp1
crw-rw-rw-  1 root  sys   10, 64 May  2  1985 glp
```

These files are connected to two Centronix parallel interfaces, which can be connected to printers, plotters, etc. Glp passes through characters unmodifed and is useful for sending data to plotters; lp and lp1 truncate data to 7 bits and convert line-feed to carriage-return + line-feed, and are good for sending to ASCII printers. These files are normally used by the **lp**(1M) system to support printing and plotting.

GRAPHICS

```
crwxrwxrwx  1 root   sys     14,  0 May  2  1985 vdi
crw-rw-rw-  1 root   sys     15,  0 Nov  5 12:00 display
crw-rw-rw-  1 root   sys     15,  0 Nov  6 22:43 gpj
```

VDI is used to access the implementation of VDI (Virtual Device Interface) graphics. The VDI must be installed in DOS for this interface to work. Display (and "gpj") are used by Valid's programs such as GED to do graphics. Only one will appear on your system.

NETWORK

```
crw--w--w-  1 root   sys     19,  0 Oct 28 17:08 ttyT0
crw--w--w-  1 root   sys     19,  1 Oct 28 17:07 ttyT1
crw--w--w-  1 root   sys     19,  2 Oct  7 18:31 ttyT2
crw-rw-rw-  1 root   sys     19,  3 Oct  7 18:31 ttyT3
crw-rw-rw-  1 root   sys     19,  4 Oct  7 18:31 ttyT4
crw-rw-rw-  1 root   sys     19,  5 Oct  7 18:31 ttyT5
crw-rw-rw-  1 root   sys     19,  6 Oct  7 18:31 ttyT6
crw-rw-rw-  1 root   sys     19,  7 Oct  7 18:31 ttyT7
crw-rw-rw-  1 uucp   sys     19,  8 Oct 29 11:28 ttyT8
drwxrwxrwx  2 root   sys     592 Nov  8 21:31 EXOS

/dev/EXOS:
crw-rw-rw-  1 root   sys     20,  0 Oct  7 18:31 admin
crw-rw-rw-  1 root   sys     20,  1 Oct  7 18:31 errlog
crw-rw-rw-  1 root   sys     22,  0 Oct 28 17:07 socket0
crw-rw-rw-  1 root   sys     22,  1 Oct 29 11:29 socket1
crw-rw-rw-  1 root   sys     22, 10 Oct 24 19:21 socket10
crw-rw-rw-  1 root   sys     22, 11 Oct 24 19:21 socket11
crw-rw-rw-  1 root   sys     22, 12 Oct 24 19:21 socket12
crw-rw-rw-  1 root   sys     22, 13 Oct 24 19:21 socket13
crw-rw-rw-  1 root   sys     22, 14 Oct  7 18:31 socket14
crw-rw-rw-  1 root   sys     22, 15 Oct  7 18:31 socket15
crw-rw-rw-  1 root   sys     22, 16 Oct  7 18:31 socket16
crw-rw-rw-  1 root   sys     22, 17 Oct  7 18:31 socket17
crw-rw-rw-  1 root   sys     22, 18 Oct  7 18:31 socket18
crw-rw-rw-  1 root   sys     22, 19 Oct  7 18:31 socket19
crw-rw-rw-  1 root   sys     22,  2 Oct 29 11:11 socket2
crw-rw-rw-  1 root   sys     22, 20 Oct  7 18:31 socket20
```

```
crw-rw-rw-  1 root    sys     22, 21 Oct  7 18:31 socket21
crw-rw-rw-  1 root    sys     22, 22 Oct  7 18:31 socket22
crw-rw-rw-  1 root    sys     22, 23 Oct  7 18:31 socket23
crw-rw-rw-  1 root    sys     22, 24 Oct  7 18:31 socket24
crw-rw-rw-  1 root    sys     22, 25 Oct  7 18:31 socket25
crw-rw-rw-  1 root    sys     22, 26 Oct  7 18:31 socket26
crw-rw-rw-  1 root    sys     22, 27 Oct  7 18:31 socket27
crw-rw-rw-  1 root    sys     22, 28 Oct  7 18:31 socket28
crw-rw-rw-  1 root    sys     22, 29 Oct  7 18:31 socket29
crw-rw-rw-  1 root    sys     22,  3 Oct 29 11:28 socket3
crw-rw-rw-  1 root    sys     22, 30 Oct  7 18:31 socket30
crw-rw-rw-  1 root    sys     22, 31 Oct  7 18:31 socket31
crw-rw-rw-  1 root    sys     22,  4 Oct 28 09:19 socket4
crw-rw-rw-  1 root    sys     22,  5 Oct 26 23:15 socket5
crw-rw-rw-  1 root    sys     22,  6 Oct 26 23:15 socket6
crw-rw-rw-  1 root    sys     22,  7 Oct 26 23:15 socket7
crw-rw-rw-  1 root    sys     22,  8 Oct 26 23:13 socket8
crw-rw-rw-  1 root    sys     22,  9 Oct 26 23:13 socket9
crw-rw-rw-  1 root    sys     21,  0 Oct 28 20:29 xmem
```

The network interface is managed by the Ethernet smart interface. The EXOS directory contains various special devices to manage that interface, none of which are directly useful to general users. Files named ttyT? are network virtual terminals which remote users use to log into the system.

MISCELLANEOUS

```
br--r-----  1 sys     sys     0,  0 May  2  1985 swap
cr--r-----  1 sys     sys     3,  0 Jul 21  1983 mem
cr--r-----  1 sys     sys     3,  1 Jul 21  1983 kmem
crw-rw-rw-  1 root    sys     3,  2 Dec  2 05:33 null
cr--r--r--  1 root    sys     8,  0 Jul 21  1983 error
crw-rw-rw-  1 root    sys    11,  0 May  2  1985 prf
crw-rw-rw-  1 root    sys    12,  0 May  2  1985 sxt
cr--------  1 root    sys    13,  0 May  2  1985 dos
```

Swap is a read-only interface to the swap file; programs such as **ps**(1) use it to extract information about processes. *Mem* is an interface to the physical memory of the system, and *kmem* is an interface to the kernel address space; these

are also used by programs to gather system status. *Null* is a convenient way to dump data down a hole (write) or give a program an empty file (read). *Error* is used by **errdemon**(1M) to log system error messages. *Prf* can be used to gather system profiling information. *Sxt* (Western Electric TTY-5620 window support) is not used by Valid. *Dos* is used to communicate with, you guessed it, DOS. Programs such as **dos**(1) and **ducp**(1) use it to manage the interfaces.

3.4 ADDING A PRINTER OR PLOTTER

To use the GED hardcopy command you must install the hardcopy software package. After the hardcopy package has been installed, the machine is set up correctly to do hardcopy to an Epson LQ-1500, FX-80, or FX-100 attached to parallel port number one.

In order to use an HP pen plotter or a remote plotter, you must use the UNIX System V lineprinter administration tools to configure the spooling sytem. The proceedures to do this will be explained in this section.

SETTING UP AN HP PEN PLOTTER

Valid supports a pen plotter connected to the IBM PC/AT via an RS-232 serial interface. The first RS-232 serial port (com1) is assigned to the mouse, so you must have a serial port configured as com2, which maps to the UNIX device */dev/tty1*, in order to use a pen plotter. See the IBM manual *Installation and Setup* for instructions to configure a serial port as com2.

The serial ports on the IBM PC/AT are Data Terminal Equipment (DTE). That is, they want to talk to Data Communication Equipment (DCE). The HP pen plotter also thinks it's DTE so you will need a null modem cable between them. The pen plotter should be set up to run at 1200 baud.

In order to let the lp spooling system know about the HP pen plotter, you must use the **lpadmin**(1M) command. (In this example, we assume an HP7475 plotter. You could also attach an HP7580):

Shut down the spooler:

```
# /usr/lib/lpshut
```

Add the HP plotter:

```
# /usr/lib/lpadmin -php7475 \
                   -v/dev/tty1 \
                   -i/usr/spool/lp/model/hp.pen
```

Restart the spooler:

```
# /usr/lib/lpsched
```

For more information on **lpadmin**, **lpshut**, or **lpsched** see the man pages for them in *UNIX System V, Volume III.*

Next you must tell the spooling system to enable and accept requests for the new printer:

```
# /usr/lib/accept hp7475

# /usr/bin/enable hp7475
```

The system is now ready to handle requests for HP pen plots.

REMOTE PLOTTING

To do remote plotting, you must have installed the Ethernet hardware and software as well as the hardcopy software.

Setting up remote plotting is not automated, so be sure to do the following steps carefully. You may want to read

"LP Spooling System" in the *UNIX System V System Administrator's Guide* before you get started, so that you have a better understanding of the spooling system.

Valid provides a template for you to use in creating a remote printer interface model: */usr/spool/lp/model/vers11*. This file is almost ready to use as a remote 11-inch Versatec model. This is how it looks before you modify it:

```
copies=$4

#skip the info on owner, login name, etc
shift; shift; shift; shift; shift

files="$*"
i=1
while [ $i -le $copies ]
do
   for file in $files
   do

     # for Valid release 7.27 or older hosts
     rsh machine_name "Vpr" \
       < $file 2>> /usr/spool/lp/vers11.err

     #for Valid release 8.0 (Berkeley 4.2) hosts
     #rsh machine_name "lpr -Pvers11" \
        < $file 2>> /usr/spool/lp/vers11.err
   done
   i=`expr $i + 1`
done
exit 0
```

In order for this file to work, you must replace *machine_name* with the name of the host you want to use to service plot requests. If you have an 8.0 host, you must comment out the 7.27 rsh line with a "#" and remove the "#" from the rsh line after the 8.0 comment. Here is an example of */usr/spool/lp/model/vers11* set up for an 8.0 host "host1":

```
copies=$4

#skip the info on owner, login name, etc
shift; shift; shift; shift; shift

files="$*"
i=1
while [ $i -le $copies ]
do
   for file in $files
   do

     # for Valid release 7.27 or older hosts
     # rsh machine_name "Vpr" \
        < $file 2>> /usr/spool/lp/vers11.err

     # for Valid release 8.0 (Berkeley 4.2) hosts
     rsh host1 "lpr -Pvers11" \
        < $file 2>> /usr/spool/lp/vers11.err
   done
   i=`expr $i + 1`
done
exit 0
```

OTHER REMOTE PRINTERS

In order to plot to other types of printers you must create other model files. Copy the vers11 model file to a new file named */usr/spool/lp/model/printer_name* and change the -P*printer_name* argument in the file. The following are the correct names for printers:

"vers11"	11" Versatec
"vers22"	22" Versatec
"vers36"	36" Versatec
"vers42"	42" Versatec
"Cvers42"	42" Color Versatec
"B9424"	24" Benson
"hp7580"	D size HP pen plotter
"hp7475"	B size HP pen plotter
"calcomp1043"	E size CalComp pen plotter
"calcomp5744"	E size CalComp electrostatic plotter

An example of the model file */usr/spool/lp/model/hp7580* for remote printing to a 7.27 host named "godzilla" follows:

```
copies=$4

#skip the info on owner, login name, etc
shift; shift; shift; shift; shift

files="$*"
i=1
while [ $i -le $copies ]
do
   for file in $files
   do

      #for Valid release 7.27 or older hosts
      rsh godzilla "cat > /dev/hp" \
        < $file 2>> /usr/spool/lp/hp7580.err

      #for Valid release 8.0 (Berkeley 4.2) hosts
      #rsh machine_name "lpr -Php7580" \
        < $file 2>> /usr/spool/lp/hp7580.err
   done
   i=`expr $i + 1`
done
exit 0
```

Note that "Vpr" was changed to "cat > /dev/hp" and "vers11" was changed to "hp7580". The name of the model file is */usr/spool/lp/model/hp7580.*

LP ADMINISTRATION FOR REMOTE PLOTTERS

After you have created a model file, you must use lp administration commands to let the lp spooling system know about the printer. In this example, we assume remote printing to an 11-inch Versatec:

Shut down the spooler:

```
# /usr/lib/lpshut
```

Add the vers11 plotter:

```
# /usr/lib/lpadmin -pvers11 \
                   -v/usr/spool/lp/vers11.err \
                   -i/usr/spool/lp/model/vers11
```

Restart the spooler:

```
# /usr/lib/lpsched
```

Tell the spooling system to enable and accept requests for the new printer:

```
# /usr/lib/accept vers11

# /usr/bin/enable vers11
```

Note that the device entry (**-v***device*) in the **lpadmin** command is the error file vers11.err because the model for vers11 does not send its output to standard out; it sends the data file to the remote print server. Also note that */usr/spool/lp/vers11.err* must exist before you do the **lpadmin** command. ("Touch" */usr/spool/lp/vers11.err.*)

To set up other plotters, use the same commands but change the plotter name "vers11" to the desired plotter (see "Other Remote Printers" above for a list of printer names).

CHANGES TO THE PRINT SERVER HOST

After doing the above, the IBM PC/AT system is ready to handle requests, but the host server is not. You must add the "lp" user to the server. The "lp" user should have a user id of 71 and a group id of 2. Thus, there should be an */etc/passwd* entry on the print server similar to this:

lp:*encrypted_password*:71:2:System V print spooler:/tmp:

You must also make sure that your IBM PC/AT machine's host name is in the following three files on the print server machine:

/etc/hosts
/etc/hosts.equiv
/.rhosts

After you have made the above changes to the print server you should be able to do remote plotting. (See the "S32 Network Administration Guide" for more information on how to change host files.)

TROUBLESHOOTING

Here are some things to check if you have trouble with hardcopy:

- If the file named */usr/spool/lp/member/gedplot* does not exist, then **spool.install** must be run. Extract */valid/ibin/spool.install* by inserting the plotting package disk and entering the following commands:

  cd /

  cpio -iBmducv valid/ibin/spool.install

  To run **spool.install**, stop the scheduler by typing:

  /usr/lib/lpshut

Then type:

/valid/ibin/spool.install

After you have run **spool.install** you can restart the scheduler by typing **/usr/lib/lpsched**.

- The **lp** spooling system's main error log is */usr/spool/lp/log*. This file contains a list of requests to print files, plus any error messages that the the scheduler (**lpsched**) generates.

- The program that converts GED files into files for a perticular type of printer, **/usr/lib/hpf**, has an error file in */usr/spool/lp* named **hpferrorlog**. This file contains the last 3K bytes of error messages from **hpf**. Each invocation of **hpf** starts with:

  New invocation of /usr/lib/hpf: <date>

  After the header, error messages (if any) will appear. Common problems are: can't find a body; permission problems; etc. Also, if there are bugs in **hpf** or GED there will be internal error messages that will help pinpoint the problem. If you include these internal error messages in a bug report, it will make it much easier for Valid to fix the problem.

- Each printer has an error file named *printer_name*.**err** in the directory */usr/spool/lp*. You should check the error file for the printer with problems.

- You can use the **lpstat(1)** command to find out other information about the spooling system. Refer to the **lpstat** man pages for details.

3.5 MOVING PROCEDURES

Refer to the IBM manual *Guide to Operations* Section 3, "Moving Your IBM Personal Computer AT."

SECTION 4
ETHERNET APPLICATION UTILITIES

The Ethernet TCP/IP software package includes eight network application utilities, in addition to five network systems utilities and the protocol software. The application utilities provide important, often-used functions.

The network application utilities are the following:

- **rlogin** – remote login
  Allows login to a remote UNIX or UNIX-derived host on the network.

- **rsh** – remote shell
  Allows execution of a single command on a remote UNIX or UNIX-derived host.

- **rcp** – remote copy
  Allows file transfer between UNIX and UNIX-derived hosts on the network.

- **rwho** – remote who
  Displays current users on remote UNIX or UNIX-derived hosts.

- **ruptime** – remote uptime
  Displays status of remote UNIX or UNIX-derived hosts.

- **ftp** – file transfer utility
  Allows file transfer between UNIX, UNIX-derived hosts and non-UNIX hosts on the network.

- **telnet** – virtual terminal utility
  Allows use of a terminal on a local host as a virtual terminal connected to a remote host.

The following sections describe these utilities in a user guide style. The man pages in *UNIX System V, Volume II* provide a detailed, formal description of each utility.

4.1 RLOGIN – THE REMOTE LOGIN UTILITY

The **rlogin**(1) utility allows you to login to a remote UNIX or UNIX-derived system on the network. You can login to any remote UNIX host listed in the *hosts* file. The connection is a virtual terminal connection and has all the characteristics of a remote, normal, tty-driven physical terminal connection. Echoing, flow control (using CTRL-S and CTRL-Q), and signal processing are all implemented.

To login to a remote system, type a command in the form

rlogin *remote_host*

where *remote_host* is the name of the remote system. Once you have typed the **rlogin** command line, you may or may not be prompted for a password, depending on the *hosts* and *.rhosts* files on the remote system.

Once you are logged in to the remote system, your session proceeds as it normally would on the local system except that all the commands you type are interpreted by the remote system.

To terminate a remote login session, type a tilde (~) followed immediately by a period and a carriage return:

~.

You can change the local escape character – the tilde – to any other character using the **– e** option of **rlogin.** For example, if you are logging in to the remote system "india" as the user "boojum" and you want your escape character to be an asterisk, you would type

```
% rlogin india – e \* – l boojum
```

Note, however, that you have to escape (that is, precede with a backslash or place in apostrophes) the new escape character when you use it since it has a special meaning to the shell.

4.2 RSH – THE REMOTE SHELL UTILITY

The **rsh**(1) utility executes a single command on a remote UNIX, or UNIX-derived system. This command performs its standard I/O on the PC and runs under the local user ID name. Because it does not request a password, you can only execute commands on other systems with **rsh** if the remote system lists your PC in its /etc/hosts.equiv file or if your PC is listed in the remote user account's .rhosts file.

In almost all cases, **rsh** is used to execute a single command. Although it can invoke an interactive shell, it cannot handle programs that modify **tty** modes, such as the full-screen editor vi. **rsh** is also useful for applications such as print spooling, since it can pipe I/O streams between systems.

To execute a command on a remote system, type a command in the form

rsh *remote_host command*

or

rsh *remote_host* **– l** *user_name command*

remote_host is the remote system on which *command* is to be executed. *command* can be any command the remote system recognizes, including any options that may be required. The – l option allows you to login to an account with a different name than the one you are currently logged in as on the local system. *user_name* is the login name of a user on the remote system if you are logging in to an account with a name different from your current one on the PC host. The **rsh** command will fail if a password is required for *user_name.*

You can redirect the output of the **rsh** command to a file on the local host. For example, to redirect the results of the **cat** command on the remote system "india" to the file my_local_file, you might type

% **rsh india cat** *remote_file_list* > **my_local_file**

Similarly, you can redirect input from the PC to the **rsh** command. The general format of this is as follows:

rsh *remote_host command* < *local_file*

Note that in all **rsh** commands, any metacharacters (such as the > and < redirection symbols) are interpreted by the PC, not by the remote host. To have metacharacters interpreted by the remote host, they must be quoted (enclosed in apostrophes). For example, to redirect the output of the **cat** command shown above to a file on the remote system, you might type

% **rsh india 'cat** *remote_file_list* > **my_remote_file'**

Any remote filenames that you include in the **rsh** command are interpreted relative to your remote login directory, unless they begin with a slash (/), in which case they are interpreted relative to the root directory on the remote system.

After the command indicated on the **rsh** command line completes execution, control returns to the host PC.

4.3 RCP – THE REMOTE COPY UTILITY

The **rcp**(1) utility copies files and directories from one UNIX or UNIX-derived system to another over the Ethernet. This command is similar to the UNIX command **cp**(1). You can copy files between systems via **rcp** only if the /hosts.equiv file on the remote host contains an entry for your PC or if both the user's home directories on the local and remote hosts contain entries in the .rhosts file.

To copy a file from the PC to a remote host, type a command in the following format:

rcp *local_file remote_host:remote_file*

local_file is the name of the file on the PC. *remote_host* is the name of the system the file is being copied to; it must be followed immediately by a colon (:). *remote_file* is the name the file is to have on the remote system. If the system is listed in the .rhosts file, remote_host should be specified as *remote_host.user_name*, where *user_name* is the name of the user whose account contains the .rhosts file.

To copy a file from a remote host to the PC, type a command in the following format:

rcp *remote_host:remote_file local_file*

remote_host:remote_file specifies the name of the remote system and the file to be copied. *local_file* is the name of the file to which the remote file is to be copied.

If you use metacharacters (such as *, ?, [, and]) for filename expansion when specifying the name of the file, it is expanded by the PC. If you want the filename expanded by the remote host, you must quote it (that is, place it in apostrophes or precede the metacharacter with a backslash).

4.4 RWHO – THE REMOTE WHO UTILITY

The **rwho**(1) utility is similar to the UNIX **who**(1) utility. **who** displays the login name, terminal name, and login time for each user currently logged in to your PC. **rwho** displays the same information for your PC and for all remote UNIX or UNIX-derived hosts on the system along with the names of the remote systems.

To display a list of all users logged in to UNIX and UNIX-derived hosts on the network, type

```
% rwho
```

The list of users displayed is similar to this one:

```
albert    excelan:tty15      Jul  3 08:17
arnold    excelan:ttyp4      Jul  3 09:48 :25
boojum    ma:tty03           Jul  3 09:38
manny     ca:ttyp2           Jul  3 09:10 :07
root      isfront:console    Jul  2 16:38 :40
```

The first column gives the name of the user. The second column indicates the system the user is logged into and the number of the terminal they are logged in from. The remainder of the line gives the date and time the user logged in, in the format *month date hour:minute:second.*

4.5 RUPTIME – THE REMOTE UPTIME UTILITY

The **ruptime**(1) utility displays status information about all UNIX or UNIX-derived hosts on the network, including the current time, and how long the system has been up.

To display a status list of all UNIX systems on the network, type

```
% ruptime
```

The status list is similar to this one:

```
ca          up 12+ 18:54,       4 users
india       down 18+ 13:24
isbox       down  5+ 19:53
isfront     up ??:??,           1 user
ma          up  5+ 02:45,       6 users
```

In this status list, the first column specifies the name of the system. The second column shows whether the system is up or down and for how long (in the format *days+ hours:minutes*) it has been up or down, if known. The third column shows the number of users currently logged in to the system.

4.6 FTP – THE FILE TRANSFER UTILITY

The **ftp**(1) utility is a client program that uses the DARPA Internet standard File Transfer Protocol (FTP) to transfer files between your PC and a remote host.

In addition, **ftp** can access directories/files on a remote host and allow you to perform usual operations, such as list and change working directories, list files at various levels, and rename directories and files.

While the local system is (obviously) a UNIX system, the remote host can be any system – UNIX or otherwise – that supports the FTP protocol. When communicating with a remote host that is running UNIX or a UNIX derivative, the remote host's FTP-server program **ftpd** is the responding entity. Most implementations of TCP/IP support the FTP protocol.

Prerequisites for using the **ftp** include the following:

- Both the local and remote host support ARPANET's standard File Transfer Protocol.
- The FTP server program is running on the remote host.
- File specification for the remote host is done according to the remote host's conventions.

All **ftp** operations are performed by first invoking the utility and then executing various **ftp** commands.

This section describes the **ftp** invocation and remote login procedure and also gives examples of usage of some of the commands. The man pages for **ftp(1)** give a comprehensive, formal description of the **ftp** utility, its options, and commands. The man pages for **ftpd(1M)** also provide a formal description of the FTP server **ftpd** program. (The **ftpd** program is the FTP server program supplied as part of the Ethernet TCP/IP software package.) The FTP server program description is included for completeness; normally, a user or even a system administrator will not need to use

it explicitly except for ensuring that it is running on the systems that are to be accessed for file manipulation.

INVOCATION OF FTP AND REMOTE LOGIN

To illustrate the invocation of **ftp** and login to a remote host, let us assume that a user "dave" on the local system "finance" needs to login to remote host "warehouse." "dave" can login as himself (by entering his name) or he can login as any other user (let us say "mark"). In either case, the user must be a recognized user of the remote host. Furthermore, the user should provide a password if required by the remote host. (The password requirements are set by the remote host, which may or may not require typing a password.)

"dave" can invoke **ftp** and login to the remote host "warehouse" by using one of the two methods, shown below as command sequences. The numbers indicate the response from the **ftp** server.

Method 1:

```
% ftp
ftp> open warehouse
Connected to warehouse.
220 Warehouse FTP Server (<version> <date>) ready.
Remote user name: mark
331 Password required for mark.
Password: password
220 warehouse FTP server (<version> <date>) ready.
ftp>
```

Method 2:

```
% ftp warehouse
Connected to warehouse.
220 Warehouse FTP Server (<version> <date>) ready.
Remote user name: mark
331 Password required for mark.
Password: password
220 warehouse FTP server (<version> <date>) ready.
ftp>
```

In both cases the local system enters the **ftp** command mode.

In Method 1, the command mode simply displays the "ftp>" prompt. Then, in response to the user input **open warehouse**, the command mode displays the "Remote user name" prompt line.

In Method 2, the command mode implicitly interprets and executes the **open warehouse** command and then displays the "Remote user name" prompt line.

In response to the "Remote user name" prompt, "dave" could have entered his own name. In the present case, however, dave enters **mark**. The system then displays the "Password" prompt line. After dave enters the password, the system returns to the command mode, displays the prompt "ftp>," and awaits further user input.

USING FTP COMMANDS

After you have logged in to a remote host as described above, you can use **ftp** commands for such operations as change working directories, list directories, remove files, rename files, and copy/append files. This section illustrates the use of only a few of the **ftp** commands. A comprehensive, formal description of all the **ftp** commands is provided in the man pages for **ftp(1)**. Table 4-1 lists all **ftp** commands by function.

NOTE

All **ftp** commands are interactive. That is, if you enter a command without the required parameter(s), the system prompts you for the parameter(s).

The **help** command provides a brief description of each **ftp** command. It can also be used for listing all the available **ftp** commands. The format for using the **help** command is as follows:

ftp> **help** *command*
ftp> **help**

The first command displays information for the specified *command.* The second command lists all the available **ftp** commands.

Some commands simply set or reset various **ftp** options in a toggle fashion. That is, if an option is "on," executing the command sets the option to "off." Executing the command a second time resets the option to on. For example, when you first login, the *verbose* option, which gives detailed messages, is on by default. You can turn the *verbose* option off by typing the command

ftp> **verbose**

You can reset the verbose option to on by executing the above command one more time.

The state of all **ftp** options at any given time can be displayed by typing the command

ftp> **status**

The above command displays the options status in the following format:

```
Mode: Stream;      Type: ascii;  Form: non-print;
Structure: file;  Verbose: on;  Bell: off;
Prompting: off
```

Table 4-1: FTP Commands Grouped by Function

Group/ Function	Command
Invoking/Quitting	
Establish connection to remote host's server	open
Terminate FTP session, but remain in FTP command mode	close
Terminate FTP session and return to operating system	bye, quit
Directory Operations	
Change current working directory	cd
Copy remote directory information to local file	dir
Change current working directory to local directory	lcd
Display contents of local directory	ldir
Display short form of directory information for local directory	lls
Display current local working directory	lpwd
Display short form of directory information for remote directory	ls
Write directory information for remote file(s) into local file	mdir
Create remote directory	mkdir
Write directory information for remote files to local file	mls
Display name of current remote directory	pwd
Delete remote directory	rmdir
File Operations	
Delete remote file	delete
Delete remote file(s)	mdelete
Rename remote file	rename
File Transfer	
Append local file to remote file	append
Copy remote file to local file	get, recv
Place remote file(s) into local working directory	mget
Copy local file(s) to remote working directory	mput
Copy local file to remote working directory	put, send

Table 4-1: FTP Commands by Function (Continued)

Group/ Function	Command
File Transfer Parameters	
Set file transfer form	form
Set file transfer mode	mode
Set file transfer structure	struct
File Transfer Data Format	
Set file transfer type to ASCII	ascii
Set file transfer type to binary	binary
Display or set file transfer type	type
Modifying FTP Environment	
Sound bell on completion of each command	bell
Toggle debugging mode	debug
Toggle interactive prompting	prompt
Toggle verbose mode	verbose
Miscellaneous Commands	
Toggle filename globbing	glob
Display on-line help documentation	help, ?
Send arbitrary FTP command to server	quote
Display help documentation from remote host	remotehelp
Toggle use of port commands	sendport
Display current status of FTP options	status
Identify a user to remote host	user

MANIPULATING FILES

The main function of the **ftp** utility is to transfer files between your PC and a remote host. However, it allows you to perform several additional operations such as list/change directories and/or list files of the remote host before actually transferring the files.

The following command lists all the directories and files in the current working directory of the remote host:

ftp> **dir**

You can change the current working directory on the remote host by the following command:

ftp> **cd** *remote_directory*

You can change the working directory on your PC by using the following command. This command works whether or not you have established a connection with a remote server.

ftp> **lcd** *local_directory*

If *local_directory* is not specified, the current working directory changes to your home directory.

get and **put** are the two basic commands for transferring files between your PC and a remote host. The **get** command copies files from the remote host to your PC. The **put** command copies files from your PC to the remote host. The format of these commands is as follows:

ftp> **get** *remote_file* [*local_file*]
ftp> **put** *local_file* [*remote_file*]

The **append** command appends a copy of file on your PC to a file on a remote host. The syntax for this command is as follows:

ftp> **append** *local_file* [*remote_file*]

If *remote_file* is omitted from the **put** and **append** commands, the local file is copied or appended to a file on the remote host that has the same name as the file on your PC. Conversely, if *local_file* is omitted from the **get** command, *remote_file* retains its original name when copied to your PC. Whether they are specified explicitly or not, *local_file* in the **get** command and *remote_file* in the **put** and **append** commands must be legal filenames on both the local and remote hosts.

USING FTP COMMANDS BY REDIRECTING INPUT OR FROM A SHELL SCRIPT

You usually enter **ftp** commands from the keyboard. However, you can specify a predefined **ftp** session by redirecting standard input interactively or from within a shell script. The only difference between these **ftp** sessions and a normal session is that autologin is disabled when **ftp** takes command from a file, so you have to use the **user** command to login.

The following example shows an interactive session that uses a predefined command script, in this case in the file *ftp.com*, to enter **ftp**:

% **ftp < ftp.com**

The contents of *ftp.com* might be the following:

```
user boojum          /* the username */
pass snark           /* the password */
get x.c              /* an ftp command */
bye                  /* conclude ftp session */
```

The following example shows a C shell script that can be called to run **ftp**:

```
#ftp C shell script
ftp remote  <<  EOF
user boojum
pass snark
get x.c
bye
EOF
# continue with any other commands
```

LOGGING OUT

Once you have finished the **ftp** session with a remote system, you can either break the connection with (logout of) the remote system while still remaining in **ftp** or you can logout, exit **ftp**, and return to UNIX on your PC.

To logout of the remote system while staying in **ftp**, enter the following command. After this, you can login to another remote system without having to re-invoke the **ftp** utility.

ftp> **close**

To logout, exit the **ftp**, and return to UNIX, enter the following command:

ftp> **quit**

4.7 TELNET - THE VIRTUAL TERMINAL UTILITY

The **telnet**(1) utility lets you emulate a virtual terminal connected to a remote host. You can connect to any remote host on the network that supports the TELNET protocol and perform all operations as if you were using a terminal physically connected to the remote host. (The **telnet** utility differs from the **rlogin** utility in that **telnet** lets you connect to any system on the network, while **rlogin** only lets you connect to other UNIX or UNIX-derived systems.)

INVOCATION OF TELNET AND LOGGING IN

To initiate a virtual terminal connection, you first invoke the **telnet** utility and then establish a connection to the remote host. This requires execution of two simple commands. Alternatively, you can perform this operation by executing a single command that includes the *remote_host* as an argument.

The two methods to "connect" your system to *remote_system* are given below as command sequences.

Method 1:

```
% telnet
(to) remote_host
Connected to <remote_host>
Escape character is "^]"
```

Method 2:

```
% telnet remote_host
Connected to <remote_host>
Escape character is "^]"
```

In Method 1, the first command invokes the **telnet** utility; the second command connects to *remote_host*.

In Method 2, the first command invokes the utility and connects to *remote_host*.

Following connection to the remote host, you can login to the remote host in the normal manner: enter the user name and, when prompted, enter the password. After that you can proceed as though you were directly connected to *remote_host*. Of course, to perform these operations, you need to use the syntax conventions native to *remote_host*.

USING TELNET COMMANDS

This section demonstrates the use of some of the **telnet** commands. A comprehensive, formal description of the TELNET utility and its commands is provided in the man pages for **telnet(1)**.

Telnet commands can be executed only when **telnet** is in command mode. (This mode is indicated by the presence of "telnet>" prompt.) **telnet** can be forced into command mode by entering the current escape character on the command line. The default escape character is "^]" (CTRL-]), which means, while holding the "CTRL" key, press the "]" key. An example of forcing the **telnet** into command mode is shown below.

```
% ^ ]
telnet>
```

You can find out the name of the remote host you are connected to by using the **status** command, as follows:

```
% ^ ]
telnet> status
Connected to <remote_host>
Escape character is '^ ]'
```

You can change the escape character by using the **escape** command, as follows. Let us assume that you want to change the default escape character ^] to ^ A.

```
% ^ ]
telnet> escape
new escape character: ^ A
Escape character is '^ A'
%
```

You can display on-line help information by using the "?" command, as illustrated below. Let us assume you want information about the **status** command.

```
% ^ ]
telnet> ? status
Print status information
telnet>
```

If the "?" command is entered without any argument, **telnet** lists all its available commands.

LOGGING OUT

Once you have finished the **telnet** session with a remote host, you can break the connection with (logout of) the remote host and return to the operating system on the local host by entering the following command:

```
telnet> quit
```

4.8 ETHERNET UTILITY ERROR MESSAGES

The **ftp** and **telnet** network application utilities provided with the Ethernet TCP/IP software return two types of error messages. Those preceded by two phrases separated by a colon are UNIX errors. For an explanation of these messages, refer to the UNIX programmer's manual. Error messages not generated by the operating system are due to a malfunction of either the network software or the utility itself. Most of the error messages result from problems with the network software and should never appear when you are using the utilities. If they do appear, however, they indicate a serious problem that should be resolved by the system administrator.

This appendix lists and explains the error messages that may be displayed when you are using **ftp** and **telnet**.

FTP ERROR MESSAGES

Error messages that appear while you are using **ftp** may either originate from the remote host or from the local host.

Error messages that are preceded by three digits originate from the remote host. They indicate that an attempt was made to perform an invalid operation on the remote host. These messages should be self-explanatory; for further explanation, consult the documentation for the remote system.

The error messages that originate from the local host are listed in alphabetical order and explained below. If any other messages appear that are not self-explanatory and that interrupt your work, contact your system administrator.

Error Message	Explanation
Already connected to ..., use close first.	You must close the existing connection before you can open another.
Ambiguous command	The command you typed can be interpreted in more than one way. Use the **help** command for a list of valid commands.
Connection refused	The remote server is too busy or it does not have the proper server software. In the first case, try establishing the connection later. In the second case, contact your system administrator or the administrator of the remote system.
Connection reset by peer	Something probably went wrong at the remote end of the connection and the remote system detected it. Contact your system administrator or the administrator of the remote system.
Connection timed out	The attempt to establish communication with the remote host did not succeed. Most likely, the remote host is down or the path to it has been severed. Try again later or contact your system administrator.
Invalid command	The command you typed was not valid. Use the **help** command for a list of valid commands.
Login failed	The name and/or password you supplied to the remote system were not acceptable.

Network dropped connection on reset or Lost connection	Communication to the remote host was interrupted and lost, and the state of the operation that was in progress, if any, cannot be determined. Try the operation again.
Network is down	The local host cannot communicate with the network. Try again later.
Network is unreachable	No route to the remote host and/or the remote network is known. Contact your system administrator.
No route to host	There is no route to the remote host. Contact your system administrator or the administrator of the remote host.
No such file or directory	You attempted to access a nonexistent file or directory.
Not connected	File operations cannot be performed until you have established a connection to the remote host. Use the **open** command to do this.
Not sufficient privilege for operation or Permission denied	You attempted an operation for which you do not have the proper permission(s).
... : Unknown host	The host you specified is not in the hosts database, the file that lists the remote systems that you can connect to from your system. Contact your system administrator.
usage: ...	You specified options incorrectly on the command line. This error message shows the correct command line format. For more information, refer to the man pages for **ftp**.

TELNET ERROR MESSAGES

All error messages that occur when you are using the **telnet** command originate from the local host. They are listed in alphabetical order and explained below. If any other messages appear that are not self-explanatory and that interrupt your work, contact your system administrator.

Error Message	Explanation
Connection refused	The remote server is too busy or it does not have the proper server software. In the first case, try establishing the connection later. In the second case, contact your system administrator or the administrator of the remote system.
Connection reset by peer	Something probably went wrong at the remote end of the connection and the remote system detected it. Contact your system administrator or the administrator of the remote system.
Connection timed out	The attempt to communicate with the remote host did not succeed. Most likely, the remote host is down or the path to it has been severed. Try again later or contact your system administrator.
Host is down	The remote host is not functioning. Contact your system administrator or the administrator of the remote host.
Invalid command	The command you typed was not valid. Use the **help** command for a list of valid commands.
Network dropped connection on reset	Communication to the remote host was interrupted and lost, and the state of the operation that was in progress, if any, cannot be determined. Try the operation again.

Network is down	The local host cannot communicate with the network. Try again later.
Network is unreachable	No route to the remote host and/or the remote network is known. Contact your system administrator.
No route to host	There is no route to the remote host. Contact your system administrator or the administrator of the remote host.
Not sufficient privilege for operation or Permission denied	You attempted an operation for which you do not have the proper permission(s).
Software caused connection abort	Something probably went wrong at the remote end of the connection and the local system detected it. Contact your system administrator.
... : Unknown host	The host you specified is not in the hosts database, the file that lists the remote systems that you can connect to from your system. Contact your system administrator.
usage: ...	You specified options incorrectly on the command line. This error message shows the correct command line format. For more information, refer to **telnet(1)**.

SECTION 5
TROUBLESHOOTING

If you are having problems with your Valid Designer PC/AT you should look through this section for the solution. We have compiled the following list of frequently encountered problems and their solutions. If you run into a problem that you think should be on our list please let us know.

Problems found during installation of your machine are most often caused by misreading the installation instructions. Please review the installation manual to verify that you have not skipped a step or performed a step out of order. Have someone else check your work.

If you need to repair the UNIX file system, see the UNIX System V documentation on **fsck**, the file system check program.

If you suspect a hardware failure is your problem, please refer to the IBM manual *Guide to Operations*, Section 2 "Testing your IBM Personal Computer PC/AT."

5.1 SYSTEM AND DISK INSTALLATION

Cover Problems:

Problem: Five screws have been removed but the cover still won't come off.

Explanation: The cover slides toward the front of the PC/AT and has a key lock.

Solution: Unlock the cover and pull towards the front of thePC/AT. Be careful not to catch or pinch the disk cables with the cover.

New Disk Problems:

Problem: Boot seems to hang during memory check after installing a new disk.

Explanation: Telling SETUP that you have a new disk causes the boot procedure to go look for it. A new unformatted disk will not respond. Eventually this will time out.

Solution: Wait a little longer. (About 2-3 minutes total.) If you are installing an unformatted disk be sure you add the OPDISK command to your *config.sys* file before you run FMT.

Problem: "Bad format: block error" displays during a hard disk format.

Solution: Look at the "d.bad" bad tracks created with **edfmt** for errors. Retry the format.

Problem: "Recalibrate Failed" displays during a hard disk format.

Solution: Check all disk cable connections, disk switch settings, and check SETUP. Check for a terminator pack on disk drive D. Remove it if present.

5.2 POWER ON AND BOOTING DOS

Setup Problems:

Problem: When you are booting up your machine you get the message: "162-System Options Not Set (Run SETUP)" even though you have already run SETUP.

Explanation: The hardware in your machine and the information you gave to the SETUP program do not match.

Solution 1: Check that the "Display Switch" on the "System Board" is set properly. (See RED IBM binder: Installation and Setup - "Internal Option Installation" - "Color Graphics Monitor Adapter" pp. 3-26)

Solution 2: Check again how much memory you have installed. (See IBM Chart 2, "Installing Your Internal Options" Step 12.)

Parity Problems:

Problem: When you are re-booting the machine you get the message:

PARITY CHECK 2
?????

Explanation: UNIX on the coprocessor was not shut down properly.

Solution: Cycle the power on your PC/AT.

5.3 IN DOS

Time Problems:

Problem: You set the time using the TIME or the DATE command, but the time is wrong when you next switch on the machine.

Explanation: Only the software clock is set by these commands.

Solution: Run the SETUP program on your Diagnostics diskette to set the hardware clock.

5.4 BOOTING UNIX

Kernel Panic Problems

Problem: You type UNIX from DOS and after the auto test you get the message "Kernel Panic".

Explanation: Ethernet board is in incorrect location in memory.

Solution: Recopy \XLN\HARDWARE\EXCELAN.HDW from the Ethernet package disk. Remember, Ethernet will not work when there is a monochrome card in the PC/AT.

Other Problems

Problem: "Opdisk: drive init failed" displays.

Solution: Press <Alt>+<Ctrl>+<Del> simultaneously to reboot. This is not a fatal error, so rebooting is sufficient.

Problem: "OPUS5: coprocessor segment not found" displays during **vinit**.

Solution: Check the co-processor board installation, and look for an incorrect switch setting on the board.

5.5 IN UNIX

csh Problems:

Problems: Typing "**csh**" generates the message "csh: Command not found".

Explanation: **Csh** is not supported in UNIX System V.

Solution: Wait until Valid's **csh** is released.

Time Problems:

Problem: You set the time using the **date** command, but the time is wrong when you go back to DOS and the next time you boot the machine.

Explanation: Only the UNIX software clock is set by this command.

Solution: Run the SETUP program on your Diagnostics diskette to set the hardware clock.

SECTION 6
SOFTWARE INSTALLATION

This chapter describes the software installation procedures for the Logic Designer AT with the GX display. The software installation includes the following tasks:

- Installing the DOS resident software
- Configuring Unix
- Installing Unix and GED on the co-processor
- Installing additional SCALD software
- Installing ethernet software (optional)
- Preparing the system for use

NOTE: If you are reinstalling the system and don't want to reconfigure the disks, begin with Section 6.4, "Installing Unix and GED."

6.1 ORGANIZATION OF THIS CHAPTER

This chapter is organized sequentially and is intended to take you through the installation from start to finish. There are several places where the installation procedure is dependent on the particular configuration (number and type of disks, whether Ethernet is to be installed, etc.). Each configuration is documented as a separate subsection. The configurations described are:

1. One 20Mbyte (C) and one 85Mbyte (D) disk
2. One 85Mbyte (C) disk
3. Two 85Mbyte (C and D) disks

which are referred to as 20/85, 85, and 85/85. If a section

heading is followed with a configuration in parentheses, such as (20/85), that section only applies to installations with that configuration.

NOTE: This installation procedure should be followed in the order presented.

6.2 INSTALLING THE DOS RESIDENT SOFTWARE

This section describes how to install the DOS resident programs as well as the DOS operating system itself.

Parts Needed

1. Valid DOS Master Installation Diskette.

INSTALLATION INSTRUCTIONS

1. Put the DOS Master Installation diskette into the floppy drive. Reboot the PC/AT by pressing <Ctrl>+<Alt>+<Del> simultaneously.

2. Type "a:install".

3. **Install** will check to be sure that **setdisk** was properly run. If **setdisk** was not properly run, **setdisk** will be run again.

4. The display will show the current disk configuration, such as "I20F85" for IBM 20Mbyte C disk and Fujitsu 85 Mbyte D disk. "F85T85" shows a Fujitsu 85Mbyte C disk and Toshiba 85Mbyte D disk.

5. The Valid DOS software and IBM PC-DOS software is now installed. This takes several minutes.

6. Remove the Valid DOS Master diskette from the disk drive, and reboot the machine (press <Ctrl>+<Alt>+<Del>).

7. If the message "opdisk:drive init failed" is displayed, reboot the machine again by pressing <Ctrl>+ <Alt>+ <Del>.

6.3 CONFIGURATION OF UNIX

This section describes how to configure the Unix operating system for the co-processor board.

CONFIGURATION INSTRUCTIONS

1. If you have not entered this section from the previous installation step, remove any diskettes from the floppy drive. Then reboot the machine by pressing "<Ctrl>+ <Alt>+ <Del>" simultaneously.

2. If you are in the Pacific time zone, go to step 4. If you are not in the Pacific time zone, you must edit three files (*unix.cfg*, *opus.cfg*, and */etc/timezone*) to specify your timezone. Time zones in the continental United States are in the following formats:

   tz=PST8PDT if the timezone is Pacific.
   tz=MST7MDT if the timezone is Mountain.
   tz=CST6CDT if the timezone is Central.
   tz=EST5EDT if the timezone is Eastern.

   For time zones outside the continental United States, use this format:

   tz=xxxnzzz

   where *xxx* is the standard local time zone abbreviation, *n* is the difference in hours from Greenwich Mean Time (GMT), and *zzz* is the abbreviation for the daylight-saving local time zone, if any. Note that *n* can be two digits, and it is negative in areas east of GMT and west of the international date line.

a. Type "cd valid".

b. Type "edlin unix.cfg".

c. Type "1" (the number one).

d. Overstrike the timezone definition ("tz=PST8PDT") to reflect your timezone.

e. Type "e".

f. Repeat steps b through e for *opus.cfg*.

g. Edit the line "TZ=PST8PDT" to your time zone in */etc/timezone*.

Here is an example where user entries are shown in **bold** type:

```
C:> cd valid
C:VALID > edlin unix.cfg
End of input file
*1
   1:*[tz=PST8PDT]
   1:*[tz=CST6CDT]   <--- user types this
*e
C:VALID >
```

h. Don't be alarmed if the configuration doesn't seem right when you run "vinit". The timezone will be correct when you boot the finished installation.

3. Type "vinit".

4. Ignore the following messages if they appear:

```
Opus5: Unknown device: gpj
Opus5: Unknown device: ether
```

5. The configuration program begins. It displays a welcome message and waits for any response. Type any character.

6. The co-processor board-level tests now execute.

7. Type "1" (the number one) when asked for the system type (PC/AT).

8. Reconfirm PC/AT as the system by typing "y".

9. Enter the time zone when prompted.

10. The Unix device configuration is printed. Note that the entries for SPECIAL, TTY and MISC may not be displayed. Do not worry about this, and do not attempt to add the devices.

11. If the system configuration is **20/85**, the Unix device configuration shown should be:

```
PARAMETER[tz=xSTnxDT][i=x][s=?]
    [a=128]

SYSTEM<clock=clock><dos=dos><console=console>

DISK<dsk/0=d:><flpa=a:><flpb=b:>

TTY<tty0=com1><tty1=com2>

LP  <lp=lpt1><lp1=lpt2>

SPECIAL<vdi=vdi>

MISC<gpj=gpj>
```

12. If the system configuration is **85/85**, the Unix device configuration shown should be:

```
PARAMETER[tz=xSTnxDT][i=x][s=?]
    [a=128]

SYSTEM<clock=clock><dos=dos><console=console>

DISK<dsk/0=d:><dsk/1=c:><flpa=a:>
    <flpb=b:>
```

TTY<tty0=com1><tty1=com2>

LP <lp=lpt1><lp1=lpt2>

SPECIAL<vdi=vdi>

MISC<gpj=gpj>

13. If your system configuration is **85**, the Unix device configuration shown should be:

PARAMETER[tz=xSTnxDT][i=x][s=?]
[a=128]

SYSTEM<clock=clock><dos=dos><console=console>

DISK<dsk/0=c:><flpa=a:><flpb=b:>

TTY<tty0=com1><tty1=com2>

LP <lp=lpt1><lp1=lpt2>

SPECIAL<vdi=vdi>

MISC<gpj=gpj>

14. Answer "y" if the configuration is correct; otherwise, correct it as directed. If you modify the configuration, the installation program reboots the co-processor and restarts the configuration program back at the point where it asks you for the machine type: PC/AT. YOU MUST MODIFY C:\VALID\UNIX.CFG TO REFLECT THESE CHANGES (they are written to C:\VALID\OPUS.CFG); see "Completion Instructions" at the end of this section.

15. Logical drive information for disk drive 0 is displayed. Answer "y" to confirm or else correct the information if necessary.

16. Answer "y" to the question: "Access partition in single section?". Disk partitioning information is displayed. Answer "y" to confirm.

17. File system information is printed for drive dsk/0. The numbers displayed are the defaults and should appear as below (these numbers are approximate only):

System Configuration	Unix Root	Swap
20/85	128000	4000
85	128000	4000
85/85	128000	4000

18. About 1% of the partition will be reserved for a bad track table.

19. The available disk space will be 0. This number shows the number of blocks not used for Unix root, swap, or spares. This number is the "remainder" disk space.

20. Change the file system assignments to make the swap space 14000 blocks. When you change one of the assignments, the available disk space changes. When the swap is increased, the available disk space will become negative. To correct this, change the root area by subtracting the same number of blocks from the root area as are listed in the available disk space. This will reduce the root area and make the available disk space 0.

21. When complete, the file system partitions should appear as follows (numbers are approximate):

Configuration 20/85 or 85/85:

(1)	Root file system	125000 blocks (62.5 Mbytes)
(2)	Swap area	14000 blocks (7.0 Mbytes)
(3)	Spare block area	1400 blocks (0.7 Mbytes)
	Available	0 blocks (0.0 Mbytes)

Configuration 85:

(1)	Root file system	85000 blocks (42.5 Mbytes)
(2)	Swap area	14000 blocks (7.0 Mbytes)
(3)	Spare block area	1200 blocks (0.0 Mbytes)
	Available	0 blocks (0.0 Mbytes)

22. When satisfied with the file system assignments, type "y" to confirm.

23. Type ENTER to start the bad block mapping process for dsk/0.

24. This process will take about 45 minutes. As the process continues, the percentage of checked cylinders is displayed (for example, 11%). Occasionally, the system will print a series of numbers. These are the block numbers that have been found to be bad and are being spared out.

25. If there are more bad blocks than there are spares, **vinit** will tell you the required number of spare blocks.

    a. Write down the required number of blocks, rounded up to the nearest 50 (e.g. 1504 -> 1550).

    b. Press ENTER.

    c. Change the file system assignments to make the spare space the number that you wrote down above.

d. Reduce the root area by the amount indicated in the "Available" area.

e. Confirm the new file system assignments by typing "y" to confirm.

f. Press ENTER to restart the bad block mapping process for dsk/1.

26. If your configuration is 85/85, you will be prompted to continue with the dsk/1 disk. Otherwise, continue to the next section, "Installing UNIX and GED on the Co-Processor."

27. Answer "y" when asked if the disk is to be accessed in a single partition.

28. The block assignments are shown for Unix file system and spares only (no swap). No changes are necessary. Answer "y" to confirm.

29. WRITE DOWN AND SAVE THE VALUE FOR THE NUMBER OF BLOCKS ALLOCATED TO THE UNIX FILE SYSTEM. THIS WILL BE NEEDED IN A LATER STEP IN THE INSTALLATION. The number should be about 100000.

30. Type ENTER to start the bad track mapping process. This process will take about 45 minutes. As the process continues, the percentage of checked cylinders is displayed (for example, 11%). Occasionally, the system will print a series of numbers. These are the block numbers that have been found to be bad and are being spared out.

31. If there are more bad blocks than there are spares, **vinit** will tell you the required number of spare blocks.

a. Write down the required number of blocks, rounded up to the nearest 50 (e.g. 1504 -> 1550).

b. Press ENTER.

c. Change the file system assignments to make the spare space the number that you wrote down above.

d. Reduce the root area by the amount indicated in the "Available" area.

e. Confirm the new file system assignments by typing "y" to confirm.

f. Type ENTER to restart the bad block mapping process for dsk/0.

32. Once this process is complete, the Unix configuration is complete. You can now install Unix on the co-processor.

6.4 INSTALLING UNIX AND GED

This section describes how to perform the initial installation of Unix and GED on the co-processor and Unix file systems. If you did not enter this step from the previous section, "Configuration of Unix," then type

vinit -R

to start the reinstallation at the proper place.

Parts Needed

1. ROOT diskette.
2. VROOT diskette.
3. UNIX diskettes (labeled UNIX #1 through UNIX #5).
4. GED diskette.

INSTALL MINIMAL UNIX SYSTEM

1. The installation program prompts you to insert the diskette labeled "ROOT" into the floppy drive and hit ENTER. Do so.

2. A minimal root system is loaded into the co-processor. This will be just enough to boot the system and continue to install the rest of the software.

3. The Unix system boots from the swap area.

4. Part of this process involves checking the file systems. The **fsck** run on dsk/0 should be OK. The **fsck** on dsk/1 may fail. This is because no Unix file system has been created on the disk. If it asks whether to continue, answer NO. This will be taken care of later.

5. When the system prompts for the VROOT diskette, remove the ROOT diskette from the floppy drive, insert the VROOT diskette, and press ENTER.

6. It will take several minutes to load VROOT and do a file system check. When this is complete, the display shows the following prompt:

```
INIT: SINGLE USER MODE
#
```

MAKE /u1 FILE SYSTEM (85/85)

If your configuration is 85/85, you must create a file system for dsk/1. Perform the following:

1. Type "mkdir /u1". Ignore the message "mkdir: cannot make directory /u1" **if it appears.**

2. **Type "mknewfs** *size*" where *size* is the size of the file system on the dsk/1 disk in blocks. This number was copied down during the previous step ("Configuration of Unix"). If you don't know the

size of the file system, use **100000**. This command will make the new file system and load it as the /u1 file system.

3. The /u1 file system on the C disk (dsk/1) has now been created.

INSTALL FULL UNIX SYSTEM (All disk configurations)

1. Type the command "vinstall". To proceed with the installation of Unix, answer "y" to the prompt "Do you want to install Unix?".

2. **Vinstall** now verifies the current installation. When asked whether you wish to proceed, type "y". The **vinstall** procedure will prompt for each of the Unix installation diskettes in order. Load each as requested. The diskettes are labelled Unix #1, Unix #2, etc. There are 5 of them and all five must be installed.

3. **Vinstall** will ask for the host name. Type "valid". Next you will be asked for the time zone. Refer to Section 6.3, "Configuration of UNIX" for more information on setting the time zone.

4. After the Unix diskettes are installed, **vinstall** will request that you install the standard library diskette. Load it into the floppy drive.

5. You will now be prompted for the GED security password. This is available on the proof-of-purchase form for this particular system.

6. **Vinstall** will request that you install the GED diskette. Load it into the floppy drive.

7. Once GED has been installed, **vinstall** will announce "You have completed installation of all requested packages."

8. Be sure to type "exit" to reset. You should then see the prompt "INIT: SINGLE USER MODE" followed by "#".

REMOVE REFERENCE TO /u1 (20/85, 85)

If your system configuration is 85/85, skip to the next step. If your configuration is 85 or 20/85, you should do the following:

1. Edit the file */etc/rc* (by typing "vi /etc/rc"), and delete the line:

   /etc/mount /dev/dsk/1s0 /u1

2. To do this, place the cursor anywhere on the line and type "dd".

3. Once the line has been deleted, write the file and exit **vi** by typing ":wq" and pressing ENTER.

4. Type "sync".

5. Basic system installation is now complete.

6. Return to DOS by typing "dos".

7. Reboot Unix by typing "reboot".

6.5 INSTALLING OTHER SCALD SOFTWARE

This section describes how to install the SCALD programs. The user must first have installed Unix and GED (see the previous section). This section does not describe how to install the Ethernet software. See the next section for Ethernet installation.

Parts Needed

1. Diskettes for the additional SCALD programs and libraries to be installed.

INSTALLATION INSTRUCTIONS

1. For each additional software package to be installed, type "vinstall" while in Unix and then insert the first diskette of the package to be installed.

2. The vinstall program will check to see if previous versions of the package exist on the system and then will ask for confirmation to proceed with the installation. You should answer "y".

3. The installation of each software package is verified and a record of the installation is recorded on the machine.

4. When all of the packages have been installed, the Logic Designer AT is ready to use.

6.6 INSTALLING ETHERNET SOFTWARE (Optional)

This section describes the installation of the Ethernet software. This is an optional package and should not be installed if not ordered. Do not install the software if you do not have the board installed. Do not install the software if you will not install the DOS side. Only install the software if you are certain you want to have Ethernet installed.

Parts Needed

1. Ethernet package installation diskette.

PREREQUISITES

1. Unix is installed.

2. The Ethernet board is installed in the Logic Designer AT.

3. The systems that your Logic Designer AT will communicate with conform to the ARPANET/DOD TCP/IP Ethernet standard. To communicate with an S-32 host, it must have Operating System release 7.27 or higher.

INSTALLATION INSTRUCTIONS

Install UNIX and DOS Ethernet Software

1. Type "unix".

2. Type "vinstall". Be sure that the diskette containing the Ethernet package is in the floppy drive.

3. The Unix Ethernet software is installed and recorded.

4. When complete, remove the Ethernet diskette from the floppy drive.

Create the Hosts File

1. Contact the network administrator to determine the network internet address for this machine. It will be a number of the form NN.NN.NN.NN.

2. Determine the name of this machine by asking the network administrator or the owner of the machine. This is the host name.

3. Use **vi** to edit */etc/hosts*.

4. The first line of the file */etc/hosts* is a comment line beginning with "#". The second line in the file is:

   NN.NN.NN.NN *hostname* localhost

   where NN.NN.NN.NN is the network address, *hostname* is the name of the machine, and localhost should be typed in exactly as shown: "localhost".

5. The third line of the file should be typed exactly as:

   127.00.00.00 loopback lb

6. Other lines of the file can contain references to other machines in the form:

   NN.NN.NN.NN *machinename alias*

   where *machinename* is the name of some machine on the net, NN.NN.NN.NN is the network address of the machine, and *alias* is an alias for the machine name. If no alias is desired, do not enter one. For example, if the remote machine is named "other" at the address 99.00.00.01 and has the alias "machine," the entry would appear as:

   99.00.00.01 other machine

7. NOTE: The file must NOT contain any tabs. Also, the machine name and alias are case sensitive.

8. A complete */etc/hosts* file for the machine *valid* might appear as follows:

   99.00.00.99 valid localhost
   127.00.00.00 loopback lb
   99.00.00.01 other machine

9. Write out the file when complete. (Use the command ":wq".)

10. Copy the hosts file by typing:

    udcp -a /etc/hosts C:/xln/tcp/hosts

11. Edit the hosts equivalence file with **vi**:

    vi /etc/hosts.equiv

12. There should be an entry for each host in */etc/hosts.equiv* for each host that you will be communicating with in the following format:

    machinename

    e.g., if your machine is named "valid." the file might look like this:

    valid
    other
    another
    loopback

    Note that you should have your machine name and "loopback" in the file. The order in which the names are listed is not significant for */etc/hosts.equiv.*

Setup for Remote Logins on Local Machine

1. If you wish to let people log in to your Logic Designer AT from another machine (including yourself from another machine), type:

```
cat /etc/inittab.net >> /etc/inittab
```

2. For more information about how to set up the system for Ethernet usage, see the section "Network Database" in the Ethernet Administration appendix to the *Valid Guide to Operations.*

Setup for Remote Who

Use this section if you want to run the **rwho** (remote who) daemon. (See "Rwho - the Remote Who Utility" in the *Valid Guide to Operations* for more information.) Note that there is a substantial CPU time penalty for running the daemon. Valid does not recommend running the **rwho** daemon.

1. Edit the file */etc/rc.network* with **vi.**

2. Remove the number signs ("#") before the lines with "rwhod" in them:

```
#echo  rwhod
#/net/rwhod
```

Remove the number signs by placing the cursor on the "#" and pressing "x" (lower-case) once. This deletes the character. The */etc/rc.network* file should look something like this when you are done:

```
#/bin/sh
# Valid Ethernet TCP/IP setup program
/valid/bin/dos ...
/net/netload -p 3
echo Starting Ethernet Daemons:
echo  rshd
/net/rshd
   :
   :
echo  rwhod
/net/rwhod
```

3. Move the **rwho** local name file to reflect your machine name:

```
# mv /usr/spool/rwho/rwhod.valid
     /usr/spool/rwho/rwhod.localname
```

where *localname* is the machine name of your Logic Designer AT. (The "mv" command is typed all on one line, but for purposes of display in this manual, it is shown on two lines.)

4. Type "sync".

Check DOS Configuration

1. Return to DOS by typing "dos".

2. Ensure that the configuration file, *C:\VALID\UNIX.CFG*, has the correct entry for the Ethernet by entering this command:

```
C:\VALID> find/c "ether" \valid\unix.cfg
```

If the response is:

```
---------- \valid\unix.cfg: 1
```

then the DOS configuration is all right. Continue to the next section, "Test the Ethernet Installation." Otherwise, edit the file *unix.cfg* with **edlin**. In the following example, the portions you type appear in **bold**, and machine responses are in ordinary type.

```
C:\VALID> edlin unix.cfg
End of input file
```

Now type the following to add a line to the file. This line contains the DOS Ethernet configuration data. The commands you enter are shown in bold type.

```
*#
*-1i
```

Add the line (nn) as shown:

nn:*<**ether=ether(5,0xbc00,0x310)**>

Press Ctrl-Z and then press ENTER to end the line addition:

nn+ 1:***^Z** <Ctrl-Z><ENTER>

Type "e" and press ENTER to exit **edlin** and return to DOS:

```
*e
C:\VALID>
```

3. For more information on **edlin**, see the chapter "Line Editor" in the *DOS 3.10 Reference Manual.*

Test the Ethernet Installation

1. Return to Unix. Type "unix".
2. Type "mv /etc/rc.network /etc/rc.test".
3. Type "/etc/rc.test".
4. Ethernet should give some startup messages, including the Ethernet and Internet address. (The Internet address is the number that you assigned when you edited the */etc/hosts* file.) If the messages are not displayed, detect and correct any hardware or software problem. If **rc.test** still doesn't work, call Valid and DO NOT DO THE NEXT STEP.
5. If the Ethernet was successfully installed, type:

   mv /etc/rc.test /etc/rc.network

6. Type "sync".
7. Return to DOS by typing "dos".

Installation Completion

1. Cycle power on the PC/AT.

2. Reboot Unix by typing "unix".

3. When Unix reboots, it will start up Ethernet. You should see the Ethernet initialization messages. You should also see your Ethernet address and Internet Address. If this is not the case, there may be a problem with your host file, Ethernet cable, or Ethernet card. See the *Valid Guide to Operations* for more information.

4. Ethernet is now installed.

6.7 PREPARING THE SYSTEM FOR USE

This section describes how to prepare the system for use, completing the installation.

1. If you are in Unix, type "sync" twice.

2. Type "dos". This switches operation to PC-DOS.

3. If you modified any configuration parameters with **vinit**, type

   copy \VALID\OPUS.CFG \VALID\UNIX.CFG

4. Type "reboot" to restart Unix.

5. The system should boot and display a login prompt.

6. The system is now ready for use.

APPENDIX A
SYSTEM MESSAGES

This chapter describes system messages and error codes in **opmon** messages, including Level 1 and Level 2 selftest messages; **opsash** messages; and other UNIX messages.

This chapter has the following sections:

- An alphabetized list of messages, their source, and meaning, excluding I/O messages.
- I/O messages grouped according to device.
- Instructions on how to turn on **opmon** error reporting.
- An explanation of the coprocessor hardware status bits.

See the man pages on **opmon**(1) and **opsash**(1) for more information.

A.1 MESSAGES

?

Response to <alt-128>. Type RETURN for menu, <space> to cancel. See **opmon**(1V) for option explanations.

ABT TRAP ENCOUNTERED <STATUS=xx>

Any standalone program can produce a TRAP error. TRAP errors are always fatal. An ABT trap means an unexpected MMU abort occurred. The status bits report the coprocessor hardware status. See "Status Bits" below.

bad blk on DK drive <dn>, section <sn>
bn = <bn> er = <er1>,<er2>
The file system residing on the indicated disk device has an inconsistency in its free block list, if the block number is out of the range of the file system.

<dn> is the UNIX drive number (the minor device number divided by 8).

<sn> is the partition number (minor device number modulo 8).

<bn> is the disk block number relative to the beginning of the section specified by <sn> on the drive specified by <dn>.

<er1> is the **er** value returned by **opmon.**

<er2> is the **ds** value returned by **opmon.**

bad count on DK drive <dn>, section <sn>
bn = <bn> er = <er1>,<er2>
The file system residing on the device is inconsistent with information maintained in memory. (See "bad blk on DK" above.)

Bad free count on DK drive <dn> section <sn>
bn = <bn> er = <er1>,<er2>
The file system residing on the disk device has an inconsistent free block list. See "bad blk on DK".

BPT TRAP ENCOUNTERED <STATUS=xx>
Any standalone program can produce a TRAP error. TRAP errors are always fatal. A BPT trap means the Breakpoint instruction was executed. The status bits report the hardware status.

Configuration: 32.16 2MB console dos dsk/0 dsk/1 flpa flpb lp tty0 tty1
An inventory of your system, including system type, amount of memory, and devices, printed

during boot by **opsash**. This list is derived from the DOS file *UNIX.CFG*, and so it will differ from system to system. (See the man pages on files for more information on *UNIX.CFG*.)

Device error on <device name, device number>
bn = <bn> er = <er1>,<er2>
A device has encountered a physical problem in accessing a block.

If **opmon** error reporting is turned on, the UNIX kernel usually reports **er** and **ds** messages in addition to the messages generated by **opmon**. Within the kernel message, <er1> is **er** and <er2> is **ds**. Both numbers are in hex.

UNIX does not generate messages when an error is corrected after retry (er1=1) or corrected after error correction (er1=2). When messages are generated both by **opmon** and UNIX, the **opmon**-generated errors are output first, followed by a blank line, and then the UNIX message.

Device error on DOS device <dn>
bn = <bn> er = <er1>,<er2>
The DOS driver has encountered a problem while communicating with **opmon**. **<dn>** is the minor device number for the DOS interface (usually **0**). **<bn>** is **0**. Note that errors in the execution of DOS calls are passed back for use by **opdos**(1), and do not cause the kernel to print error reports.

dma error: eia = <xxx>, pc = <yyy>
panic: memory
A parity error has been detected during access of coprocessor memory via the PC bus. *<xxx>* is the address of the data access and *<yyy>* is the UNIX program counter location at the time of the error.

DVZ TRAP ENCOUNTERED <STATUS=xx>

Any standalone program can produce a TRAP error. TRAP errors are always fatal. A DVZ trap means an attempt was made to divide by zero. The status bits report the hardware status.

FLG TRAP ENCOUNTERED <STATUS=xx>

Any standalone program can produce a TRAP error. TRAP errors are always fatal. A FLG trap means an illegal flag was encountered. The status bits report the coprocessor hardware status; see "Status Bits" at the end of this appendix.

FPU TRAP ENCOUNTERED <STATUS=xx>

Any standalone program can produce a TRAP error. TRAP errors are always fatal. A FPU trap means a floating point exception was encountered. The status bits report the coprocessor hardware status; see "Status Bits" at the end of this appendix.

ILL TRAP ENCOUNTERED <STATUS=xx>

Any standalone program can produce a TRAP error. TRAP errors are always fatal. An ILL means an illegal operation was encountered. The status bits report the coprocessor hardware status; see "Status Bits" at the end of this appendix.

Insert diskette for drive A: and strike any key when ready.
Insert diskette for drive B: and strike any key when ready.

When two diskette drives are simulated on a single physical drive, these messages prompt the operator to insert the appropriate diskette when a change is required.

**** I/O req: dix=xx cmd=xx st=xx er=xx**
ds=xxxx ad=xxxx cnt=xxxx blk=xxxx
**** I/O done: dix=xx cmd=xx st=xx er=xx**
ds=xxxx ad=xxxx cnt=xxxx blk=xxxx
**** I/O err: dix=xx cmd=xx st=xx er=xx**
ds=xxxx ad=xxxx cnt=xxxx blk=xxxx

I/O request and I/O done messages are output for specified device index numbers (dix(es)) by trace logic, if enabled. I/O error message is ouput by error-reporting logic, enabled by <alt-128>. See the section "I/O Error Messages" below for more information.

Level 1 Test: PASS

Level 1 test passed.

Level 1 Test: xx

Level 1 test running; xx is test phase. The phases are:

- 0x - status tests
  - 00 - initial (reset) status
  - 01 - interrupt (PC to coprocessor) status
  - 02 - stuck bits
  - 03 - run mode off
  - 04 - run mode on
  - 05 - touch memory while reset
- 1x - test coprocessor memory byte 0
  - 10 - pattern 00
  - 11 - pattern 55
  - 12 - pattern AA
  - 13 - pattern 01
- 14-43 - test bytes 1-2-4-8...8000 as above
- 5x - test coprocessor memory (constant data)
  - 50 - pattern 0000
  - 51 - pattern FFFF
  - 52 - pattern 5555
  - 53 - pattern AAAA
  - 54 - pattern 0001 (low byte parity)
  - 55 - pattern 0100 (high byte parity)
- 6x - test coprocessor memory addressing
  - 60 - pattern = address
  - 61 - pattern = inverted address
- 9x - test status while executing
  - 90 - initial status
  - 91 - status while running
- FF - reset coprocessor and zero memory

Level 1 Test: xx FAIL...

Level 1 test phase xx failed.

Level 1 Test: xx FAIL: DMA abort detected.
An MMU abort of a DMA cycle was detected during level 1 test.

Level 1 Test: xx FAIL: memory location nnnn; wrote nn, read nn, then nn.
Level 1 memory test failed; the results of two successive reads are reported.

Level 1 Test: xx FAIL: parity error detected.
A memory parity error was detected during level 1 test.

Level 1 Test: xx FAIL: status bits nn always on; bits nn sometimes on.
Level 1 status test failed for the reasons indicated. The status bits mentioned here are PC status bits.

Level 1 Test: xx FAIL: status should be nn but was nn, then nn.
Level one status test failed; the results of two successive status reads are reported. The status bits mentioned here are PC status bits.

Level 2 Test: FAIL
Level 2 test terminated prematurely.

Level 2 Test: PASS
Level 2 test passed.

Level 2 Test: TIMEOUT
Level 2 test timed out.

Level 2 Test: xx
Level 2 test is running; xx is timer.

MEMORY TEST: DMA: nnnnnn ERRORS
The memory test found a DMA error; nnnnnn is the total number of DMA errors in hex.

MEMORY TEST: nKb PASSED nKb FAILED nnnnnn ERRORS

```
BANK (1 MB)  -->        #0        #1        #2        #3
                     +++++++   nnnnnnnn*  .......   ........
```

ADDR=nnnn BANK n WRITTEN=nnnnnnnn
READ=nnnnnnnn (BIT nn)

or

ADDR=nnnn (BANK n) WRITTEN=nnnnnnnn
READ1=nnnnnnnn, READ2=nnnnnnnn

...

MEMORY TEST: PARITY: n ERRORS

A memory error was encountered during a Level 2 Test. These messages are reported by **opsash**, which runs the Level 2 Test. "nKb" is a multiple of 128 Kbytes and represents the percentage of memory that passed or failed the memory test. "ERRORS" is the total accumulated number of errors. This number is in hex and can be very large even if only one chip is bad.

The next line displays a "map" of memory, showing which banks contain bad bits or parity errors. The numbers 0, 1, 2, etc. are 128-Kbyte bank numbers. Interpret the display as follows:

```
++++    If the bank is without errors
nnnn    If the bank contains bad bits;
          the bits on in the hex number
          ''nnnn'' indicate the bad bits
*       If the bank contains a parity error
```

Then, for each of the first 8 memory errors, the test prints an **ADDR=** line such as one of the above. ADDR is the memory address (in hex) where the error occurred. BANK is the memory bank; each memory bank contains 128 Kbytes. Banks are numbered in hex beginning with 0, so bank 0 is 0-127 Kbytes, bank 1 is 128-256 Kbytes, etc. WRITTEN is the hex

value written to the address, and READ is the value read back. An error is noted when the values for WRITTEN and READ differ. Two reads are done; an error is also noted if the results of the two reads do not agree.

The first form of the message listed above (with the BIT field) occurs when there is only one single-bit error, and the test can pinpoint it. The second form appears when the results of the first read differ from the results of the second; then the results of both reads are printed.

The rest of the errors, after the first 8, are neither printed nor saved.

MEMORY TEST: PARITY: nnnnnn ERRORS

The memory test found a parity error; nnnnnn is the total number of parity errors in hex.

no space on DK drive <dn>, section <sn>
bn = <bn> er = <er1>,<er2>

The file system residing on the disk device has no free blocks left. You should delete files to make more space. See "bad blk on DK".

Out of inodes on DK drive <dn>, section <sn>
bn = <bn> er = <er1>,<er2>

The file system residing on the disk device has no free inodes. Delete files to make more inodes. See "bad blk on DK".

opsash: <filename>: not found

Opsash could not find the bootfile. Check pathnames and spelling.

opsash: <filename>: could not load

Opsash found the bootfile but could not load it. Check for permissions, file corruption.

*****Opus5: can't resume**

Can't resume after pause (coprocessor not

running).

*****Opus5: dma abort; iorb=xxxx**
DMA access to coprocessor aborted by MMU; xxxx specifies I/O request block in process during failure (0000 indicates no iorb in process).

*****Opus5: error in configuration file: <file name>**
The configuration file (default *UNIX.CFG*) contains an error. Check syntax, spelling.

*****Opus5: error loading boot file: <file name>**
DOS error while opening/reading boot program (usually opsash or opdiag).

*****Opus5: illegal option (use /L /C /P)**
opmon command line option was unrecognized.

*****Opus5: parity error; iorb=xxxx**
Parity error during DMA access to coprocessor; xxxx specifies I/O request block in process during failure (0000 indicates no iorb in process).

*****Opus5: stack underflow *****
The **opmon** stack underflowed; fatal error.

*****Opus5: unknown device: <device name>**
A device name in the configuration file was not recognized.

parity error: pc = <xxx>
panic: memory
A parity error has been detected during access to coprocessor memory via the internal bus. <*xxx*> is the UNIX program counter location at the time of the error.

SVC TRAP ENCOUNTERED <STATUS=xx>
Any standalone program can produce a TRAP error. TRAP errors are always fatal. An SVC trap means a supervisor call was encountered. The status bits tell the state of the coprocessor hardware; see "Status Bits" below.

TRC TRAP ENCOUNTERED <STATUS=xx>

Any standalone program can produce a TRAP error. TRAP errors are always fatal. A TRC trap means "trace on" call was encountered. The status bits tell the state of the coprocessor hardware; see "Status Bits" below.

UND TRAP ENCOUNTERED <STATUS=xx>

Any standalone program can produce a TRAP error. TRAP errors are always fatal. A UND trap means an undefined opcode was encountered. The status bits tell the state of the coprocessor hardware; see "Status Bits" below.

A.2 I/O ERROR MESSAGES

This section describes I/O error messages generated when **opmon** error reporting is turned on. The section has three parts:

- Error Message Format
- How to Interpret the Dix (Device Index) Field
- Device Error Messages (er=0x09) Grouped by Device

ERROR MESSAGE FORMAT

Opmon gives device status with two codes, called **er** and **ds**. **er** is the generic error code, and **ds** is device-dependent status. **ds** is not always used, and when not used is 0.

If **opmon** error reporting is turned on, the contents of any I/O request block (iorb) with **er** not 0 are printed just before the iorb is marked as done. The format of the message is as follows:

**** I/O err: dix=xx cmd=xx st=xx er=xx**
ds=xxxx ad=xxxx cnt=xxxx blk=xxxx

dix = device index in hex. Assignments are as follows:

1 = first <device> in **opmon** configuration file (*UNIX.CFG*)
2 = next <device>
3 ...

The console and asynchronous devices (<com1>, <com2>, etc.) each take three **dix** entries: one each for input, output, and control.

cmd = I/O command
0 = nop
1 = reset
2 = status request
3 = read
4 = write
5 = read; no wait
6 = I/O control

st = iorb status
0 = idle
1 = go
2 = busy
4 = done

er = error type (hex)
00: e_ok no error
01: e_rt ok after retry
02: e_ce ok after correction
03: e_cmd illegal command
04: e_da illegal device address (e.g., disk block number)
05: e_ma illegal memory address (outside 0000-EFFF)
06: e_na device not available
07: e_ro attempt to write read-only device
08: e_de data error
09: e_dev device error (details in device status word)
0A: e_dix illegal device index
0B: e_can canceled by system request
0C: e_ni no input available
0D: e_dma DMA abort (MMU abort during DMA)
0E: e_par parity error during DMA
0F: e_crit DOS 'critical error' (int 24h) -- (not currently used)
10: e_to device timeout

11: e_sync IORB request with IORB status <> GO
12: e_badb bad block on disk (flagged block)

ds = device-dependent status
This field varies according to device; see below.

ad = memory address within coprocessor segment
Legal values are 0000-EFFF.

cnt = count (usually byte count) in hex

blk = block number (e.g., disk block number) in hex

HOW TO INTERPRET THE DIX

I/O messages can be reported by devices of the kinds listed below. The device names shown (such as <fil0> or <a:>) are in *UNIX.CFG* format. When a message is reported, the **dix** (device index) is given; this number maps to the position of the device in the file *UNIX.CFG*.

For example, if the *UNIX.CFG* looks like the sample below, the **dix**es are assigned as listed.

```
[s=?]
[i=7]
[tz=PST8PDT]
<clock=clock>
<console=console>
<dos=dos>
<dsk/0=c:>
<flpa=a:>
<flpb=b:>
<lp=lpt1>
<tty0=com1>
<tty1=com2>
<end>
```

Within this sample file, the **dix** (device index) numbers would be the following:

<clock> 1

<console>	2, 3, 4 (input, output, and control)
<dos>	5
<c:>	6
<a:>	7
<b:>	8
<lpt1>	9
<com1>	A, B, C (input, output, and control)
<com2>	D, E, F (input, output, and control)

DEVICE ERROR (er=0x09) MESSAGES GROUPED BY DEVICE

This section lists messages generated for each kind of device listed in *UNIX.CFG.*

- Asynchronous I/O
- IBM PC Hard Disk (BIOS)
- IBM PC Diskette (BIOS Driver)
- IBM PC Diskette (DOS Driver)
- IBM PC Line Printer (Parallel Port)
- DOS Driver
- DOS-Based Virtual Drive Files

Asynchronous_I/O

UNIX driver: AS
opmon devices: <com#>

These are messages from serial lines; serial lines are named devices <com1>, <com2>, etc., in *UNIX.CFG.* The messages are coded in bits and can be OR'd. They are output in hex.

0001 - break detected
0002 - parity error detected
0004 - framing error detected

0008 - input overrun (hardware)
0010 - input overrun (software buffer)

0080 - change in modem status detected

0100 - DTR active (out) - data terminal ready
0200 - RTS active (out) - request to send

1000 - DSR active (in) - data set ready
2000 - CTS active (in) - clear to send
4000 - CD active (in) - carrier detect
8000 - RI active (in) - ring indicator

Example:

```
dix=0D cmd=03 st=02 er=09 ds=0380
       ad=0000 cnt=0000 blk=0000000D
```

This means that on the device having the device index number 0D (<com1> in our example), when a read (03) was attempted, with iorb status busy (02), a device error (09) occurred of type 0380 (DTR active, RTS active, change in modem status detected).

IBM_PC_Hard_Disk_(BIOS)

UNIX driver: DK
opmon devices: <c:> ...

The codes issued by the IBM PC BIOS (basic input/output system) disk driver are listed here. The BIOS is code that resides in ROM on the PC system board or on disk controllers. It provides a standard interface to standard PC hard disks or any other compatible hard disk and controller; this interface allows programs to access the disk without having to be concerned with its device address or operating characteristics.

This device is used only if you use partition-based UNIX virtual drives.

VALID codes (hex):

DE - bad ecc on disk read - not correctable
CE - bad ecc on disk read - corrected
BD - bad block

AT only (hex):

E0 - status error/error reg=0
CC - write fault on selected drive
BB - undefined error
AA - drive not ready
80 - attachment failed to respond (timeout)
40 - seek operation failed
20 - controller failed
09 - data crosses 64K boundary
07 - drive parameter activity failed
05 - reset failed
04 - requested block not found
02 - address mark not found
01 - bad command passed to disk I/O

Example:

```
dix=20 cmd=03 st=02 er=09 ds=0004
       ad=0840 cnt=0200 blk=0000000D
```

This means that on the device having the device index number 20 (not listed in our example because not part of standard *UNIX.CFG*), when a read (03) was attempted, with iorb status busy (02), a device error (09) occurred of type 0004 (requested block not found).

IBM_PC_Diskette_(BIOS_driver)

UNIX driver: DK
opmon devices: <a:bios> <b:bios>

The codes issued by the IBM PC BIOS diskette driver are listed here.

VALID codes (hex):

DE - bad crc on disk read

AT (hex):

80 - attachment failed to respond (timeout)
40 - seek operation failed
20 - controller failed
09 - data crosses 64K boundary
08 - DMA overrun
07 - drive parameter activity failed
04 - requested block not found
03 - write attempted on write-protected diskette
02 - address mark not found
01 - bad command passed to disk I/O

Example:

```
dix=0A cmd=04 st=02 er=09 ds=0080
       ad=0D44 cnt=0014 blk=00000000
```

This means that on the device having the device index number 0A (<a:bios> in our example), when a write (04) was attempted, with iorb status busy (02), a device error (09) occurred of type 0080 (attachment failed to respond). This can happen if no floppy is loaded in the floppy drive.

IBM_PC_Diskette_(DOS_driver)

UNIX driver: DK
opmon devices: <a:dos> <b:dos>

Valid provides a driver that talks to the PC floppy disk controller, via DOS interrupts 25h and 26h, instead of via the BIOS.

Valid codes (hex):

DE - bad crc on disk read

er = e_dev (0x09):

hex 80 - attachment failed to respond (timeout)
40 - seek operation failed
20 - controller failed
10 - bad CRC on read
08 - DMA overrun
04 - requested block not found
03 - write attempted on write-protected diskette
02 - address mark not found
00 - other

Example messages for the <a:dos> device would be similar to those for the BIOS floppy disk device, except that the *UNIX.CFG* file would have to contain an entry for <a:dos>.

IBM_PC_Line_Printer_(parallel_port)

UNIX driver: LP
opmon devices: <lpt1> ...

The driver for the PC line printer (listed as <lpt1> in the sample *UNIX.CFG*) sends the following messages (in hex):

80 - busy
20 - paper out
10 - printer not selected (offline)
08 - I/O error

Example:

```
dix=0C cmd=04 st=02 er=09 ds=0020
       ad=0D44 cnt=0014 blk=00000000
```

This means that on the device having the device index number 0C (<lpt1> in our example), when a write (04) was attempted, with iorb status busy (02), a device error (09) occurred of type 0020 (paper out).

DOS_Driver

UNIX driver: DOS
opmon device: <dos>

The DOS driver enables UNIX to make DOS system calls. It is listed as <dos> in *UNIX.CFG.*

The error messages for <dos> are the same as the messages for the DOS-based virtual drive files (see next section).

The **cmd** field in DOS I/O messages is not meaningful (always 00) for the <dos> device.

DOS-based_Virtual_Drive_Files

UNIX driver: DK
opmon devices: <fil#> <a:> <b:>

UNIX usually resides in DOS files that are seen as physical disk drives by the kernel. These DOS-file-based virtual drives can reside on hard disk (<fil0>, <fil1>, etc.), or on floppy diskettes (<a:>, <b:>).

The messages for <dos> are the same as the messages for DOS-file-based virtual drives.

1 - invalid function number
2 - file not found
3 - path not found
4 - too many opened files
5 - access denied
6 - invalid handle
7 - memory control blocks destroyed
8 - insufficient memory
9 - invalid memory address
A - invalid environment
B - invalid format
C - invalid access code
D - invalid data
F - invalid drive

10 - attempt to remove current directory
11 - not same device
12 - no more files
13 - attempt to write on write-protected diskette
14 - unknown unit
15 - drive not ready
16 - unknown command
17 - data error (CRC)
18 - bad request structure length
19 - seek error
1A - unknown media type
1B - block not found
1C - printer out of paper
1D - write fault
1E - read fault
1F - general failure
20 - sharing violation
21 - lock violation
22 - invalid disk change
23 - FCB unavailable
24-4F - reserved
50 - file exists
51 - reserved
52 - cannot make
53 - fail on INT 24

Example:

```
dix=05 cmd=00 st=02 er=09 ds=0009
          ad=0D44 cnt=0014 blk=00000000
```

This means that on the device having the device index number 05 (<dos> in our example), with iorb status busy (02), a device error (09) occurred, of type 0009 (invalid memory address).

A.3 TURNING ON OPMON ERROR REPORTING

To turn on **opmon** error reporting from UNIX, press <alt-128>. (Hold down the ALT key and type the numbers 1, 2, and 8 *on the numeric keypad.* Then release the ALT key.) This causes a question mark to be printed. Type "e" after

the question mark as shown below.

$ <alt-128>?e

Either "on" or "off" is printed, depending on the current state. To turn off error reporting, do the same operation you used to turn it on. The "e" is a toggle; that is, typing "e" again reverses the effect of the previous "e".

Error reporting can be turned on in *UNIX.CFG* by placing [e] in the file.

A.4 STATUS BITS

Several special physical addresses can be read or written by the coprocessor to monitor and control external events. These addresses are distinguished from memory addresses by the fact that they have bit 23 set to one (800000 hex). Note that these are physical addresses; the full 24-bit virtual address space is still available.

These I/O locations are:

Address	Name	Function
810000	STAT	Status byte:

Bit 7: INT = interrupt from PC
Bit 6: IRQ = interrupt to PC active
Bit 5: PAR = parity error
Bit 4: DMA = DMA abort
Bit 3: OPT = option switch open
Bit 2: EIRQ = IRQ enabled
Bit 1: 0
Bit 0: 0

APPENDIX B
DEVICES

The PC has direct access to all devices on the system. The Valid-supplied program **opmon**(1), running on the PC under DOS, mediates between UNIX and the PC to provide access to devices from UNIX. **Opmon** contains drivers for all devices in the system. The purpose of device drivers under UNIX is to provide information for **opmon**. Only **opmon**'s drivers talk directly to the device controllers.

Opmon knows devices by the names specified in the DOS file *UNIX.CFG*; these filenames are enclosed in angle brackets like this: <UNIX_name=opmon_name>. (See the man pages on files for more information on *UNIX.CFG*.) These device names can be modified by editing \VALID*UNIX.CFG*. The UNIX kernel knows devices by the major and minor device numbers listed in the UNIX directory */dev*. These numbers can be accessed by listing the directory */dev* using the command **ls -l**, as shown below.

```
crw-rw-rw-   1 root   sys  10,  0 Nov 16 16:58 lp
|                          |    |
|__ character device       |    |__ minor device number
                           |__ major device number
```

UNIX operating systems have two kinds of devices: block devices and character devices. Block devices are listed by **ls -l** with "b" as the first character in the line; character devices have "c", as shown above. Block devices do buffered I/O; character devices do not. Some devices, e.g., printers and terminals, are never accessed in block mode. Others, e.g., disks and diskettes, sometimes use block mode and sometimes character mode. Typically, block mode is used only when a device contains a file system. Character mode is sometimes referred to as "raw" mode -- hence the "r" in */dev/rdsk*.

There is only one kernel driver, DK, for all UNIX disk devices. However, there are three **opmon** drivers for hard disks.

1. The <c:xt> driver uses DOS BIOS calls and is optimized for the IBM PC XT controller. This controller can be used with a number of different disks.

2. The <c:> driver uses "generic" DOS BIOS calls. Any BIOS-compatible disk and controller pair can use this driver.

3. The <fil0> driver uses DOS calls. This driver is for disks and controllers that cannot use BIOS calls.

The BIOS is code that resides in ROM on the PC system board or on controllers such as the XT disk controller; it provides a standard interface to the standard PC hard disks, or any other compatible hard disk and controller.

Both the PC XT hard disk driver and the BIOS driver support partition-based logical drives. However, disks that do not use the BIOS interface cannot use partition-based logical drives; they must use DOS file-based logical drives, and this involves some performance loss. Thus, non-BIOS-compatible disks can be used with UNIX, but are not recommended.

This chapter discusses the correspondence between device names in /dev and *UNIX.CFG*. For each major device number in /dev and for each **opmon** driver listed in *UNIX.CFG*, we present the kernel driver name, the kernel device name(s), and the **opmon** driver(s) called by each kernel driver.

The following major device numbers are defined in UNIX:

	Character	**Block**
0	console	dk
1	-	
2	sys	
3	mem	
4	character dk	

5 -
6 -
7 -
8 err
9 as
10 lp
11 prf
12 sxt
13 dos
14 vdi, gpib

See section 7 of the *UNIX System V Administrator Reference Manual* for more information on the devices discussed here.

CONSOLE DRIVER

UNIX driver:	console
UNIX names:	/dev/console, /dev/systty
major device number:	0
opmon drivers:	<console> <doscon>

Two **opmon** console drivers are available: <console> and <doscon>. Both drive the system keyboard and monitor. Only one console entry may appear in *UNIX.CFG*.

<console> uses PC BIOS commands to control the keyboard and monitor. Its *terminfo*(4) definition is TERM=opus-pc.

<doscon> uses DOS commands to control the keyboard and monitor. Its *terminfo*(4) definition is TERM=opus-ansi. If no console device is specified in *UNIX.CFG*, <doscon> is used by default.

DK DRIVER

UNIX driver:	dk
UNIX names:	/dev/dsk/0s0, ... , /dev/rdsk/0s0, ... /dev/flpa, ... , /dev/rflpa, ...
major device number:	0 (block), 4 (character)
opmon drivers:	<fil#> <a:> <b:> <x:##> <x:##.xt> <a:bios> <b:bios> <a:dos> <b:dos>

The dk driver controls two types of devices: block (buffered) disk devices and character (unbuffered) disk devices. Block dk devices are in the directory */dev/dsk*; the files */dev/flpa*, */dev/flpb*, etc., are also block devices. Raw dk devices are in the directory */dev/rdsk*; the files */dev/rflpa*, */dev/rflpb*, etc., are also raw devices.

Each block dk drive corresponds to a disk partition or DOS file and can contain one or more UNIX file systems. The kernel treats these partitions or DOS files as disk drives, capable of being divided into "sections."

Each such virtual drive controlled by the dk driver can contain up to 8 logical "sections", numbered 0 though 7. UNIX file systems can be created on section boundaries with **mkfs**(1M). Section 7 is reserved for use by the system.

The major device number of block dk devices is always 0; the minor device numbers specify the virtual drive and the section within it. There can be up to 16 dk devices, numbered 0-15; numbers 13-14 are ordinarily reserved for **opmon** devices <b:> and <a:>, respectively; number 15 is reserved.

The naming conventions for block dk devices are as follows:

```
/dev/dsk/nnsn
          | |__logical section number within DOS file (0-7)
          |
          |__dk drive number (0-15)
```

For dk devices, the integer result of dividing the minor device number by 8 gives the drive number. For example, a minor device number of 12 for a dk device means drive 1 section 4 and is called */dev/dsk/1s4*.

With the exception of **opmon** devices <a:>, and <b:>, noted below, dk devices are assigned UNIX drive numbers in the order in which they are listed in *UNIX.CFG* unless overridden. For example:

```
<fil0>    /dev/dsk/0s0 - /dev/dsk/0s7
<c:10>    /dev/dsk/1s0 - /dev/dsk/1s7
<fil1>    /dev/dsk/2s0 - /dev/dsk/2s7
  .              .
  .              .
  .              .
```

<fil#> DEVICES

<fil#> devices in *UNIX.CFG* are DOS files seen by the kernel as virtual disk drives. Up to 16 such devices (<fil0>, <fil1>, ... <fil15>) can take the following parameters:

<fil0(max_size,filename,UNIX_drive_number)>

where

max_size — is the maximum size of the DOS file that will contain the UNIX file system(s). The default is "no limit". During initialization, you must tell **opinit** the size of each UNIX file system, so it can execute the command **mkfs**(1M); this keeps UNIX from writing past the end of the file system. Specifying a maximum size here in *UNIX.CFG* is an extra precaution that forces the driver not to write past the end of the DOS file. This

number is specified in 512-byte blocks.

filename — is the DOS filename. By default, <fil0> uses DOS file *\opus\opfs\opfil0*, <fil1> uses *\opus\opfs\opfil1*, etc.

UNIX_drive_number — is the UNIX drive number (0-15 decimal) assigned to this file, overriding the default dependence on *UNIX.CFG* sequence.

<a:> and <b:> are special <fil#> devices. <a:> and <b:> in *UNIX.CFG* are dk devices that use DOS files on diskette drives a: and b:. They correspond to UNIX devices */dev/flpa*, */dev/flpb*, etc., and */dev/rflpa*, */dev/rflpb*, etc. These devices are used during system installation; the normal UNIX tape commands such as **cpio**(1) and **tar**(1) also typically use these devices. These devices contain DOS files which are logical UNIX disk devices. Devices <a:> and <b:> default to:

<a:(,a:\opfil,14)>
<b:(,b:\opfil,13)>

<x:##>

These dk drives are located in disk partitions instead of DOS files. "x:" refers to the DOS drive designation of the drive containing the partition (C: or D: for most PCs), and "##" refers to the partition type field. "##" defaults to 10. Types 10-13 are recognized by UNIX as UNIX partitions.

These devices take three optional arguments, as shown below:

<x:##(drive_number, hw_drive_number, partition_type)>

where

drive_number is the kernel device drive number (0-15), as defined above.

hw_drive_number is the physical unit number. Defaults are:

Drive_#	Unit_#
C	0
D	1
E	2
F	3

partition_type overrides the specification of the partition type. See example below. This is rarely used.

Examples:

<c:>	partition type 10 on drive C:
<c:10>	partition type 10 on drive C:
<c:11(2)>	partition type 11 on drive C:, to be used for /dev/dsk/2s0
<e:10(3, 2, 12)>	partition type 12 on drive D:, to be used for /dev/dsk/3s0 In effect, maps e:10 to d:12.

<#:bios> AND <#:dos>

Two kinds of drives control diskettes directly, rather than through a DOS file. These drives are useful for reading diskettes which are not readable by either DOS or UNIX. These drives are:

- devices <a:bios> and <b:bios>, which do I/O with PC BIOS calls.
- devices <a:dos> and <b:dos>, which do I/O with DOS calls. If these are used, they replace the BIOS devices.

Since these logical drives are rarely used, there are no default UNIX devices for them. If you want to use them,

you must set up the UNIX devices using the **mknod**(1M) command. By default, the a: drive has minor device number 12, and the b: drive has minor device number 11; so your **mknod** commands for the a: diskette would look like the following:

```
mknod block_device_name b 0 96
mknod char_dev_name c 4 96
```

You must then insert the appropriate entry (<a:bios>, <a:dos>, etc.) into *UNIX.CFG* using **EDLIN** so **opmon** will know which driver to use.

SYS DRIVER

UNIX driver:	sys
UNIX device:	/dev/tty
major device number:	2
opmon device:	none

The UNIX device */dev/tty* is associated with major device number 2. It is a virtual device; it is a synonym for the controlling terminal for any given process.

MEM Driver

UNIX driver:	mem
UNIX devices:	/dev/null, /dev/mem, /dev/kmem
major device number:	3 (character)
opmon driver:	none

/dev/mem is a UNIX special file that is an image of the core memory of the computer. See **mem**(7). */dev/kmem* is the same as */dev/mem* except that kernel virtual memory rather than physical memory is accessed. See **mem**(7). */dev/null* is the null file. See **null**(7).

ERR Driver

UNIX driver:	err
UNIX device:	/dev/err
major device number:	8 (character)
opmon driver:	none

Minor device 0 of the **err** driver is the interface between a UNIX process and the system's error-record collection routines. See **err**(7).

AS DRIVER

UNIX driver:	as
UNIX devices:	/dev/tty0, /dev/tty1 ...
major device number:	9 (character)
opmon driver:	<com#> <comh#>

The UNIX TTY driver controls access to devices */dev/tty0* and */dev/tty1*, which are associated with modems or terminals. The TTY driver calls the **opmon** <com#> or <comh#> driver.

<com#> devices are associated with the standard serial port(s) on the PC. There can be one or two <com#> devices. <comh#> devices are associated with additional serial ports provided by the Hostess Multiuse Host Adapter by Control Systems, Inc. There can be up to 8 <comh#> devices, named <comh1> through <comh8>.

The first <com#> or <comh#> device listed in *UNIX.CFG* corresponds to devices */dev/tty0* and */dev/tty128*, the second to */dev/tty1* and */dev/tty129*, and so on.

A *tty#* device created by adding 128 to the minor device number of a *tty#* device does not require Carrier Detect to be present before initiating communication across a serial line. This can be useful if you want the same modem to

serve for both incoming and outgoing calls, or if no modem is attached.

<com#> or <comh#> devices have two arguments for I/O address and interrupt level, respectively.

<com#(io_address, interrupt_level)>
<com#(io_address, interrupt_level)>

These arguments are optional for <com#> devices, but required for <comh#> devices, since Valid cannot determine which I/O addresses might cause a conflict in your system.

LP DRIVER

UNIX driver:	lp
UNIX devices:	/dev/lp, /dev/lp1 ...
major device number:	10 (character)
opmon driver:	<lpt#>

The UNIX **lp** driver controls access to line printer devices. It calls the **opmon** driver <lpt#>.

PRF DRIVER

UNIX driver:	prf
UNIX device:	/dev/prf
major device number:	11 (character)
opmon driver:	none

The file */dev/prf* is a pseudo-device with no associated hardware. It provides access to activity information in the operating system. See **prf**(7).

SXT DRIVER

UNIX driver:	sxt
UNIX devices:	/dev/sxt000, ... , /dev/sxt037
major device number:	12 (character)
opmon driver:	none

The **sxt** pseudo-devices are used by the shell layering facility **shl**(1). See **sxt**(7).

DOS DRIVER

UNIX driver:	dos
UNIX device:	/dev/dos
major device number:	13 (character)
opmon driver:	<dos>

The dos driver provides access to DOS system calls. It is used for the DOS interface from UNIX. See also **opdos**(1).

CLOCK DRIVER

UNIX driver:	none
UNIX device:	none
major device number:	none
opmon driver:	<clock>

Opmon provides UNIX with a real-time clock. Regardless of the PC clock rate, the kernel expects **opmon** to simulate a 60-Hz clock. Therefore, in order to derive the 60-Hz clock rate, **opmon** needs to know the PC real-time clock rate. **Opmon** assumes that PCs clock at 18.2 Hz.

The defaults can be adjusted by specifying the clock frequency in Hz as **n**umerator and **d**enominator of a fraction:

<clock(n,d)>

For example, to change the default to 18.75 Hz, change <clock> in *UNIX.CFG* to read

<clock(1875,100)>

If the clock rate on your PC is different from the defaults, and you do not change *UNIX.CFG*, UNIX will run but keep incorrect time.

There is no device in the UNIX directory */dev* corresponding to the clock.

APPENDIX C
ETHERNET ADMINISTRATION

Network administration of an Ethernet network includes management of the following tasks:

- Administer host names and the corresponding Internet and Ethernet addresses.
- Maintain network databases.
- Disconnect/relocate systems.

It should be mentioned here that it is not necessary that all hosts on the network be running under the UNIX operating system. Nor is it necessary that all hosts be using the Ethernet front-end processor boards and the Ethernet software. It is necessary, however, that hosts which are to communicate with each other support the TCP/IP protocols and the application programs to be used for communication. If the Address Resolution Protocol (ARP) is not used, it may also be necessary to emulate a common Ethernet address block to accommodate front-end processors or link-level boards from different manufacturers.

C.1 NETWORK DATABASE MANAGEMENT

As a network administrator, you will be required to manage and maintain the network database, which consists of several files and tables. However, before discussing database management, it is important to understand the host naming conventions, and Internet and Ethernet addressing schemes. It is also important to understand the logical-to-physical host address translation concept and the function of the Address Resolution Protocol (ARP). In addition, understanding the routing of communications to nodes on other networks is also important.

This section first discusses the above-mentioned topics and then discusses database management.

HOST NAMES, INTERNET ADDRESSES AND CLASSES, AND ETHERNET ADDRESSES

On Ethernet a host can be referenced by a name (previously assigned to it), its Internet address, or its Ethernet address, depending on the context. A host name is a string that can be up to 32 characters long. The **rwhod**(1M) server, which is used by the **rwho**(1) and **ruptime**(1) utilities, requires that the first nine characters of a host name are alphanumeric and are unique across the network.

The Internet address is a 32-bit quantity. It can be specified in a textual form as digit groups consisting of decimal, octal, or hexadecimal digits separated by period(s), with each digit group corresponding to a byte.

The address class of an Internet address is determined by the value in the high-order bits of the Internet address. The address itself consists of a network number and the number of the host on that network. An Ethernet address is the "physical address" of an Ethernet front-end processor and is a 48-bit quantity.

Host Names

A host can be assigned any name up to 32 characters (letters, digits, and special characters) long; the first character must be a letter. (Uppercase and lowercase letters are considered equivalent.) This assignment is done by including the name (along with the corresponding Internet address and aliases, if any) in the Hosts file (the file /etc/hosts). At the user interface level the host name is used in command lines and in database files. At the system level the name is translated into its logical Internet address, which is eventually translated into the corresponding physical Ethernet address.

Internet Addresses and Address Classes

The Internet address for any host is a four-byte numeric value which can be specified in decimal, octal, or hexadecimal form. (Decimal numbers have no prefixes and do

not begin with a zero; octal numbers are preceded by a 0 [zero]; hexadecimal numbers are preceded by the 0x or 0X character pair.)

The high-order bits of an Internet address determine its class, as follows:

0 in the high-order one bit indicates Class A address.
10 in the high-order two bits indicate Class B address.
110 in the high-order three bits indicate Class C address.

This means that the first byte of a Class A address must be from 0 to 127, the first byte of a Class B address must be from 128 to 191, and the first byte of a Class C address must be 192 to 255.

All hosts on a given network must use the same address class and network number. Thus, it is possible to refer to a network as a Class A, Class B, or Class C network.

The following addresses are reserved and may not be assigned to any host:

CLASS A	0.xxx.xxx.xxx
	127.xxx.xxx.xxx
CLASS B	128.000.xxx.xxx
	191.255.xxx.xxx
CLASS C	192.000.001.xxx
	223.255.255.xxx
	224.000.000.000 to
	255.255.255.255

The Internet addresses of different classes can be broken down as follows:

Class	Bits to Identify Network	Bits to Identify Host
A	8	24
B	16	16
C	24	8

The following example shows a Class A address:

89.0x81.0x01.0x82

The first eight bits specify the network address, which is Class A 89 (decimal). The remaining 24 bits specify the host address, which is 0x81.0x01.0x82.

The following example shows a Class B address:

0x80.0x5.0x01.0x82

The first 16 bits specify the network address, which is Class B 0x5. The last 16 bits specify the host address, which is 0x01.0x82.

The following example shows a Class C address:

0xC0.0.0x6.0x02

The first 24 bits specify the network address, which is Class C 0x6. The remaining 8 bits specify the host address, which is 0x02.

Ethernet Addresses

The Ethernet address is the unique, six-byte physical address for each Ethernet host on a network. This address is in a block assigned by the Xerox Corporation. For Ethernet front-end processors, it is permanently stored in PROM on the processor board. This means that the Ethernet address of any host is the physical address of the Ethernet front-end processor used in that host. The high-order three bytes of this address represent the manufacturer's identification code (also known as the address block); the low-order three bytes represent the host identification.

Normally, you will not need to reference an Ethernet address.

LOGICAL-TO-PHYSICAL HOST ADDRESS TRANSLATION

The TCP/IP protocol software uses logical Internet addresses, while the network hardware uses the physical Ethernet addresses. Thus, in order to send a message from one host to another, the protocol software must translate the Internet address of the remote host to its Ethernet address before transmitting the message to the network.

There are two methods to accomplish the Internet-to-Ethernet address translation:

- The ARP method
- The constant mapping method

The method to be employed is selected prior to downloading the protocol software to the Ethernet front-end processor by modifying the command line for the **netload**(1M) utility in the file /etc/rc.network. The default translation method is ARP.

The ARP Method

The ARP method uses the Address Resolution Protocol. When IP is requested to send a message to an Internet address, it consults ARP for the associated Ethernet address. If ARP does not have this information in its cache, it first broadcasts a packet containing the target host's Internet address. All hosts on the network receive this packet. However, only the target host responds with its Ethernet address. On receipt of a response, ARP on the sending host saves the address translation in its cache and uses it for current and future communication with the target host.

No correlation between the Internet and Ethernet addresses is necessary to use the ARP method. However, it is necessary that all hosts which are to communicate with each other either support the Address Resolution Protocol (ARP) or one of the hosts contain in its Internet-to-Ethernet translation table a *publ* option for the non-ARP-supporting host.

The Constant Mapping Method

When this method is used, the IP on a local host creates an Ethernet address by concatenating the uppermost three bytes of its physical address (manufacturer's ID part) with the lowermost three bytes (host ID part) of the target host's Internet address. The local host obtains the target host's Internet address from the /etc/hosts file.

This method can be used only for Class A networks, where the low-order three bytes of the Internet address of any host must equal the low-order three bytes of the the Ethernet address for that host. In addition, either the uppermost three bytes of Ethernet address for each host must be same or they must be emulated to be same. See the description of the **netload** utility for a description of Ethernet address emulation.

ROUTING ACROSS NETWORKS

Communication between a host on a local network and a host on a remote network is established through gateways that physically connect the local network and the remote network. At the user interface level, the packets intended for a host on the remote network are simply sent to the target host. Internally, however, the target host's Internet address is mapped to the Internet addresses of the gateway associated with it and eventually mapped to the Ethernet address of the gateway. This means that when you send a packet to a host on a remote network, the software sends the packet to the Internet address of an intermediate gateway.

The gateway can be a regular host that also performs the gateway function or it can be a dedicated gateway that performs only the gateway function.

The network software does not support direct attachment to more than one Ethernet. This means that only one Ethernet board can be inserted in a PC.

The gateway mapping information is maintained in the routing table. In order to reach a system on a network to which your PC is not directly attached, the routing table must be initialized. This is typically done at system startup time in the /etc/rc.network script, with the **route**(1) utility. In the simplest case, it is sufficient to install a single default route. This is where all packets not destined for the local network are sent. In a more complex environment, it may be desirable to route packets destined for different networks to different gateways on the network.

If an intelligent gateway on the network generates ICMP redirect messages, the Ethernet modifies its routing tables accordingly. For further information, see the man pages for **route**.

NETWORK DATABASE

The network database consists of the following files and tables. Each of these is described in the sections that follow.

- System Startup File
- Hosts File
- Board Statistics Table
- Internet-to-Ethernet Translation Table
- Routing Table

The System Startup File

The file /etc/rc.network is typically the system startup file. By editing this file, command lines that invoke the **/net/init** and **netload** programs can be added. The **/net/init** program downloads the NX kernel to the Ethernet front-end processor, and the **netload** program downloads the TCP/IP object code to the board. Editing this file also allows you to deselect several default options and/or select several new options. These options include manipulation of Internet and Ethernet addresses and enabling/disabling of the Address Resolution Protocol (ARP).

If new options are to be selected, the command line(s) in the system startup file should be edited and then the system rebooted to download the protocol software to the Ethernet front-end processor.

The command line and the various options for the **init**(1M) and **netload** utilities are described in the man pages.

The Hosts File

The file /etc/hosts is the Hosts file. It contains mappings between the Internet addresses and the names and aliases for various hosts on the network. This file exists on all hosts on the network. When a user executes a utility that references a host by name, the underlying application uses this file to translate the host name into an Internet address.

The mapping for each host is specified on a single line in the following format:

Internet_address system_name [[*alias*] ...]

Internet_address is four-byte value specified in decimal, octal, or hexadecimal using dot notation. Internet address specification formats are described in "Internet Addresses and Address Classes."

system_name is the name of the host associated with the *Internet_address*. It can be any string up to 32 characters long. The first character must be an alphabetic character. *system_name* must be unique across the network. Uppercase and lowercase letters are considered equivalent. Embedded blanks are not permitted in name specifications. The **rwhod**(1M) server, which is used by the **rwho** and **ruptime** utilities, requires that the first nine characters of a host name be alphanumeric and be unique across the network.

alias is an alternate name for the host. Typically, it would be a shorter name. More than one alias for a given host is permitted. An alias need not be unique across the network.

The entry in the Hosts file for the local host must have the alias *localhost*. The **netload** utility uses this to define the Internet address of the local host, and **ftp**(1) requires this. There must also be an entry in all Hosts files for "loopback," which should have an address of 127.0.0.0 and an alias of lb. The **ud**(1M) utility requires this.

For example, the Hosts file for a host named "california" could have the following entries:

```
185.0.5.2 california ca localhost
185.0.5.4 oregon or og
185.0.5.2 mexico mex
127.0.0.0 loopback lb
185.0.5.10 finance fin
185.0.3.22 manufacturing manu
```

The Internet address 127.0.0.0 for the host "loopback" is a special address for the local Ethernet front-end processor. Any communication sent to the host associated with this address is returned to the sender without ever getting on the network. Thus any communication transmitted to the host "loopback" will be received by "california." This is a useful feature for testing/debugging applications.

For communication among all hosts, the /etc/hosts file on each host must contain mappings for all other hosts on the network.

The hosts.equiv File: System Access Control

This section discusses access permissions on a network. When accessing remote hosts, the **rlogin**(1) and **telnet**(1) utilities rely on the server provided with the Ethernet, which uses a standard login mechanism that requires a password. These utilities can "connect" a PC terminal to any other host on the network. (**Rlogin** uses the **rsh** access control if it is set up.)

The **rsh**(1) and **rcp**(1) utilities do not require a name and password. Instead, they refer to the file hosts.equiv (hosts equivalence), which is typically located in the /etc directory, and the file .rhosts (remote hosts), which is located in the user's home directory on the remote system. If the two systems involved in an **rcp** or **rsh** transaction list each other in their own hosts.equiv or .rhosts files, then they share account access permissions. The local host's user name must also be in the remote system's /etc/passwd file (but the user ID numbers do not have to be the same).

The hosts.equiv (hosts equivalence) file contains a list of remote hosts with which it is equivalent, that is, with which it shares account names. The host names must be the standard, complete names, not the aliases. For instance, the following hosts.equiv file designates all systems in the hosts file shown in the preceding section:

```
loopback
california
oregon
mexico
finance
manufacturing
```

The .rhosts file is located in the user's home directory and applies only to that user. It lists the remote systems from which the user may access this local system if the remote systems are not listed in the hosts.equiv file.

In addition, the .rhosts file can be used to allow those who do not have an account on that host to access it through the account of the user who has the .rhosts file. The

entries in the .rhosts file can contain the name(s) of the remote host(s) and the names of the other users who are allowed to use the local user's account. For example, to allow the user "joe" to access your local directory from the remote host "mexico," your .rhosts file would contain the following entry:

```
mexico joe
```

The Board Statistics Table

The board statistics table resides in the Ethernet front-end processor memory. The statistics in this table are compiled and stored by the NX kernel, which is downloaded to and executes on the Ethernet board. The table contains the following counts:

- Packets received from the network
- Packets transmitted to the network
- Packets received with an alignment error – the packet lengths were not in eight-bit multiples
- Packets received with CRC errors
- Packets lost because no receive buffers were available
- Transmissions that failed because of DMA underrun
- Packets that suffered "heartbeat" failure

The table also contains the version numbers for Ethernet software, NX kernel, and Ethernet front-end processor hardware.

Anyone can display the statistics in this table using the **bstat**(1M) utility described below in "BSTAT - Board Statistics Utility." A super user can also reset the board statistics.

The Internet-to-Ethernet Translation Table

The Internet-to-Ethernet translation table resides in the Ethernet front-end processor memory. It is created and dynamically maintained by the Address Resolution Protocol (ARP). This table contains the Internet addresses of those hosts on the network that responded to individual ARP broadcasts and their corresponding Ethernet addresses. The table also contains unresolved host entries.

The entries in this table contain information in the following format:

Internet_address Ethernet_address

The following display shows examples of resolved and unresolved entries. This display is composed by the **arp**(3E) utility, which uses both the Internet-to-Ethernet translation table and the Hosts file for the purpose.

```
munich  (0x59.0x60.0x0.01)  at  0x8:0x0:0x14:0x60:0x0:0x1
munich  (0x59.0x60.0x0.01)  ---  incomplete
munich  (0x59.0x60.0x0.01)  ---  no  entry
```

Anyone can display the entries in the ARP table using the **arp** utility. A super user can also add entries to and delete entries from this table. Additions can be permanent or temporary.

The Routing Table

The routing table resides in the Ethernet front-end processor memory. The initial entries are based on the gateways available as a consequence of hardware/software configuration in the system. The table contains single-line entries for available routes in one of the following two formats:

host_Internet_address gateway_Internet_address
network_Internet_address gateway_Internet_address

In the above formats, *host_Internet_address* is the Internet address of the host on a remote network. Similarly,

network_Internet_address is the Internet address of the remote network. *gateway_Internet_address* is the Internet address of the gateway on the local network, which is used when routing communication to hosts on remote networks.

Anyone can display gateway entries in the routing table using the **route**(1) utility. A super user can also add and delete gateway entries.

C.2 NETWORK SYSTEM UTILITIES

The Ethernet TCP/IP software package includes several utility programs in addition to the protocol software. These utilities provide very important, often-used functions such as file transfer between hosts on the network, terminal emulation, and manipulation of data tables internal to the TCP/IP and the Ethernet front-end processor. These utilities are grouped into two categories:

- Network Application Utilities (**rcp**, **rsh**, **rlogin**, **rwho**, **ruptime**, **ftp**, **telnet**)
- Network System Utilities (**arp**, **bstat**, **init**, **netload**, **route**)

The Network Application Utilities, which are typically used by end users, are described below in "Network Application Utilities."

The Network System Utilities, which are typically used by network/system administrators, are described below in a user's guide style. These descriptions are not meant to be comprehensive. A detailed, formal description of each of these utilities is provided in the *UNIX System V* documentation.

Utility	Name
ARP	The Address Resolution Control Utility
BSTAT	The Board Statistics Utility
ROUTE	The Routing Control Utility

The **arp** utility allows a user to display the Internet-to-Ethernet address mapping for those hosts that have responded to the Address Resolution Protocol (ARP) broadcasts and other entries made manually by a user. It also allows a super user to add and delete mappings from the ARP cache.

The **bstat** utility displays the board statistics (packet traffic) for an Ethernet front-end processor, which in effect means a local host. It also allows resetting the statistics and displaying the version numbers of the current TCP/IP software, NX kernel, and Ethernet board hardware.

The **route** utility allows examination, insertion, and deletion of gateway addresses for internetwork communication.

These utilities can be used by anyone to view current values or states. However, only a super user can alter the values or states.

ARP – THE ADDRESS RESOLUTION CONTROL UTILITY

The **arp** utility displays and manipulates the entries in the Internet-to-Ethernet translation table generated by the Address Resolution Protocol (ARP). This table contains the Internet and Ethernet address mappings entered by a user (usually for non-ARP-supporting hosts) and for those hosts on the network that responded to individual ARP broadcasts. The table also contains unresolved host/address entries. (See the man pages for a detailed, formal description of the **arp** utility.)

NOTE

Only a super user can manipulate the ARP translation table.

Using the ARP Utility

You can display the current entry for a specific host (say "munich") in the Internet-to-Ethernet translation table by entering the command shown below. Depending on whether the entry is a resolved entry, an unresolved entry, or a nonexistent entry, one of the three responses shown will be displayed. (Note that in order to include the host name in the response lines, the **arp** utility uses both the *Hosts* file and the Internet-to-Ethernet translation table to compose the response lines.)

```
$ arp munich
munich (0x59.0x60.0x0.0x1) at 0x8:0x0:0x14:0x60:0x0:0x1
munich (0x59.0x60.0x0.0x1) incomplete
munich (0x59.0x60.0x0.0x1) no entry
```

You can display all the current resolved and unresolved entries in the translation table with the following command:

```
$ arp -a
munich (0x59.0x60.0x0.0x1)  at 0x8:0x0:0x14:0x60:0x0:0x1
india  (0x59.0x10.0x1.0x81) at 0x8:0x0:0x14:0x10:0x1:0x81
london (0x59.0x1.0x33.0x65) at 0x8:0x0:0x14:0x40:0x1:0x89
```

By adding the keyword **entire** to the above command you can display the entire translation table, which includes resolved entries, unresolved entries, entries for which no corresponding entries in the *Hosts* file exist, and "empty" entries.

The following commands can only be used by a super user.

You can delete a host (say "india") from the translation table by using the following command:

```
$ arp -d india
```

The following command inserts an entry, consisting of a host name and the corresponding Ethernet address, in the ARP translation table. Normally, the entry is permanent; if the option **temp** is used, the command makes the entry temporary. If the option **publ** is specified, the entry is "published," which enables a host to respond to ARP broadcast even if the specified Internet address is not its own address. (See the man pages for **arp**(3E) for an explanation of **temp** and **publ** options.)

$ **arp** – **s** *host_name Ethernet_address* [**temp**] [**publ**]

You can insert multiple entries in the translation table by reading them from a text file using the following command:

$ **arp** – **f** *filename*

The format for *filename* is as follows:

host_name Ethernet_address [**temp**] [**publ**]

BSTAT – THE BOARD STATISTICS UTILITY

The **bstat** utility obtains packet-traffic statistics from the Ethernet front-end processor. The NX kernel compiles these statistics and stores them in the board memory. These statistics show the total number of packets received from and transmitted to the Ethernet. Also included in the statistics are counts for the following error conditions. (See the man pages for **bstat(1M)** for a detailed, formal description of the **bstat** utility.)

- Packets received with an alignment error – the packet lengths were not in eight-bit multiples
- Packets received with a CRC error
- Packets lost because no receive buffers were available
- Transmissions that failed because of DMA underrun

- Packets that suffered "heartbeat" failure

The utility displays the statistics since the last statistics reset or software downloading to the Ethernet front-end processor, whichever occurred most recently. **bstat** prints only those statistics that have nonzero values.

In addition to the above functions, the **bstat** utility can reset the statistics on the board and display the version numbers for the current TCP/IP software, NX kernel, and the Ethernet board hardware.

NOTE

Only a super user can reset the board statistics.

Using the BSTAT Utility

To display the statistics, enter the following command:

```
$ bstat
```

The system responds with a display similar to the following one:

```
<nnnn> frames transmitted
<nnnn> SQE test failures
<nnnn> transmissions failed with DMA underrun
<nnnn> frames received
<nnnn> frames received with alignment error
<nnnn> frames received with crc error
<nnnn> frames lost (no receive buffers)
```

To reset the statistics, enter the following command:

```
$ bstat -r
```

The system then displays the current statistics. A subsequent **bstat** command does not display anything since the statistics have been reset (to zero) and **bstat** displays only nonzero statistics.

To display the version numbers for the current TCP/IP software running on the board, the NX kernel, and the

Ethernet board hardware, enter the following command. (The system response is also shown.)

```
$ bstat -v

Ethernet:Firmware Release:4.4
Ethernet:Hardware Release:0.0
Ethernet:Software Release:3.2
```

ROUTE – THE ROUTING CONTROL UTILITY

The **route** utility manipulates the routing table in the Ethernet board memory. Using this utility, you can add a new route, delete an existing route, and show an individual route or all available routes.

See the man pages of **route**(1) for a detailed, formal description of the **route** utility.

The initial entries in this table are based on the gateways available as a consequence of hardware/software configuration in the system. These table entries consist of Internet address for a host on a remote network (or for the remote network) and the associated gateway for communication with that host or network.

NOTE

Only a super user can alter the routing table.

Using the ROUTE Utility

The **route** utility has three options: **add, delete,** and **show.** The format for using these options is as follows:

route add *destination gateway*
route delete *destination gateway*
route show [*destination*]

In the option formats shown above *destination* is the host on the remote network or the remote network itself and *gateway* is the gateway to which the packets should be addressed for eventual delivery to the host on the remote network.

Both *destination* and *gateway* can be specified as names or as Internet addresses. Internet addresses must be specified in the same class as the network class.

Let us say you need to enter a new route – the destination network is "newyork" and the gateway is "ny". You can accomplish this with the following command. (The system response to your command is also shown.) Note that the logical host, network, and gateway names are translated into Internet addresses, shown in four-part dot notation, in the system responses.

```
$ route add newyork ny
add network 29.0.0.0 gateway 89.1.51.101
```

You can delete the above entry with the following command:

```
$ route delete newyork ny
delete network 29.0.0.0 gateway 89.1.51.101
```

If newyork were a host on a remote network instead of a network with the Internet address (say) 29.4.5.6, the above input commands would be identical for the indicated tasks, but the system responses would be as follows:

```
add host 29.4.5.6 gateway 89.1.51.101
delete host 29.4.5.6 gateway 89.1.51.101
```

You can see the route for a specific remote host or network (say "ca") by entering the following command:

```
$ route show ca
show network 27.0.0.0 gateway 89.1.51.101
```

You can see routes for all the remote hosts and networks supported by the local network by entering the following command:

```
$ route show
show host 27.1.2.3 gateway 89.1.51.101
show network 29.0.0.0 gateway 89.1.65.201
show host 27.3.3.3 gateway 89.1.51.101
  .
  .
```

C.3 ETHERNET MESSAGES

After initialization, the Ethernet protocol module may send various status messages to the host system. Normally, these appear on the system console. Optionally, the messages can be diverted to a disk file, or simply ignored, by opening the device /dev/EXOS/errlog. For instance, the following command, typically run from /etc/rc.network, saves error messages to a disk file:

cat – u /dev/EXOS/errlog > /usr/adm/exoslog &

C.4 RUNNING INSTALLATION DIAGNOSTICS

The EXOS 205 installation diskette provided with the Ethernet board contains the **install** program and diagnostics. These diagnostics are divided into two sets: one set is executed by the host PC; the other set is executed by the 80186 CPU on the Ethernet card. These sets of tests exercise the on-board RAM, the 82586 Ethernet Controller, the Ethernet Serial Interface chip, the connected transceiver, and the on-board Ethernet PROM. In addition, the diagnostics run three interrupt tests.

A step-by-step procedure to run the diagnostics is given below. Naturally, before running the diagnostics the Ethernet card must have been installed.

1. Boot your system from DOS.

2. Check CONFIG.SYS for this entry:

   DEVICE=ANSI.SYS

   If this file is present, skip to Step 3. If it is not already present, create the following entry and reboot the system:

   DEVICE=ANSI.SYS

3. Insert the EXOS 205 installation diskette in one of the vacant drives.

4. Enter the following command to input configuration information prior to running the diagnostics:

   A> *x*:install

   IMPORTANT: *x* in the above command must be replaced by the drive name in which the EXOS 205 installation diskette was loaded in Step 3.

   The system then displays the Installation Menu:

INSTALLATION MENU			
Item	**Type**	**Current Setting**	**Selections**
1 System	list	PC	0=PC 1=AT 2=other
2 Window Address	value	A000	
3 Window Size	list	16K	0=16K 1=128K
4 I/O Address	value	310	
5 Interrupt Level	list	2	
6 Update Config file	execute		
7 Run Diagnostics	execute		
Enter Item (number or name)			

5. You need to change items 1, 2 and 5 as follows:

   Item 1: PC → AT
   Item 2: A000 → BC00
   Item 5: 2 → 5

   Type the item number followed by a carriage return. In response to the system prompt, enter the new values.

6. The configuration is now complete. Type 6 or the word "update" to update the Configuration file.

7. Type 7 or the word "run" to run diagnostics. At this point, the system displays the test screen that shows various test names, associated parameters, and a zero error count for each test. The system then begins running the diagnostics. For each test, the test data and status flash in the lower part of the screen.

   Finally, one of the following messages appears on the screen:

   Tests completed: PASSED

   Tests completed: FAILED

   The message is followed by the DOS prompt indicating that control has returned to the operating system.

If the diagnostics failed, you should re-examine the config data, and run the procedure again. If the diagnostics still failed, you should contact Valid for assistance.

If the diagnostics passed, the Ethernet card is healthy and it is installed correctly. At the conclusion of step 7, the configuration information you entered in step 5 is written to the file \xln\hardware\excelan.hdw on the currently logged drive.