MAI® 2000 Technical

Reference Manual

August,1985
006225010

M6225A

MAI BasicFour®

PAGE STATUS

Section	Page No.	Effective Date
Cover		Aug-1985
Title Page	i/ii	Aug-1985
Page Status	iii/iv	Aug-1985
Table of Contents	v through xii	Aug-1985
Preface	xiii/xiv	Jun-1985
Section 1	i	Jun-1985
	1-1 through 3-2	Aug-1985
	4-1	Jun-1985
	4-2	Aug-1985
	4-3 through 4-5	Jun-1985
	4-6	Aug-1985
	4-7 through 4-14	Jun-1985
	4-15	Aug-1985
	4-16	Jun-1985
Section 2	i through 2-12	Aug-1985
	3-1 through 4-1	Jun-1985
	4-2	Aug-1985
	4-3	Jun-1985
	4-4	Aug-1985
	5-1 through 5-2	Jun-1985
	5-3 through 6-4	Aug-1985
	7-1	Jun-1985
	7-2 through 9-2	Aug-1985
	10-1	Jun-1985
	10-2 through 10-5	Aug-1985
	11-1	Jun-1985
	11-2	Aug-1985
Section 3	i	Aug-1985
	1-1 through 1-28	Jun-1985
	2-1 through 2-4	Aug-1985
	2-5	Jun-1985
	2-6	Aug-1985
	2-7 through 2-8	Jun-1985
	2-9 through 2-116	Aug-1985
Section 4	i through 1-3	Jun-1985
	1-3	Aug-1985
	1-4 through 3-13/14	Jun-1985
Index	1 through 3/4	Aug-1985

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Section Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Section Page

SECTION 3 - COMMAND LANGUAGE (Cont'd)

Section Page

MAI(R) 2000 TECHNICAL REFERENCE MANUAL

SECTION 1

SYSTEM INSTALLATION

AUGUST, 1985

LIST OF FIGURES

LIST OF TABLES

INSTALLATION OF YOUR MAI 2000

"Installation" refers to a group of procedures, including setting up the MAI 2000 system hardware, installing and upgrading operating system software. Hardware installation is described in the Basic Four Series 2000 Desktop Computer System Service Manual, BFISD 8079A.

This section covers the following information:

installation planning, a post-installation check to perform after the initial system installation, and software installation and upgrade procedures.

- Installation Planning. Chapter 2 covers planning for the physical placement of the system.

- Post-installation check. Chapter 3 presents a brief procedure to verify that the correct level of system software is installed on the system.

- Software Installation. Two types of installations are discussed in Chapter 4 of this section. They include:

  - Full Installation

  - Upgrade Installation

  - Utilities and BASIC Installation

INTRODUCTION

In this chapter, we describe the pre-installation procedures for your new 2000 system. Before installing the system hardware, some thought should be given to its placement.

You need to make some plans as to where to place the Central Cabinet Assembly (CCA) and the peripheral devices you have ordered. Your planning needs to take into account your own convenience for using the system, availability of electrical power outlets, the routing of cables, and some environmental restrictions.

PHYSICAL PLACEMENT

The exact equipment to be installed varies depending on the specific configuration ordered. Minimally, the system consists of three components: the Central Cabinet Assembly (CCA), a terminal and keyboard, and a printer. The placement of these units is mostly governed by your own comfort, but there are a few restrictions.

CENTRAL CABINET ASSEMBLY

The 2000 CCA can easily fit on an average office desk and still leave desktop work space. If the optional upright stand has been purchased, the system can be stood on end at the side of a desk or other convenient location. In either case, allow enough space to leave about 4 inches on all sides of the CCA for ventilation and cabling. This requires an area approximately 23" deep by 31" wide and at least 6" for its height.

TERMINAL

If you place the Central Cabinet Assembly on a desk, the terminal screen may be placed on top of it. It may also be placed alone on a desk. Position the terminal screen and keyboard so they sit firmly on the desk. Allow about 4 inches on all sides of the terminal screen for ventilation. The actual space required varies with the terminal, but a space 23" by 23" by 23" for the terminal screen is usually enough. The keyboard needs an accessible space about 8" by 21" when it is in use. Make similar plans for as many terminals as you have ordered.

PRINTER

The printer may be placed anywhere that can be reached by the printer cable. Cable for serial printers can be up to 1000 feet, while cable for a parallel printer can be up to 50 feet long. If continuous feed paper will be used, leave additional space for that. Consult the operator's manual included with your printer for specific placement requirements.

MAGNETIC TAPE DRIVE

If the system has a magnetic cartridge streamer (MCS, model TD4403), it should be placed near the central processor unit. The MCS may be placed on top of the CCA. Leave about 4 inches on all sides for ventilation and cabling.

POWER REQUIREMENTS

In general, ordinary wall outlets supply the appropriate voltage necessary for operating the MAI 2000. Specific power requirements are:

Non-European installations	6 AMPS at 110/120 VAC	50/60HZ
European installations	3 AMPS at 220/2M0 VAC	50/60HZ

The power cables for each of the separate units have grounded 3-prong plugs and require their own grounded power outlet. Therefore, if for the CCA, one terminal, one printer, and one magnetic cartridge streamer, a minimum of four 3-prong wall sockets are necessary.

If only 2-prong sockets are available, a properly installed 3-prong adapter may be used. Make sure the adapter is properly grounded.

ENVIRONMENTAL REQUIREMENTS

Temperature, humidity, and altitude generally are not problems, since the usual office environment is well within the limits shown in the following table.

Table 2-1. Environmental Requirements

TYPE OF USE	TEMPERATURE	RELATIVE HUMIDITY	ALTITUDE
Operating	10°C to 38°C (50°F to 98.4°F)	20% to 80%	Sea level to 10,000 feet
Storage	-10°C to 50°C (-50°F to 122°F)	10% to 90%	Sea level to 10,000 feet
Transit	-40°C to 71°C (-104°C to 159.8°C)	95% maximum	Sea level to 33.000 feet

CHECK OPERATING SYSTEM LEVEL

The system software comes installed on the fixed disk. When installation of the 2000 hardware is complete, the release level of the installed software must be checked to verify that it is the same as the level of the backup copy delivered with the system.

The version level is displayed during the boot procedure and by the "sysinfo" command. Both procedures are briefly described below.

BOOTING THE SYSTEM

As described in the MAI 2000 User Guide, the system boots as soon as power is turned on to the CCA. Immediately after the proprietary message and before the "set date" prompt, the operating system level is displayed, for example:

```
 Operating System version: E0S7214, BOSS/IX release 7.2A*nn
  (Aug 22, 1981 14:15)
```

Use the release number to compare as described below.

SYSINFO COMMAND

If the system is already booted, the level can be checked using the "sysinfo" command. With the command interpreter prompt displayed, type:

```
@>sysinfo
```

and press RETURN. The same information shown above will be displayed, along with other information you do not need for the check. Use the release number displayed to compare as described below.

COMPARE OPEATING SYSTEM LEVELS

A backup copy of the operating system is shipped with each system on either a Magnetic Cartridge Streamer (MCS) tape cartridge or a set of floppy diskettes. It is labeled "EOS". The software revision level of the installed operating system and of the backup operating system should be the same.

The revision level of your backup copy of the operating system is printed on the label of the MCS cartridge or floppies. The revision level is a number in the form:

```
7.#N*nn
```

where "#" is a number, "nn" is a number, and "N" is a letter. The "7" identifies the operating system as being for a 2000. The number and letter together indicate the revision level. The "*" identifies a sub-revision level, the "nn" is the specific number of that sub-revision. Note that the "*nn" designation is optional and may not be present.

For instance:

7.2A*14

indicates revision 2A, sub-revision 14 of the 2000 operating system. Compare the level installed with the level on the backup copy. If they are not the same, you should perform the Software Upgrade procedure described in Chapter 4.

INTRODUCTION

Software installation is a process of copying a software package from its release medium, MCS cartridge or floppy diskette, to your system's fixed disk. Special programs are included in the operating system software to make the installation procedures easy to perform.

In this chapter we describe the procedures for installing the software that makes up the base operating system. The base operating system consists of the Operating System software, the Utilities programs, and the Business BASIC programming language.

There are two kinds of software installation procedures:

- o Operating System Installation, of which there are two variants:
  - Full Installation, which includes installing a minimal operating system on the "boot" partition of the fixed disk, as well as the full system software on the "root" partition.
  - Upgrade Installation, which updates the system software files already on the "root" and "boot" partitions.
- o Utilities and BASIC Installation, which is the procedure used to install the system Utility programs and Business BASIC. This procedure must follow each of the preceding procedures.

This chapter contains step-by-step procedures for each of the installations listed above. Procedures for installing other software packages are not described here. Independent software vendors may write their own installation procedures or use variants of the procedure for installing the Utility programs and BASIC. If you need to install any other software package, you must follow the installation instructions supplied with that software package. Refer to the description of the "install" command (Section 3, Chapter 2) for more information on the standard procedure.

Your copy of the new base operating system, whether it is the backup copy of your original operating system or a copy of an updated version, is contained on either a single MCS tape cartridge or a set of floppy diskettes. The number of diskettes may vary from release to release. An MCS cartridge is labeled as containing the three products:

- EOS
- EUT
- EBS

A set of floppies contains floppies labeled:

- INSTALL DISKETTE
- BOOT
- EOS
- EUT
- EBS

For floppies, each of these labels covers a numbered series. For instance, the BOOT set may have BOOT #1, BOOT #2, BOOT #3, and BOOT #4.

The files and programs contained in the INSTALL DISKETTE and BOOT floppies are included on the tape containing the EOS product on MCS cartridge. It includes a boot procedure and the files that are installed on the boot partition on disk.

EUT and EBS contain the Utility programs and Business BASIC respectively.

Note that if you are upgrading the EBS product from 7.1 to 7.2, the BASIC program files will need to be converted from the 7.1 to the 7.2 format. A conversion program, "csave", is provided for this purpose, and is described in the MAI 2000 Business BASIC Reference Manual for 7.2, BFISD 6252D.

The following procedures make use of the floppies or MCS cartridge with the above labels.

FULL INSTALLATION

During a full installation, all operating system files that go on both the "boot" and "root" disk partitions are copied from the backup (or new OS) floppy diskettes or MCS tape cartridge. All other files, such as your data files and applications programs, are destroyed. The disk partitions are either redefined or left intact depending on the boot partition installation procedure selected.

Generally, you perform a full installation:

1. If your system has suffered serious damage to the operating system software on both the boot and the root partitions so that all data on disk has been lost or its condition cannot be determined.

2. If the disk has been reformatted, causing a loss of all data.

CAUTION

> The full installation procedures destroy any user files currently on the disk. Creating default partitions will destroy all files on the disk if the original partitions are different from the default. Installing the boot partition destroys the files on the boot partition, and installing the root destroys the files on the root partition.

The procedure is:

1. Perform an alternate load to the installation medium, tape cartridge or floppy diskette.

2. Install the boot partition; as part of this step, you select either to keep the current partitions (B) or to recreate default partitions (A).

3. Shutdown and reboot. The system automatically roots on the boot partition.

4. Enter the command to install the operating system.

5. Shutdown and reboot.

After this you are ready to install the Utility programs and BASIC.

It is assumed at this point that the system is shutdown. It may either be powered off, or on and displaying the reboot prompt:

```
Press 'RETURN' key to reboot ('^C'=alt-load,
'^S'=self-test):
```

ALTERNATE LOAD

The first step is to perform an alternate load of the system. If the system is powered off, turn it on. While the system is performing its self test, press:

|CTRL| + |C|

When the self test is finished, you will be prompted for the boot device.

If the system is powered on and displaying the reboot prompt, press:

|CTRL| + |C|

In both cases, you are now asked for the boot device:

```
Boot device:
```

For MCS installations, insert the MCS cartridge tape containing EOS into the MCS drive and turn the lock lever. For floppy installations, insert the floppy diskette labeled "INSTALL DISKETTE" into the floppy drive and press the lock button.

If the operating system you are installing is on floppy diskettes, type:

```
fdO
```

and press RETURN.

If the operating system is on an MCS cartridge, type:

```
cs
```

and press RETURN.

You are then asked for the system file:

```
System file:
```

Press RETURN for both floppy and MCS installations.

The system then boots from the device you specified. When it has finished booting, it immediately executes the installation procedure. The installation instructions and options shown in the following figure are displayed at this point.

```
        *** Installation system load complete! ***

   This installation system will be used to perform one of three
   functions. The first is a complete restore of the Winchester
   disk boot partition including creation of default partitions.
   The second option also restores the boot partition files, but
   does not change the current disk partitions. The last choice
   is used to restore the loader to the Winchester disk. Select
   the desired option and type the corresponding letter followed
   by the 'RETURN' key.

     A  -  Restore boot partition, CREATE DEFAULT PARTITIONS

     B  -  Restore boot partition, PRESERVE CURRENT PARTITIONS

     C  -  Restore loader to Winchester disk

     D  -  Redisplay this screen

   OPTION>
```

Figure 4-1. Installation Instructions and Options

BOOT PARTITION INSTALLATION

The displayed screen gives you the available options for installing the boot partition.

Option A creates default partitions on the fixed disk and then installs the files to the boot partition. This option is usually used only after the disk has been formatted.

CAUTION

When the default partitions are created, files currently on the disk are destroyed if the original partitions are different.

Option B keeps the current partitions and installs the files to the boot partition. Your own files, as well as any partitions you have defined, are kept intact by this procedure.

Option C restores the system loader file to the boot partition. No other files are affected. This is used if only the system loader has been damaged.

To install files to the boot partition, type the letter of your selection, either A or B, and press RETURN.

The remaining procedures for installing the boot partition differ slightly if you are installing from floppy diskettes or from MCS. Both procedures are described below.

MCS Installation

When you select to install the boot partition, the procedure begins immediately.

If you selected option B, the program begins installing files to the boot partition. If you selected option A, the program first creates the default partitions and displays information about the disk and the location and size of the partitions created. It then begins copying files to the boot partition.

All of the programs and files required for the installation are contained on a single MCS cartridge, so there is no need to change cartridges during the procedure.

The boot partition is transfered as a filesystem image, so files are not listed. When the partition has been installed, the screen displays:

```
Procedure Complete. Please Re-Boot (type CTRL-D,
shutdown, 'RETURN')
```

Go to the next step, "Shutdown and Reboot."

Floppy Installation

After selecting the boot partition installation option, you are prompted to insert the first BOOT floppy:

```
Insert first floppy 'BOOT #1' into drive, press 'RETURN'
```

Remove the "INSTALL DISKETTE" from the floppy drive, and insert the floppy diskette labeled "BOOT #1", and then press RETURN.

If you selected option B, the program begins installing files to the boot partition. If you selected option A, the program first creates the default partitions and displays information about the disk and the location and size of the partitions created. It then begins copying files to the boot partition.

The screen displays several rows of dots as the files are copied. When all the files from the first diskette have been installed, the screen displays:

```
debe: Done with input volume 1. Do you want to continue?
If so, mount volume 2 on /dev/rfd0 before answering.
Answer YES or NO:
```

Remove the floppy diskette "BOOT #1" from disk drive 0 and insert the floppy diskette "BOOT #2". Then type "YES" and press RETURN.

Continue to switch the floppies as you are prompted until all of the files have been installed. When all of the files from the BOOT floppies have been installed, the screen displays:

```
Procedure Complete. Please Re-Boot (type CTRL-D,
shutdown, 'RETURN')
```

Go to the next step, "Shutdown and Reboot."

Shutdown and Reboot

You now need to shutdown and reboot the system to continue with the installation procedure.

To shutdown, press:

|CTRL| + |D|

You are then asked:

```
single, multi, or shutdown?
```

Type "shutdown" and press RETURN. The system then shuts down and the screen displays the reboot prompt:

```
Press 'RETURN' key to reboot ('^C'=alt-load,
'^S'=self-test):
```

To reboot, press RETURN. The system then goes through the usual boot procedure, except that it roots on the boot partition. When booting is complete and you have entered the time and date, the system administrator prompt is displayed:

```
ADMIN>
```

At this point, the system is rooted on the boot partition.

ROOT PARTITION INSTALLATION

With the system now rooted on the boot partition, you are ready to install files to the root partition. The procedures for installing from floppy diskettes and MCS differ slightly. Both procedures are described below.

MCS Installation

During this procedure you will install the files contained on the MCS cartridge labeled "EOS". This is the same cartridge used in the previous steps.

Insert the cartridge labeled "EOS" into the MCS drive. To begin installing the files, type at the system administrator prompt:

```
osinstall cs
```

and then press RETURN.

The program then prepares the root partition for installation of the EOS product, by making a filesystem on the partition. The screen displays:

```
Making filesystem
making freemap
making fdmap with xxxx fds
making new fds
xxxxx blocks available, writing header
```

Making the filesystem destroys all files currently on the partition.

The program then begins copying files. The program displays the name of each file as it is copied. All of the files are contained on this single cartridge, so there is no need to switch cartridges during the procedure.

When all the files are installed, the screen displays:

```
Procedure Complete.  Please Re-Boot (type CTRL-D,
shutdown, 'RETURN')
```

Shut down and reboot the system.

Floppy Installation

During this procedure you will install the files contained on the floppy diskettes labeled "EOS". All of these floppies must be used.

Insert the floppy labeled "EOS #1" into the floppy drive. To begin installing the files, type at the system administrator prompt:

```
osinstall fdO
```

and then press RETURN.

The program then prepares the root partition for installation of the EOS product, by making a filesystem on the partition. The screen displays:

```
Making filesystem
making freemap
making fdmap with xxxx fds
making new fds
xxxxx blocks available, writing header
```

Making the filesystem destroys all files currently on the partition.

The program then begins copying files. The program displays the name of each file as it is copied. When all of the files from this floppy have been copied, you are asked to remove the current diskette and insert the next one. When you have inserted the next diskette, press RETURN.

When the files on all of the EOS floppies have been installed, the screen displays:

```
Procedure Complete.  Please Re-Boot (type CTRL-D,
shutdown, 'RETURN')
```

Shutdown and Reboot

Shutdown and reboot the system as described above. This time, the system roots on the root partition.

You now need to install the Utility programs and BASIC. The procedures for installing these are described on page INSTALL 4-15 under "Utilities and BASIC Installation."

UPGRADE INSTALLATION

During an upgrade installation, new operating system files are copied to replace out-of-date files on the root partition and the boot partition. All other files, such as your data files and applications programs, remain intact.

This procedure is used when you have received an updated version of the operating system. It may also be used to restore damaged operating system files on the root partition.

The steps are:

1. Perform an alternate boot, rooting on the boot partition

2. Enter the command to update the Operating System software

3. Shutdown and reboot

After this, you are ready to reinstall the Utilities programs and BASIC, if they also need updating.

It is recommended that you backup your user files and system configuration files before you perform an upgrade. The system files to backup are:

```
/etc/bfsdsk
/etc/class
/etc/defaults
/etc/forms
/etc/passwd
/etc/ports
/etc/printers
/etc/ptrans
/etc/terminals
```

It is assumed at this point that the system is shutdown. It may either be powered off, or on and displaying the reboot prompt:

```
Press 'RETURN* key to reboot ('^C'=alt-load,
'^S'=self-test):
```

ALTERNATE LOAD

The first step is to perform an alternate load of the system to root on the boot partition.

If the system is powered off, turn it on. While the system is performing its self test, press:

|CTRL| + |C|

When the self test is finished, you will be prompted for the boot device.

If the system is powered on and displaying the reboot prompt, press:

|CTRL| + |C|

In both cases, you are now asked for the boot device:

```
Boot device:
```

Press RETURN for the default boot device, wd0.

You are then asked for the system file:

```
System file:
```

Type:

```
,/etc/boot.conf
```

and then press RETURN. The leading comma is required. (This is a configuration file that specifies the boot partition as the root device.)

The system then executes the usual boot procedure, except that it roots on the boot partition. When the system is booted, the system administrator prompt is displayed:

```
ADMIN>
```

You are now ready to upgrade the operating system files.

SYSTEM FILE UPGRADING PROCEDURE

With the system now rooted on the boot partition, you are ready to upgrade the system files. The procedures for installing from floppy diskettes and MCS differ slightly. Both procedures are described below.

MCS Installation

During this procedure you update the files contained on the MCS cartridge labeled "EOS".

Insert the cartridge labeled "EOS" into the MCS drive. To begin updating the files, type at the system administrator prompt:

```
osupdate cs
```

and then press RETURN.

The program then begins copying files. The program displays the name of each file as it is copied. All of the files are contained on this single cartridge, so there is no need to switch cartridges during the procedure.

Near the end of the procedure, the program pauses and displays:

```
To install the default configuration files type
'/sys/installetc'

Procedure Complete.  Please Re-boot (type CTRL-D,
shutdown, 'RETURN')

ADMIN>
```

There is usually no need to install these system files. If you have modified any of them, reinstalling will lose your modifications, and you will have to repeat the modifications or restore them from backup.

Basic Four will notify you with the release letter for a new OS level if you must install these system files. They must be installed, for instance, if a file format has been changed.

If you want to keep your current files, the procedure is complete. Shutdown and reboot the system. If you want to restore the default files, follow the instructions below, "Updating System Files".

Floppy Installation

During this procedure you update the files contained on the floppy diskettes labeled "EOS". All of these floppies must be used.

To begin updating the files, type at the system administrator prompt:

```
osupdate fdO
```

and then press RETURN.

You are then prompted to insert the first EOS diskette:

```
Insert diskette number 1, <RETURN> when ready:
```

The program then begins copying files. The program displays the name of each file as it is copied.

When all of the files from this floppy have been copied, you are asked to remove the current diskette and insert the next one. When you have inserted the next diskette, press RETURN. Follow the prompts until all the diskettes have been used.

Near the end of the procedure, the program pauses and displays:

```
To install the default configuration files type
'/sys/installetc'

Procedure Complete.  Please Re-boot (type CTRL-D,
shutdown, 'RETURN')

ADMIN>
```

There is usually no need to install these system files. If you have modified any of them, reinstalling will lose your modifications, ana you will have to repeat the modifications or restore them from backup.

Basic Four will notify you with the release letter for a new OS level if you must install these system files. They must be installed, for instance, if a file format has been changed.

If you want to keep your current files, the procedure is complete. Shutdown and reboot the system. If you want to restore the default files, follow the instructions below, "Updating System Files".

Updating System Files

The usual update of operating system software files is completed when this message and prompt are displayed:

```
To install the default configuration files type
'/sys/installetc'

Procedure Complete.  Please Re-boot (type CTRL-D,
shutdown, 'RETURN')

ADMIN>
```

If the release letter from Basic Four specifies that you must install the new default system files, or if you have decided to restore them, type:

```
/sys/installetc .
```

and then press RETURN.

The program then displays:

```
Selectively installing system configuration and data
 files...

 As each file is displayed, type 'y + RETURN' to
 install and RETURN to skip.
```

and the name of the first system file to be restored.

To keep the current file, press RETURN. To install the default for this file, type "y" and press RETURN.

Repeat this procedure for each file as its name is displayed. You can choose for each file whether or not to restore it.

When you have specified the action for each file, the screen displays:

```
 Procedure Complete.  Please Re-Boot (type CTRL-D,
 shutdown, 'RETURN')
```

You are now ready to shutdown and reboot.

Shutdown and Reboot

Shutdown and reboot the system as usual. This time the system roots on the root partition.

You now need to install the Utility programs and BASIC to include the updates to these files. The procedures for installing these are described below, "Utilities and BASIC Installation."

UTILITIES AND BASIC INSTALLATION

The Utilities programs and Business BASIC programming language are integral parts of the 2000 operating system. Many optional software packages assume they are present on the system, and will not run in their absence. Any time you install or upgrade the operating system, the Utilities and BASIC must also be installed.

The following procedures should only be run in single user mode. It is assumed that the system has been booted normally and that it is rooted on the root partition.

Your copies of the Utility programs and BASIC are contained on either a series of floppy diskettes or on the same cartridge that contains the operating system software, EOS. In either case, the cartridge or floppies will be labeled "EUT", for the Utility programs, and "EBS" for BASIC.

The procedures for installing the Utilities and BASIC from floppies and MCS differ slightly. Both procedures are described below.

In both cases, the procedure simply makes use of the command language command "install." Refer to the discussion of this command (Section 3, Chapter 2) for further details.

Note that if you are upgrading BASIC (EBS) from 7.1 to 7.2, all BASIC program files need to be converted to the new file format. The conversion program, "csave", is described in the MAI 2000 Business BASIC Reference Manual, BFISD 6252D.

MCS INSTALLATION

To install the Utility programs and BASIC from MCS, insert the cartridge containing EUT and EBS into the MCS drive and turn the lock lever.

At the system administrator prompt:

```
ADMIN>
```

type:

```
install cs EUT EBS
```

and then press RETURN.

The program looks for EUT on the cartridge, displays its label, and begins copying files. The program displays the name of each file as it is copied. When it has finished copying files, the program ends and the system administrator prompt is repeated.

When you have installed the Utilities and BASIC, the procedure is finished. There is no need to shutdown and reboot.

FLOPPY INSTALLATION

To install the Utility programs and BASIC from floppy, insert the first floppy in the EUT series into the floppy drive and press the lock button.

At the system administrator prompt:

```
ADMIN>
```

type:

```
install fdO EUT
```

and then press RETURN.

The program looks for EUT on the floppy, displays its label, and then begins copying files. The program displays the name of each file as it is copied. When all the files on one floppy have been copied, you are prompted to insert the next floppy in the set. When it has finished copying the files from the last floppy, the program ends and the system administrator prompt is repeated.

Next, insert the first floppy in the BASIC series into the floppy drive, and type:

```
install fdO EBS
```

and then press RETURN.

The program displays the set label, and begins copying files. The program again displays the name of each file as it is copied. When all the files on one floppy have been copied, you are prompted to insert the next floppy. When it has finished copying files from the last floppy, the program ends and the system administrator prompt is displayed.

This is the end of the procedure. There is no need to reboot.

MAI(R) 2000 TECHNICAL REFERENCE MANUAL

SECTION 2

SYSTEM TAILORING

AUGUST, 1985

LIST OF FIGURES

LIST OF TABLES

OVERVIEW

This section describes the MAI 2000 system configuration options. These options allow you to tailor several system features.

This section does not provide complete operating instructions for the programs, utilities, and commands that adjust the configuration parameters. It references the sections of the user documentation where more information can be found.

SUMMARY OF CONFIGURATION FEATURES

The following table summarizes the configuration features. It also gives the names of the utility and/or command(s) that are used to change the feature, and the section of the MAI 2000 User Guide that contains additional information. The subjects listed in this table are discussed later in this section.

Table 1-1. Summary of Configuration Features

CONFIGURATION FEATURE	UTILITY/COMMAND	USER GUIDE
Port Configuration	configure/-	Utilities
Login Procedures	oprinfo/?	Utilities
System Startup and Shutdown Procedures	---/ved ---/vconf	Commands Not Discussed
BOSS/IX Operating Parameters	---/vconf	Not Discussed
Menus and Menu Security	menuedit/?	Utilities
Moving the System Console	configure/-	Utilities
Define Disk Partitions	---/usb	Not Discussed
Modify the NVRAM	---/---	

INTRODUCTION

Port configuration is the process of setting the operational characteristics of the serial and parallel ports on the CMB and the serial ports on the four-ways. These ports may be used to connect terminals and printers. These devices can be connected directly, using appropriate cables, or be remotely located and connected, using modems and communication circuits.

Port 1 can also be configured to be dedicated to synchronous communications.

Port 2 is the only parallel port on the MAI System 2000. All other ports are serial ports (see figure 2-1).

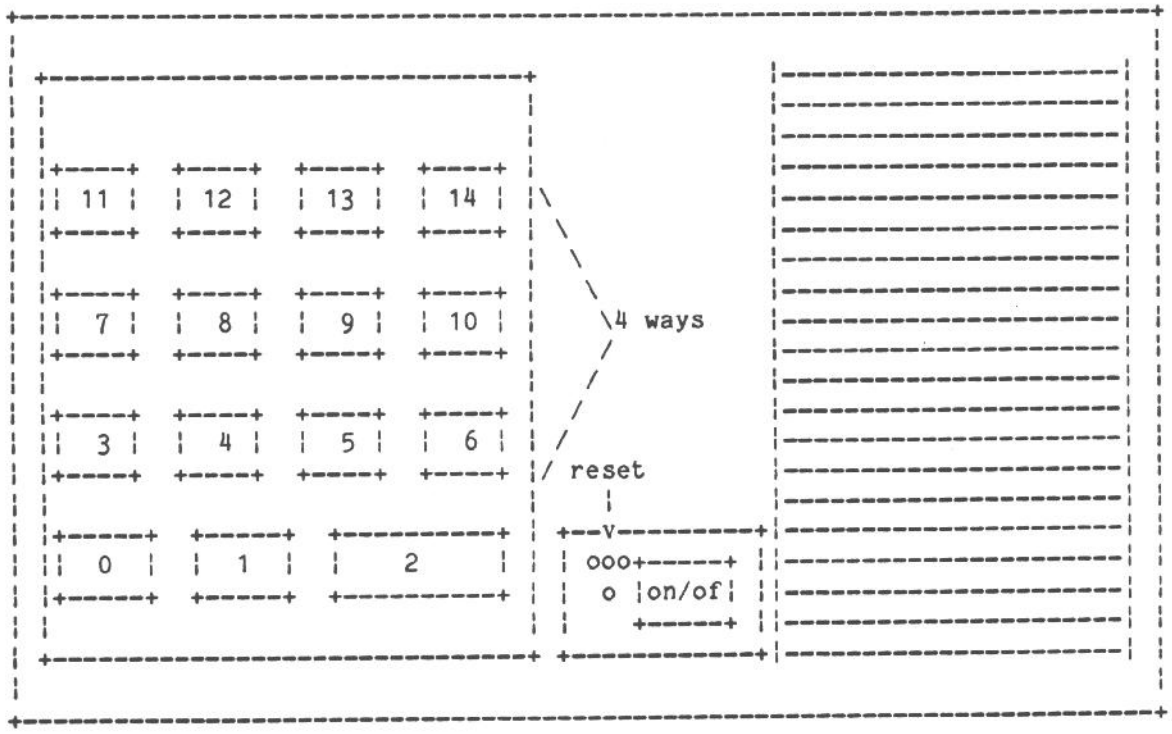

Figure 2-1. Rear View of MAI 2000

PARALLEL DEVICES

The parallel port (port 2) is used to attach a parallel printer on the MAI 2000. A parallel printer cannot be attached to any other port.

PARALLEL PRINTER CONNECTION

On the 7.1 O.S. release, there are three kinds of parallel printers that can be used: the 150 and 300 Line Per Minute Printers (1pm printer, Model 4201), and the Dual Mode 80/200 line per minute printer (dmp printer, Model 4214).

On the 7.2 O.S. the driver has been enhanced to support a variety of industry standard printers (ISP). The enhanced capability supports at least these additional parallel printers:

Centronics 351
Okidata 92A
Dutch Mannesman/Tally 180
Epson FX80

For MAI Basic Four parallel printers, the cable part number is 907759-xxx (xxx indicates the cable length). The ends of the cable are different so they can not be incorrectly installed.

Printer cables for ISP parallel printers are not sold by MAI Basic Four. For the type of cable required for an ISP printer, refer to the documentation for that device.

PARALLEL PRINTER TRANSMISSION CHARACTERISTICS

You cannot change a parallel printer's transmission characteristics. Any attempt to change them through the Port Configuration utility will be ignored. The Port Configuration utility is discussed in more detail later in this chapter.

SERIAL DEVICES

Serial devices include terminals and serial printers. There are three basic steps that you must perform to attach a serial device to your system:

1. Connect the terminal or printer (with the proper cable) to a port on the system.

2. Set the terminal's or printer's data transmission characteristics.

3. Configure the port each device is connected to.

Port 1 (CMB) may be used for serial communication products. You will need to consult the installation manual for any communication product you install. The port can be reserved for synchronous or asynchronous communications using the Port Configuration utility as described below.

TERMINAL AND PRINTER CONNECTION

There are several models of terminals supported on the MAI 2000. The standard terminals are:

EVDT - Model 4309
EDT - Model 4310
VDT7270 - Model 4301

Configuration of terminals other than these standard terminals is also supported, but the mnemonic tables must be created for them to work properly (see below), and they may require special cables.

The system console must be connected before the system can be started. The console is initially configured to be on the first 4-way. If the system does not have a 4-way:

- connect a terminal to the first CMB port, turn the system on and press CTRL + 1 while the self-tests are running
- If the above does not work, turn it on again and press ESCAPE repeatedly until the debug prompt is displayed

With the system in debug, the system console can be moved to a CMB port. Instructions for moving the system console are given in Chapter 9.

There are several models of serial printers supported on the MAI 2000. The standard printers are:

Letter Quality Printer - lqp printer, Model 4206
Dual Mode 80/200 Line Per Minute Printer with serial interface - dmp, Model 4214
80 Column Terminal Printer - tp80 printer, Model 4208
Tritel
Whisper 120/160 Character Per Second Printer - Model 4213

These terminals and printers can be put on ports 0 and 1 on the CMB and on any 4-way port, ports 3 - 14 (optional). Refer to figure 2-1. Other non-Basic Four (referred to as "foreign") terminals and serial printers can also be configured.

The part number of the cable used for ports 0 and 1 is 907754-xxx. The part number of the cable used for 4-way ports is 907753-xxx. The "xxx" varies and indicates the cable length. Foreign terminals and printers may require different cables.

Industry Standard Printer

For 7.2, a configurable, general purpose printer device, called the Industry Standard Printer (ISP), has been added to supply support for a wider selection of serial printers. These additional serial printers include at least:

Centronics 351
NEC Spinwriter
Dutch Mannesmann/Tally 180
HP Laser Jet

SET SERIAL TRANSMISSION CHARACTERISTICS

Some terminals and printers allow changing the baud rate, the number of data bits, the number of start/stop bits, and the parity for serial devices, and several other characteristics.

On the EDT and EVDT terminals, these characteristics are configurable through a setup menu. On the 7270 terminal and some printers, they are configurable by switches and jumpers. Please refer to the technical manual to determine how to set the transmission characteristics you desire.

These settings must match those specified for the port configuration.

On printers purchased from MAI Basic Four, you can select Basic Four interface or industry standard interface (not be confused with ISP). The industry standard interface. The allowed characteristics for each are described in the service manual for the printer. The industry standard interface may be used across many communications networks. The Basic Four interface supplies transmission verification.

CONFIGURING SERIAL DEVICES

There are three steps to configure serial ports on the MAI 2000:

1. Set up the port "strapping", or jumpers. The ports are originally strapped for a locally connected device, and changing the strapping is only required if the device is to be connected to a modem (see "RS232 Support" below).

2. Software-configure the port for the proper transmission characteristics using the Port Configuration utility. These must match the characteristics for the terminal or printer.

3. If more than one printer are being cofigured, it is necessary to modify the system configuration. This is done with the "vconf" command.

CONFIGURE UTILITY

The Port Configuration utility ("/util/configure") sets the transmission characteristics of the serial ports. "Configure" also allows you to add and remove devices, modify the transmission characteristics of devices already hooked up, move devices to different ports, and report the current configuration of the system.

A procedural description of the utility is given in the MAI 2000 User Guide. In the following paragraphs, the parameters required for each device type are described.

FILES "Configure" will modify the following system files in order to perform software configuration:

/etc/defaults	- specifies spooling on/off for printers
/etc/ports	- contains information about each port configured for a device
/etc/terminals	- contains information on each terminal configured on the system
/etc/plotters	- contains information about each graphics device
/etc/printers	- contains information on each printer configured on the system
/dev	- directory containing device file for each device on the system

DEVICE TYPE Port 2 on the CMB is a parallel port, and can only be configured for a parallel printer.

Port 1 on the CMB can be configured for terminals, graphics terminal, printer, or communications. If it is configured for communications, no further parameters are requested.

All other ports can be configured for terminals, graphics terminals, or printers.

TERMINALS Below are the parameters and acceptable values to be entered for configuring a video display terminal.

Terminal Type Terminal types are listed for each terminal type with entries in the "/etc/ttymntbl" directory. These include standard MAI Basic Four Inc terminals, an entry for generic "other" terminals, and any terminals for which custom mnemonics tables have been created.

Device Name The names offered for terminals are **/dev/tty<n>", where <n> is a number from 0 - 13.

Start Process The default start process for terminals and graphics terminals is:

exec,t=/dev/tty#,/bin/login{,t=0}

where # is the terminal device number. The optional "t=0" applies only to the system console. To change the start process, the full path name of the process must be given. Note that the format is the same as for command interpreter lines except that the comma must be used as the field separator.

Comment This is an uninterpreted field.

Input/Output Baud Rates The allowed values are displayed in a menu. Select the values matching the baud rates of the device.

Parity	The allowed values are displayed in a menu. Select the value matching the parity of the device.
Stop Bits	The allowed values are displayed in a menu. Select the value matching the number of stop bits required by the device.
Character Length	The character length can be 7 or 8 bits. Select the character length matching the device.
Protocol	The protocol options are displayed on a menu. Select the transmission protocol matching the device.
Input/Output Translation Files	If a terminal requires character translation, the name of the translation table must be specified. The available translation files, located in the "/etc/ttyxlt" directory, are listed in a menu. Instructions for the creation of translation files are given in the next chapter.
Slave Printer Attached	If a slave printer is to be attached to the terminal, specify "yes" for this field. Refer to the slave printer configuration description below for further discussion.
GRAPHICS TERMINALS	The parameters for configuring a graphics terminal are the same as for a non-graphics terminal, plus the following.
Graphics Terminal Name	The name for graphics terminals is a string of up to 10 characters. The default "T#", where # is the number of the terminal device used for the device name.
Graphics Terminal Parameters	This parameter specifies the contents of four additional fields in the "/etc/plotters" file used by the graphics programs. Refer to the graphics documentation for the required parameters. The default is ":::::" indicating no parameters.
PRINTERS	Below are the parameters and acceptable values for configuring a printer.
Printer Type	The available printer types are displayed on a menu. Select the printer type matching the printer being configured. The "isp", or industry standard printer, and "special" are generic entries covering many printers other than those supplied by MAI Basic Four Inc. The "isp" option supplies limited mnemonics support, and the "special" option supplies no mnemonic support.
Device Name	The device names offered for printers are "/dev/lp" or "/dev/p#" where # is a number from 0 - 12.
Comment	This is an uninterpreted field.
Input/Output Baud Rates	The allowed values are displayed in a menu. Select the values matching the baud rates of the device.

Parity	The allowed values are displayed in a menu. Select the value matching the parity of the device.
Stop Bits	The allowed values are displayed in a menu. Select the value matching the number of stop bits required by the device.
Character Length	The character length can be 7 or 8 bits. Select the character length matching the device.
Protocol	The protocol options are displayed on a menu. Select the transmission protocol matching the device. If Basic Four Interface is specified (see below), this field should be set to no protocol.
Basic Four Interface	If the printer uses the Basic Four interface, specify "yes" for this parameter. The Basic Four interface takes precedent over any parameters in conflict with it. For example, if protocol is specified in addition to Basic Four interface, the protocol field is ignored.
Spooling	If the printer is to be spooled, specify "yes" for this parameter.
Read/Status and Write Timeout	The timeout values may be any number from 1 to 5000, indicating the number of seconds.
PLOTTERS AND GRAPHICS PRINTERS	The parameters for configuring plotters and printers with graphics capability are the same as for printer, plus the following:
Plotter Name	The plotter name for graphics terminals is a string of up to 10 characters. The default is the same as the printer name.
Plotter Parameters	This parameter specifies the contents of four additional fields in the "/etc/plotters" file used by the graphics programs. Refer to the graphics documentation for the required parameters. The default is ":::::" indicating no parameters.
SLAVE PRINTERS	The parameters for configuring a slave printer are the same as for a printer with the following exceptions.
Slave Printer Type	The available slave printer types are displayed on a menu. The EDT (Model 4310) terminal supports the tp80, whisper, lqp, and isp printers as slaves. The EVDT (Model 4309) and VDT7270 (Model 4301) support the tp80 and whisper printers as slaves.
Printer Name	Slave printers are by default given the name "1#", where # is the number of the terminal it is slaved to. If the terminal name is changed, the name of its slave printer is also changed.

CONFIGURING MORE THAN ONE PRINTER

Some of the system configuration parameters are involved in printer configuration. The configuration file for the normal system startup is contained in the file n/etc/conf" on the boot partition.

The system is originally configured to support only one printer. If you add extra printers, you must modify the configuration file to support these additional printers. This involves increasing the values for a few parameters in the configuration file.

The current system configuration is displayed by this command:

```
ADMIN>vconf -os
```

(Note that this assumes the system is currently booted with the normal configuration.)

Find the following parameters in the display, and note the current value (base system values are given here):

- Number of processes allowed = 23
- Maximum number of lu's = 96
- Maximum number of open files = 40
  (on the same line with lu's)
- Maximum number of printers = 1

For each printer in addition to the one included in the base configuration, you need to increase the values of these parameters by the following amounts:

- Number of processes, increase by 1
- Maximum number of lu's, increase by 6
- Maximum number of open files, increase by 2
- Number of printers, increase by 1

For each printer being added, add the required increase for each parameter to its current value. Mount the boot partition to the directory "/mnt" and then, using these values, enter the following command:

```
ADMIN>vconf /mnt/etc/conf -save procs=# lus=#
            opens=# printers=#
```

where # is the size you have calculated for each parameter.

This completes the procedure. Note that the modified configuration does not take effect until the system has been fully shut down and re-booted.

COMMUNICATIONS

The devices menu for port 1 includes "communication port" as an option. This option dedicates port 1 to synchronous communcations, disabling all asynchronous support from the port. No further parameters are requested for this configuration option.

RS232C SUPPORT

This section describes in detail, the RS232C support provided on the MAI 2000. The CMB and 4-way ports function identically for asynchronous communications. Only port 1 supports synchronous communications.

Standard MAI System 2000 serial device cables are wired pin-to-pin. The following wiring diagrams (Figures 2-2 and 2-3) depict the required logical strappings for both 4-way and CMB. Devices that use a signal other than DTR for flow control will require a specific cable or adapter.

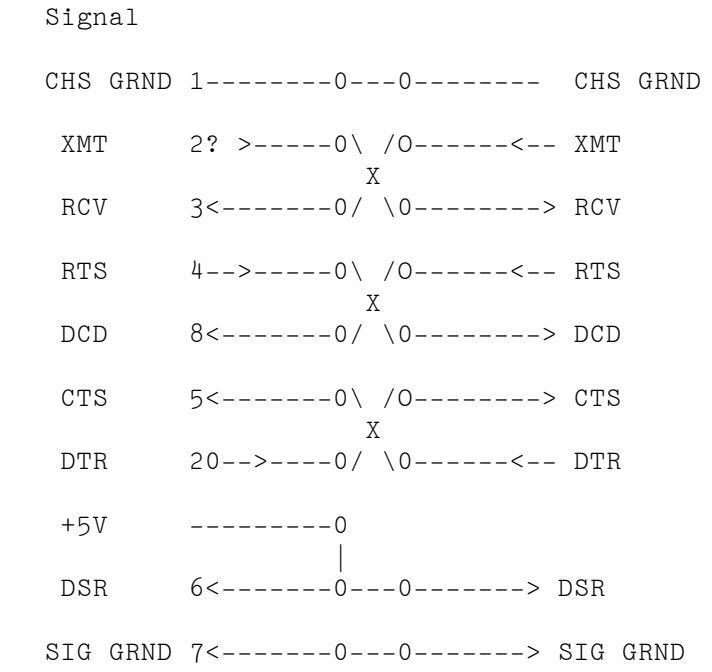

Figure 2-2. Standard Locally Connected Device Setup

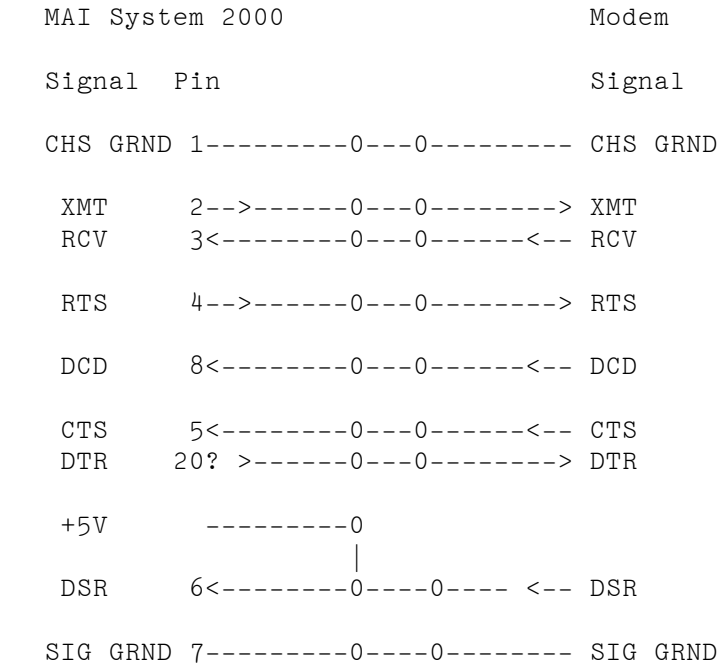

Figure 2-3. Standard Modem Setup

FLOW/MODEM CONTROL

Mode 1: No Flow/Modem Control

This mode is used for dumb terminals, printers and computer-to-computer communications.

a) Assert RTS and DTR.

b) Ignore CTS, DSR and DCD.

c) Ignore Xon and Xoff codes.

This mode is established by turning Auto Enable OFF, turning RTS and DTR on and disabling external status interrupts.

Mode 2: Xon/Xoff Flow Control

a) Assert RTS and DTR.

b) Ignore CTS, DSR and DCD

c) Enable Xon/Xoff Flow Control.

d) Pass all input (incl. Xon & Xoff) through to the host.

e) When testing input for Xon/Xoff codes, compare low order 7 bits only.

Mode 3: CTS/DTR Flow Control

This mode will be used to talk to smart terminals, printers and any other devices that use hardware signals for flow control.

a) Assert DTR when able to receive input.

b) Assert RTS (OK to toggle due to Auto Enable mode).

c) Output only when CTS is high (this is a feature of Auto Enable mode).

d) Accept input when DCD is high. (This should be high because of the pull up resistor described above.)

e) Ignore DSR.

f) Ignore Xon/Xoff.

This mode is established by turning Auto Enable on, turning DTR on and disabling external status interrupt.

Mode 4: CTS/DTR and Xon/Xoff Combined Flow Controls

a). Assert DTR when able to receive input.

b) Assert RTS.

c) Output only when CTS is high.

d) Accept input only when DCD is high.

e) Ignore DSR

f) Enable Xon/Xoff flow control for one direction only. Honor Xon/Xoff codes received from a remote device but do not send Xon/Xoff codes to the remote device.

This mode is established by turning Auto Enable on, turning DTR on and disabling external status interrupt.

Mode 5: Modem Control

a) Assert RTS and DTR.

b) Output only when CTS is high.

c) Accept input only when DCD is high.

d) Interrupt host on transition of DSR.

This mode is established by turning Auto Enable on, turning RTS and DTR on and enabling external status interrupt.

Mode 6: Modem Control and Xon/Xoff Flow Control

a) Assert RTS and DTR.

b) Output only when CTS is high.

c) Accept input only when DCD is high.

d) Interrupt host on transition of DSR.

e) Enable Xon/Xoff Flow Control.

f) Pass all input (incl. Xon & Xoff) through to the host.

g) When testing input for Xon/Xoff codes, compare low order 7 bits only.

This mode is established by turning Auto Enable on, turning RTS and DTR on and enabling external status interrupt.

TRANSLATION FILES

Character translation is used when the character codes generated by a particular terminal don't match the character codes expected by the computer. "Configure" allows you to specify input and output terminal character translation table files. A character translation table is a one-to-one mapping of characters from one value to another. Translation can take place on input, output, or both. The default is to have no terminal character translation.

If you need to enter a character code that your terminal doesn't generate (have a key for), you can specify input character translation. You would choose one of the keys on your terminal and have it translated (on input) to the character you want generated. You do this by creating a character translation file and specify, via "configure", that it is to be used for input translation. It follows then, that if you have a terminal that can't display certain characters being sent to it, you would want to translate them to characters your terminal can print. This would be accomplished by specifying output translation.

Before describing the translate table creation process, a brief discussion of character sets is in order. The MAI 2000 uses a 256 character set with each character represented in one byte. The character set is divided into 2 portions. There are the "low-order" 128 characters, which can be represented in 7 bits, with the upper 8th bit set to 0, and the "high-order" 128 characters, represented in 7 bits with the upper 8th bit set to 1.

In the United States, the "low-order" 128 characters make up the U.S. English (ASCII character protocol) set. This set includes codes for all the characters on standard keyboards used in the United States (see Figure 3-1). The "high-order" 128 characters make up the additional characters required for the international character set (see Figure 3-2). These characters are many of those used in other languages. If you need your terminal to handle text written in German, for example, you would want to use a character set like that shown in Figure 3-3. This requires using the characters in both the "low-order" and "high-order" portions of the character set.

Some terminals cannot generate characters from the "high-order" portion of the character set, and thus, would not be able to display the international characters directly. The EDT and the EVDT terminals can be set up to generate any of the 256 characters in the character set. Consult the respective operator manuals for these terminals. VDT7270 terminals with, for example, the German ISO character set, cannot generate the correct character codes. This would require the use of translation tables.

BINARY LSB → / MSB ↓	HEX	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0	NUL <0>	SOH <1>	STX <2>	ETX <3>	EOT <4>	ENQ <5>	ACK <6>	BEL <7>	BS <8>	HT <9>	LF <10>	VT <11>	FF <12>	CR <13>	SO <14>	SI <15>
0001	1	DLE <16>	DC1 <17>	DC2 <18>	DC3 <19>	DC4 <20>	NAK <21>	SYN <22>	ETB <23>	CAN <24>	EM <25>	SUB <26>	ESC <27>	FS <28>	GS <29>	RS <30>	US <31>
0010	2	SP <32>	! <33>	" <34>	# <35>	$ <36>	% <37>	& <38>	' <39>	(<40>	) <41>	* <42>	+ <43>	, <44>	- <45>	. <46>	/ <47>
0011	3	0 <48>	1 <49>	2 <50>	3 <51>	4 <52>	5 <53>	6 <54>	7 <55>	8 <56>	9 <57>	: <58>	; <59>	< <60>	= <61>	> <62>	? <63>
0100	4	@ <64>	A <65>	B <66>	C <67>	D <68>	E <69>	F <70>	G <71>	H <72>	I <73>	J <74>	K <75>	L <76>	M <77>	N <78>	O <79>
0101	5	P <80>	Q <81>	R <82>	S <83>	T <84>	U <85>	V <86>	W <87>	X <88>	Y <89>	Z <90>	[<91>	\ <92>	] <93>	^ <94>	_ <95>
0110	6	` <96>	a <97>	b <98>	c <99>	d <100>	e <101>	f <102>	g <103>	h <104>	i <105>	j <106>	k <107>	l <108>	m <109>	n <110>	o <111>
0111	7	p <112>	q <113>	r <114>	s <115>	t <116>	u <117>	v <118>	w <119>	x <120>	y <121>	z <122>	{ <123>	| <124>	} <125>	~ <126>	DEL <127>

Figure 3-1. ASCII (Low Order) Character Set

BINARY LSB → / MSB ↓	HEX	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1000	8	NUL <128>	SOH <129>	STX <130>	ETX <131>	EOT <132>	ENQ <133>	ACK <134>	BEL <135>	BS <136>	HT <137>	LF <138>	VT <139>	FF <140>	CR <141>	SO <142>	SI <143>
1001	9	DLE <144>	DC1 <145>	DC2 <146>	,DC3 <147>	DC4 <148>	NAK <149>	SYN <150>	ETB <151>	CAN <152>	EM <153>	SUB <154>	ESC <155>	FS <156>	GS <157>	RS <158>	US <159>
1010	A	SP <160>	Â <161>	Ä <162>	Á <163>	Æ <164>	Å <165>	Ê <166>	Ê <167>	Ë <168>	Ë <169>	Ì <170>	Ï <171>	Ì <172>	Ô <173>	Ö <174>	Ó <175>
1011	B	Ø <176>	Ù <177>	Ô <178>	Ü <179>	Ù <180>	Ç <181>	Ñ <182>	ã <183>	ê <184>	ï <185>	õ <186>	ü <187>	ì <188>	ò <189>	¨ <190>	B <191>
1100	C	à <192>	â <193>	ä <194>	á <195>	œ <196>	å <197>	è <198>	ê <199>	ë <200>	é <201>	ì <202>	ï <203>	ì <204>	ô <205>	ö <206>	ó <207>
1101	D	ø <208>	ù <209>	û <210>	ü <211>	ú <212>	ç <213>	ñ <214>	¿ <215>	¡ <216>	§ <217>	£ <218>	° <219>	˚ <220>	2 <221>	3 <222>	¢ <223>
1110	E	Ã; <224>	L <225>	⊥ <226>	⌟ <227>	— <228>	— <229>	· <230>	= <231>	⅛ <232>	¼ <233>	⅜ <234>	½ <235>	⅝ <236>	¾ <237>	⅞ <238>	g <239>
1111	F	Õ <240>	⌐ <241>	⊤ <242>	⌝ <243>	⊢ <244>	+ <245>	⊣ <246>	¤ <247>	¥ <248>	µ <249>	© <250>	® <251>	Å <252>	™ <253>	º <254>	DEL <255>

Figure 3-2. International (High Order) Character Set

BINARY LSB → / MSB ↓	HEX	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0	NUL <0>	SOH <1>	STX <2>	ETX <3>	EOT <4>	ENQ <5>	ACK <6>	BEL <7>	BS <8>	HT <9>	LF <10>	VT <11>	FF <12>	CR <13>	SO <14>	SI <15>
0001	1	DLE <16>	DC1 <17>	DC2 <18>	DC3 <19>	DC4 <20>	NAK <21>	SYN <22>	ETB <23>	CAN <24>	EM <25>	SUB <26>	ESC <27>	FS <28>	GS <29>	RS <30>	US <31>
0010	2	SP <32>	! <33>	" <34>	# <35>	$ <36>	% <37>	& <38>	' <39>	(<40>	) <41>	* <42>	+ <43>	, <44>	- <45>	. <46>	/ <47>
0011	3	0 <48>	1 <49>	2 <50>	3 <51>	4 <52>	5 <53>	6 <54>	7 <55>	8 <56>	9 <57>	: <58>	; <59>	< <60>	= <61>	> <62>	? <63>
0100	4	§ <64>	A <65>	B <66>	C <67>	D <68>	E <69>	F <70>	G <71>	H <72>	I <73>	J <74>	K <75>	L <76>	M <77>	N <78>	O <79>
0101	5	P <80>	Q <81>	R <82>	S <83>	T <84>	U <85>	V <86>	W <87>	X <88>	Y <89>	Z <90>	Ä <91>	Ö <92>	Ü <93>	^ <94>	_ <95>
0110	6	` <96>	a <97>	b <98>	c <99>	d <100>	e <101>	f <102>	g <103>	h <104>	i <105>	j <106>	k <107>	l <108>	m <109>	n <110>	o <111>
0111	7	p <112>	q <113>	r <114>	s <115>	t <116>	u <117>	v <118>	w <119>	x <120>	y <121>	z <122>	ä <123>	ö <124>	ü <125>	ß <126>	DEL <127>

Figure 3-3. German ISO Character Set

Character translation files can be created using the "makettyxlate" command. See the Command Language section of this manual for more information on this command. The format for makettyxlate is

```
makettyxlate <table name>
```

where <table name> is the name of the file you want created with the character translation information.

The translation table is a 256 byte long record, arranged as a 16-by-16 (00 through FF) position matrix, which specifies how each numeric representation is to be translated.

As in the example before, if you wanted to create a translation table to translate high order German ISO characters into ones your terminal could display, you would want to specify output translation. To do this, you would run makettyxlate with a name for your translation table file. Let's call the file German.out.

```
makettyxlate German.out
```

Since German.out is not an existing file, the makettyxlate program would set up a table with all the characters in their standard representation. You will then edit the table to specify how any characters are to be translated. Figure 3-4 shows how your screen would appear when you begin editing German.out:

```
   | 00 10 20 30 40 50 60 70 80 90 AO B0 CO DO E0 F0
   |
00| 00 10 20 30 40 50 60 70 80 90 AO B0 CO DO E0 F0
01| 01 11 21 31 41 51 61 71 81 91 A1 B1 C1 D1 E1 F1
02| 02 12 22 32 42 52 62 72 82 92 A2 B2 C2 D2 E2 F2
03| 03 13 23 33 43 53 63 73 83 93 A3 B3 C3 D3 E3 F3
04| 04 14 24 34 44 54 64 74 84 94 A4 B4 C4 D4 E4 F4
05| 05 15 25 35 45 55 65 75 85 95 A5 B5 C5 D5 E5 F5
06| 06 16 26 36 46 56 66 76 86 96 A6 B6 C6 D6 E6 F6
07| 07 17 27 37 47 57 67 77 87 97 A7 B7 C7 D7 E7 F7
08| 08 18 28 38 48 58 68 78 88 98 A8 B8 C8 D8 E8 F8
09| 09 19 29 39 49 59 69 79 89 99 A9 B9 C9 D9 E9 F9
0A| 0A 1A 2A 3A 4A 5A 6A 7A 8A 9A AA BA CA DA EA FA
0B| 0B 1B 2B 3B 4B 5B 6B 7B 8B 9B AB BB CB DB EB FB
0C| 0C 1C 2C 3C 4C 5C 6C 7C 8C 9C AC BC CC DC EC FC
0D| 0D 1D 2D 3D 4D 5D 6D 7D 8D 9D AD BD CD DD ED FD
0E| 0E 1E 2E 3E 4E 5E 6E 7E 8E 9E AE BE CE DE EE FE
0F| 0F 1F 2F 3F 4F 5F 6F 7F 8F 9F AF BF CF DF EF FF

0-F to change, ^F - Right, ^H - Left, ^P - Up, ^N - Down, ^C
- Save and exit
```

Figure 3-4. Original Output Table for "makettyxlate" Program - Standard Character Representation

If you look at Figure 3-3, you can see what characters the German ISO character set contains. Some of these characters replace characters in the U.S. English ASCII set. For the MAI 2000, these replaced characters are in the International character set (figure 3-2). The MAI 2000 only supports the U.S. English ASCII and International character sets.

The hex values for the ASCII characters and for the International characters that must be interchanged to create the German set are:

ASCII	40	5B	5C	5D	7B	7C	7D	7E
Int'l	D9	A2	AE	B3	C2	CE	D3	BF

To translate these characters so that they would appear on your terminal, the values in the International (high-order) character set would be replaced by the values of the characters that they correspond to in the German ISO set. The translation table you are creating with makettyxlate, German.out will be modified to show this. Figure 3-5 shows how the modified output table would appear on your screen:

```
   |
00|10 20 30 40 50 60 70 80 90 AO B0 CO DO E0 F0
   |
00|10 20 30 40 50 60 70 80 90 AO B0 CO DO E0 F0
01|11 21 31 41 51 61 71 81 91 A1 B1 C1 D1 E1 F1
02|12 22 32 42 52 62 72 82 92 5B B2 7B D2 E2 F2
03|13 23 33 43 53 63 73 83 93 A3 5D C3 7D E3 F3
04|14 24 34 44 54 64 74 84 94 A4 B4 C4 D4 E4 F4
05|15 25 35 45 55 65 75 85 95 A5 B5 C5 D5 E5 F5
06|16 26 36 46 56 66 76 86 96 A6 B6 C6 D6 E6 F6
07|17 27 37 47 57 67 77 87 97 A7 B7 C7 D7 E7 F7
08|18 28 38 48 58 68 78 88 98 A8 B8 C8 D8 E8 F8
09|19 29 39 49 59 69 79 89 99 A9 B9 C9 40 E9 F9
OA|1A 2A 3A 4A 5A 6A 7A 8A 9A AA BA CA DA EA FA
0B|1B 2B 3B 4B 5B 6B 7B 8B 9B AB BB CB DB EB FB
0C|1C 2C 3C 4C 5C 6C 7C 8C 9C AC BC CC DC EC FC
0D|1D 2D 3D 4D 5D 6D 7D 8D 9D AD BD CD DD ED FD
0E|1E 2E 3E 4E 5E 6E 7E 8E 9E 5C BE 7C DE EE FE
OF|1F 2F 3F 4F 5F 6F 7F 8F 9F AF 7E CF DF EF FF
```

Figure 3-5. Edited Output Table for "makettyxlate" Program - German ISO Set.

In order to be able to enter characters from your German ISO terminal and have them correctly interpreted by the computer as the desired characters in the International (high-order) character set, you need to set up an input translation table. This is basically the reverse of the output translation table process. You need to map the low-order codes the terminal generates into the high-order codes the computer expects. An example of the input translation table is shown below:

	00	10	20	30	40	50	60	70	80	90	AO	B0	CO	DO	E0	F0
00	00	10	20	30	D9	50	60	70	80	90	AO	B0	CO	DO	E0	F0
01	01	11	21	31	41	51	61	71	81	91	A1	B1	C1	D1	E1	F1
02	02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2
03	03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3
04	04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4
05	05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5
06	06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6
07	07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7
08	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
09	09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9
OA	OA	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA
0B	0B	1B	2B	3B	4B	A2	6B	C2	8B	9B	AB	BB	CB	DB	EB	FB
0C	OC	1C	2C	3C	4C	AE	6C	CE	8C	9C	AC	BC	CC	DC	EC	FC
0D	OD	1D	2D	3D	4D	B3	6D	D3	8D	9D	AD	BD	CD	DD	ED	FD
0E	OE	1E	2E	3E	4E	5E	6E	BF	8E	9E	AE	BE	CE	DE	EE	FE
OF	OF	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF

Figure 3-6. Edited Input Table for "makettyxlate" Program

INTRODUCTION

"Login" is a security feature that allows an MAI 2000 system to be set up so that only authorized people can use it. You can eliminate the operator login procedure if it does not suit your needs. You can define your system so that when it is booted, the terminals start up in the command interpreter, BASIC, or an application program. You can also set up your system so that some terminals require login while others startup without requiring login.

If you don't use "login" on your system, you will lose the effects of some security features. The file and directory access modes are not effective, menu security is not active, and ownership of files, directories, and tasks cannot be identified.

If you want to use the operator login security feature on any terminal, you must define the operators that are allowed to log in.

STARTING THE TERMINALS

The "configure" utility is used to define which terminals require login. It also defines the program that is started on each terminal when the system is booted.

When configuring a terminal, the system must be told information about the terminal itself, as well as what program to start. This latter piece of information is communicated to the system by specifying a "start process". This defines the initial process (program) that is run on that specific port, when the system goes into multi-user mode. Generally, by default, a "login" process is started on each terminal port. This enables you to "log in" and work interactively with the system. You may, however, modify start processes to suit your individual needs. Any program that is to run as a start process must be run via the "exec" command (see "exec" in the Command Language sections of this manual. The "exec" command runs a program in foreground or background mode. It allows the redirection of stdin, stdout, and stderr to files or devices. Here is the default start process for the system console ("/dev/ttyO").

```
exec,t=/dev/tty0,/bin/login,t=0
```

All arguments in the start process command are separated by commas with no added spaces. The "t=/dev/ttyO" is a parameter for exec. "t=" specifies the device that the program being "exec"ed is to run on; in this case, "/dev/ttyO". The "/bin/login" is the name of the program being nexec"ed. The "t=0" is a parameter for "/bin/login". This explains why there are two "t=" parameters: the first is for "exec", and the second is for /bin/login. The "t=" for "/bin/login" specifies the time (in minutes) the user has to logon to the system. "t=0" means no time limit for logging on. If the "t=" is not specified, the time limit defaults to 3 minutes. Here is the default start process for terminals other than the system console:

```
exec,t=/dev/tty<#>,/bin/login

  where <#> is the tty number
```

When n/bin/loginn is executed, it will: prompt for the users account and password, print the message of the day (contained in file "/etc/motd"), execute any commands specified in file ".init" in the user's home directory, and execute the initial program specified in the password file ("/etc/passwd"). The initial program could be the command interpreter, a menu, an application program, BASIC, etc. More information on setting up operators is included later in this chapter.

An example of another startup process is below.

```
exec,t=/dev/ttyn,/bin/basic,s=128,pgm=/application
```

Once again "exec,t=/dev/ttyn" is used. In this example, "/bin/basic" is started after an escape has been entered. "s=128" is the start size (128 pages), "pgm=/application" is the name of the basic program which will be executed. When "/application" ends, "basic" will immediately stop and the "exec" command will be restarted. If you wish to enter console mode you will need to include ",-nr," (no release) in the parameter line, (",bin/basic,-nr,s=128...").

As a final example of a startup process, we present one that will enter the user immediately into a menu. In this case, it will be "/util/menu".

```
exec,t=/dev/ttyn,/util/menu
```

When the system goes into multi-user mode, the first menu in the program will be issued on terminal "/dev/ttyn".

DEFINING OPERATORS

The login process ("/bin/login") requires a user to enter an operator name and password in order to access the system. For systems that use the login process, operators must be defined. If the the login process is not used, there is no need to define operators.

Operator definitions are stored in the file "/etc/passwd". This file will exist even if your system doesn't use the login procedure. It holds the name and password of the system administrator, "admin", operator.

The "oprinfo" utility allows the system administrator to define operators. It sets the parameters for each operator: name, security codes, home directory, ID number, initial program, and password. The "oprinfo" utility is fully described in Utilities section of the MAI 2000 User Guide. Here are some sample operator definitions:

	Operator 1	Operator 2
Name:	John	martha
ID number:	1	2
Home directory:	/usr/john	/usr/martha
Initial program:	menus	/bin/basic,s=128, pgm=/basic/pgm
Initial menu:	ut.cisys	---
Security code:	xyz	---
Password	ford	marty

When John logs onto a terminal, the Intersystem Cartridge Transfer menu ("ut.cisys") is displayed. His working directory is set to "/usr/john". John is allowed to execute any menu or menu option that has a security code of "x", "y", "z", or no security (security code = blank).

When Martha logs in, her working directory is set to "/usr/martha", and the BASIC program "/basic/pgm" is executed. When the program is done, Martha is automatically logged off the system.

The operator security codes control the menus and menu options that the operator is allowed to execute. The security codes control the operator only as long as he is within the menu system. In the above example, Martha has not been assigned any security codes. Because Martha's operator definition has been set up so that she can execute ONLY "/basic/pgm", there is no need to assign security codes to her.

The "cpw" command can change an operator's password. It is described in the Command Language section of this manual.

OPERATOR STARTUP CONTROL FILE

If a command file named ".init" exists in an operator's home directory, it is executed when the operator logs onto the system, but only if the operator logs into the command interpreter.

A separate ".init" file can be defined for each operator who logs into a unique home directory. If two operators log into the same home directory, the same ".init" file will be executed for each.

The ".init" file is executed before the initial program. (The initial program is defined in the operator's parameters.) The "ved" command is used to create and modify the ".init" files.

The global and local abbreviation, or "macro," files, ".globals" and ".locals", are described in the Command Language section. The global file exists in an operator's home directory and are available only for that operator. Macros are described in more detail in the Command Language section of this manual.

INTRODUCTION

The startup procedures control the system's operation while it is being booted. The shutdown procedures are used when the system is being shutdown.

BOSS/IX STARTUP CONTROL PARAMETERS

The "vconf" command allows you to set some of the parameters that control the BOSS/IX operating system, including the startup procedures. The Command Language section of this manual describes how to execute the command. Also, chapter 6 of this section describes some of the other "vconf" parameters.

These are the startup parameters that can be changed by "vconf":

root device (root=)
This parameter specifies the disk partition that contains the root filesystem. (The default is 1403, "/dev/root")

swap device (swap=)
This parameter specifies the fixed disk partition that is used for task swapping. The default device is "/dev/swap".

swap size (size=)
This parameter specifies the size of the swap device in blocks. The size specified must be no larger than the defined size of the device (see "usb" command). The default size is 4096 blocks (1000 hexadecimal).

system ID (system=)
You can set the system name that is displayed when the system is booted.

mount error override (-oride, -noride)
You can specify how the system should handle a filesystem mount error. This error occurs when trying to boot a system that was shutdown without the proper procedures. You can tell the system to ignore the error.

A mount error indicates that the root filesystem, and any other filesystems that were mounted when the system was shutdown, are probably damaged.

single/multi boot (-single, -multi)
You can tell the system to boot into single-user mode or directly into multi-user mode.

boot messages (-silent, -verbose)	You can tell the system to boot silently or to display boot status messages. Verbose mode is sometimes useful when you are attempting to diagnose a problem with booting the system.
secure boot (-secure, -nonsecure)	You can tell the system to require the system administrator ("admin") password in order to boot. This security feature is active only if the system is set up to boot into single-user mode. If the system boots into multi-user mode, the regular password security will be implemented.

AUTOMATIC FILESYSTEM CHECK

When the system is started, it will look for filesystems that were left mounted when the system was shutdown. This condition is usually caused by an improper shutdown procedure or a power failure.

When such a filesystem is found, it will be checked for errors and the errors will be repaired, if possible. The repair process will recover all the data it can. Some data might be lost, but the filesystem will be restored to a usable condition.

The file "/etc/filesystems" can hold a list of the filesystem device names that should be checked and, if necessary, repaired. Filesystems that are not named in this file are not checked or repaired. If "/etc/filesystems" is not found, all filesystems on the disks and diskette will automatically be checked, "/etc/filesystems" can be created by using "ved". It should contain the names of the unbuffered filesystem devices that should be checked. Each name should end with a <RETURN>. For example:

```
/dev/rroot<RETURN>
/dev/rboot<RETURN>
/dev/rother<RETURN>
```

The "/etc/filesystems" file should be created on the root and the boot filesystems.

STARTUP PROCEDURE CONTROL FILES

The "/etc/sinit" command file is executed when the system is booted. It is executed before the system goes into single- or multi-user mode. It can be modified to perform such tasks as mounting filesystems. Your system's initial "sinit" contains a command that requests the system date and time during boot.

The "/etc/startup" command file is executed when the system goes to multi-user mode. You can use it for the same kinds of tasks as the "/etc/sinit" file. It can be used to start processes.

LOGIN MESSAGE FILES

The terminal log on message files contain text that is displayed at various points in the terminal log on procedure.

The "/etc/exem" text file contains the message that is displayed on the terminal screen with the "Press 'CTRL'+'C or 'ESCAPE'..." message when the terminal is started in wait mode.

The "/etc/logm" text file is displayed on the login screen that asks the user to enter his "account name" (operator name) and "password".

The contents of the "/etc/motd" text file is displayed on the screen after the operator logs on.

The "/etc/linit" command file can be created to display messages or execute commands when any user logs onto the system. It is executed following the "/etc/motd" file is displayed.

SHUTDOWN MESSAGE FILE

The "/etc/shtdwn" file is a command file which is executed when the system is being shutdown from single user mode. Commands can be added and deleted from the file by using the text editor, "ved".

OVERVIEW

The "vconf" command allows you to set some of the parameters that control the BOSS/IX operating system. The values of these parameters are based on your system's software and hardware components. Following the values in table 6-1 will prevent degradation of the system's performance.

Chapter 7 of this section describes how to calculate your system's memory requirements. Some of the "vconf" parameters described in this chapter affect the amount of memory your system needs.

Only parameters affecting system performance are described here. For the full list of "vconf" parameters, refer to the description of the "vconf" command in the Command Lanugage section.

CONTROL PARAMETERS

The BOSS/IX control parameters that can be changed by the "vconf" command are listed below. Recommended values for the parameters are given in Table 6-1.

crash dump (-dump, -nodump)
You can tell the system to generate a dump when the system crashes. The dump might show the cause of the crash.

linekill character (linekill=)
You can set the keyboard character for the linekill function. The default is the <DELETE> key.

erase character (erase=)
You can set the keyboard character for the erase function. The default is the <BACKSPACE> key.

mountable filesystems (mfsys=)
You can specify the maximum number of filesystems that can be mounted on the root filesystem. You will need to increase this number if you define additional filesystems. You will also need to increase this parameter if you add a diskette drive.

printers (printers=)
You can specify the maximum number of printers that can be configured on your system.

debug (-debug, -nodebug)
You can specify whether or not the system will enter the O.S. debugger on a system crash.

processes (procs=)	You can specify the maximum number of processes that can execute on the system. You might need to increase this number if you frequently run out of processes, have many terminals, use LAN or ghost tasks, or execute several applications simultaneously. If you change the number of processes, you should also change these parameters: open files event calls shared text segments logical units file locks
file locks (locks=)	You can specify the maximum number of files that can be locked at the same time.
LAN sockets (sockets-)	You can set the number of well-known LAN sockets on your system.
dynamic LAN sockets (dsockets)	You can set the number of dynamic LAN sockets on your system.
LAN buffers (lanbuffers=)	You can specify the number of LAN buffers on your system.
system buffers (buffers=)	You can specify the number of BOSS/IX buffers. The buffers are defined in memory. They are used to cache disk sectors. The number of buffers affects system performance. If you use a lot of keyed files, you may be able to improve performance by increasing the number of buffers.
open files (opens=)	You can specify the maximum number of files that can be open at one time.
event calls (eventcalls=)	You can specify the maximum number of event calls that can be opened a time.

shared text segments (tsegs=)	You can set the maximum number of shared text segments (program code segments) on your system.
logical units (lus=)	You can set the maximum number of logical units (lu's) on your system.
input buffer size (ibsize=)	You can specify the maximum number of characters which may be transferred on a terminal read.
type ahead buffer size (tbsize=)	You can specify the maximum number of characters which may be queued for a future terminal read.
RAM disk size (ram=)	The RAM disk is a feature that is used by the software installation procedures. It cannot be changed.

RECOMMENDED "vconf" VALUES

The following table shows the recommended values for the BOSS/IX "vconf" parameters.

Table 6-1. Recommended Values for BOSS/IX Operating Parameters

SYSTEM DESCRIPTION -----> / "vconf" BOSS/IX PARAMETERS V	base system: 2 user 1 printer no ghosts	add for each extra user (above 2)	add for each ghost task	add for each extra printer (above 1)	maximum system config- uration
buffers=	16	4			64
printers=	1			1	14
procs=	23	6	1	1	80
tsegs=	16	4			64
sockets=	6				6
dsocket=	8				36
lanbuffers=	8				16
locks=	64	5	5		160
ibsize=	512				2048
tbsize=	64				512
mfsys=	3				*
lus=	96	20	20	6	256
opens=	40	8	10	2	206
eventcalls=	64	5	5		152

Allow one mountable filesystem slot ("mfsys") for each mountable disk partition. The default of three allows for a boot partition, a root filesystem partition, and a diskette filesystem.

NOTE

The BOSS/IX configuration parameters required for other products are described in their documentation. Specifically, the LAN and 27XX/37XX communications products will require changes to the system parameters.

INTRODUCTION

This chapter describes how to determine the amount of main memory (RAM) needed for an MAI 2000 system.

The system memory requirements are based on the hardware options configured on the system (such as number of terminals and printers) and software packages (such as LAN, communications packages, application programs, etc.) that will be used.

The system can be operated with less than the optimum amount of memory. However, the system will run slower because more swapping of tasks is necessary.

APPROXIMATING SYSTEM MEMORY REQUIREMENT

This section describes how to approximate the amount of memory the system needs. For most systems, you will be able to run application tasks with little or no swapping if you follow the guidelines in this section.

Table 7-1 shows four standard configurations. The number of users (terminals), ghost tasks, and printers is different for each system. You can use this table to approximate the amount of memory that your system needs. Find the standard configuration that most closely matches the system that you want. The table will then show you how many memory boards such a system needs.

The figures in table 7-1 assume that you will be using your system to execute 60% BASIC and BASIC applications programs, 40% non-BASIC applications. If you execute a higher percentage of non-BASIC programs, you will need more memory.

Table 7-2 shows how to set up the "vconf" BOSS/IX parameters for each of the four standard configurations. If you determine your system's memory needs from table 7-1, you should set your "vconf" parameters according to table 7-2.

The approximate method is a good way to determine your system's memory needs if your configuration is close to one of the standard configurations. If it is not, you should use the method of calculating memory requirements that follows.

Table 7-1. Approximating the Memory Requirements for an MAI 2000 System

Standard Configuration	Users/ Terminals	Ghost Tasks	Printers	Memory Needed	Memory Boards
number 1		0	1	.75 MB	3
number 2	5	1	2	1.00 MB	4
number 3	9	2	2	1.25 MB	5
number 4	12	3	3	1.50 MB	6

NOTE: 1 MB (Megabyte) = 1024 KB (Kilobyte) = 1024 * 1024 bytes

Table 7-2. "vconf" BOSS/IX Parameters for Standard MAI 2000 Configurations

"vconf" BOSS/IX Parameter	Standard Configuration (see Table 10A):			
	number 1	number 2	number 3	number 4
buffers=	16	28	44	56
printers=	1	2	2	3
procs=	23	45	72	94
tsegs=	16	28	44	56
sockets=	6	6	6	6
dsocket=	8	14	20	26
lanbuffers=	8	12	16	20
locks=	64	84	109	129
ibsize=	512	512	512	512
tbsize=	64	64	64	64
mfsys=	3	3	3	3
lus=	96	184	286	347
opens=	40	77	120	157
eventcalls:	64	85	111	132

NOTE: The "standard" configurations are described in Table 7-1.

CALCULATING SYSTEM MEMORY REQUIREMENT

This section describes how to calculate the number of memory boards that your system requires for adequate performance. The calculations are based on your hardware configuration and how the system will be used.

Step 1

To compute the memory required, add the following numbers:

NOTE: 1 KB = 1024 bytes

220 KB — This is the memory required for the BOSS/IX operating system program.

39 KB* — This is the memory required by BOSS/IX for table space for the base configuration: two users (terminals), one printer, and no ghost tasks. You will need to allow more memory for table space for each additional user as described later in this formula.

51 KB* — This is the memory requirement for the operating system's dynamic work space (or unswappable memory).

7 KB* — Add 7 KB for BOSS/IX table space for each additional user above the two that are allocated in the base system.

7 KB* — Add 7 KB for BOSS/IX table space for each ghost task.

1 KB* — Add 1 KB for BOSS/IX table space for each additional printer above the one that is included in the base system.

1 KB* — Add 1 KB for dynamic work space for each additional process above the 23 that are included in the base system.

30 KB — Add 30 KB for each additional printer (above one) that you want to print concurrently. This is a memory allowance for a despooler task.

100 KB — This is the memory allowance for system tasks, including the print spooler, the print despooler, the system error log task, the system start process, the command interpreter, etc. These processes can be swapped with each other without affecting the application processes.

---

* The four items marked with an "*" are the memory required for BOSS/IX operating system table space and dynamic work space. These numbers are the allowance for the table space memory assuming that you set the "vconf" parameters based on the figures shown in table 6-1. If you set the parameters differently, you should use the exact method of calculating the table space memory that is described in the following section and table 7-3.)

160 KB — Add 160 KB for each unique, non-BASIC task that will be executed concurrently on your system. Such tasks include the BASIC interpreter (you need one BASIC interpreter task to support any number of BASIC programs), the utilities, LAN, Informix, communications, the command interpreter, etc. When computing the memory required for application tasks, you should consider how often each application is used. For example, if you seldom execute the commands or the utilities, you do not need to allocate memory for them. When they are executed, system performance will slow, but this is probably acceptable.

15 KB — Add 15 KB for each user (terminal) that will be executing tasks concurrently. For example, if you have ten operators who are allowed to login but your system only has five terminals, you will have a maximum of five users at any one time. If one of your terminals is seldom used, you don't need to allocate 15 KB for an operator on that terminal.

64 KB — Add 64 KB for each ghost or background task that will be executed concurrently.

60 KB — Add 60 KB for each BASIC program. This space is the allowance for the data and stack segments for each BASIC program. Note: all BASIC programs share the same text (code) segment. Furthermore, this value is not exact, for some applications may require more memory and some may require less memory.

The resulting number is your system's "memory requirement".

Step 2 — Decide how much swapping of application programs you can allow on your system. Swapping slows the speed of execution.

If you want to run without swapping, you must provide the entire memory requirement that you computed in step 1.

If the cost of memory is more important than speed, multiply your system's memory requirement by a number between 0.7 and 1.0. Multiplying by 1.0 yields a system with the normal amount of swapping; multiplying by 7.0 yields a system with more frequent swapping.

DO NOT run a system with less than 70% of the memory calculated in step 1. This will cause an excessive amount of swapping.

The result of this calculation is your systems "memory need".

Step 3

You can now compute the number of memory boards that your system needs by dividing the memory needed by 256 KB (memory needed / 256 KB). 256 KB is the amount of memory on a board. If a fraction results, you should round the result to the next highest whole number. (However, if your calculation shows that you need 3.1 boards, you can probably use 3 boards without performance degradation.)

A maximum of six memory boards can be installed in an MAI 2000 system. If your figures show that you need more than six memory boards, even with the 0.7 factor that allows some swapping of application tasks, you are overloading your system!

SAMPLE MEMORY CALCULATION

This example shows how to compute the memory requirement for a system with 5 terminals and two printers. Four of the terminals are dedicated to running BASIC applications: accounts receivable, accounts payable, payroll, and word processing. The fifth terminal is used to run the utilities, the commands, or the Informix database application. One of the applications uses a ghost task to output reports to a printer.

Step 1:

Add these numbers to compute "memory required"

220	KB	Base operating system
39	KB	Operating system starting table space
51	KB	O.S. dynamic work space
7	KB x 3	Allowance for table space for three additional users (total of 5 users/terminals)
7	KB	Allowance for table space for one ghost task
1	KB	Allowance for table space for an additional printer (total of 2 printers)
30	KB	Allowance for concurrent printing to the second printer.
100	KB	Swappable system tasks
160	KB x 2	Allowance for the BASIC interpreter and for the utilities/commands/Informix
15	KB x 5	Allowance for five operators executing concurrently
60	KB	One BASIC program
0	KB	No memory is allocated for ghost tasks

TOTAL = 924 KB = memory required

Step 2: Compute "memory needed"

If we don't want to allow swapping of application tasks, memory needed is the same as memory required:

Memory needed = 924 KB.

If frequent spapping of processes is acceptable:

Memory needed = 924 KB » 0.7 = 644.7 KB

Step 3: Compute the number of memory boards required

No swapping: 924 KB / 256 KB = 3.6 boards

Rounded to the next highest whole number, this means that the example system needs four memory boards. Four boards allows extra memory that can be used for additional applications.

Swapping allowed: 644.7 KB / 256 KB = 2.52 boards

The example system needs three memory boards to operate with some swapping of application tasks.

The approximate method of determining a system's memory requirement (Table 7-1) shows that a system with 5 users (terminals), 2 printers, and 1 ghost task needs 4 memory boards. The method described in this section shows that such a system will experience some increase in swapping unless 5 memory boards are used.

CALCULATING TABLE AND DYNAMIC WORK SPACE

The calculation method given above is accurate if the system's configuration parameters are set according to table 6-1. If the BOSS/IX parameters are not set according this table, use the following method to calculate the amount of memory needed for table and dynamic work space.

Table 7-3 shows the amount of memory allocated for each operating system configuration parameter; it shows the parameters for the base system configuration; it shows how the 39 KB value for table space, used above, was claculated; and it shows the parameters and total table space required for the example system depicted above.

To determine the table space for any configuration, multiply the value of each configuration parameter by its corresponding memory allowance (column 1 of table 7-3). Next, add the result of these calculations (column 2). The result is the allowance for table space.

To determine the memory allowance for opeating system dynamic work space (i.e. unswappable memory) use the following formula:

```
dynamic memory (bytes) = (32 + (opens / 4)
                + (((procs * 2688) / 512) / 2)) * 512
```

The values and variables used in the formula are:

32 - the number of 512 byte pages for miscellaneous operating system space

opens - the number of open files configured

procs - the number of processes configured

2688 - the size (bytes) for each process's user area

/ 512 - for converting into 512 byte pages

/ 2 - taking half the number of pages

* 512 - for converting pages into bytes

For the base configureation, the dynamic memory allowance is:

```
dynamic memory = (32 + (40 / 4)
                  + (((23 * 2688) / 512) / 2)) * 512
                  = 52416 bytes = 51 KB
```

TABLE 7-3. BOSS/IX Operating Parameters and Memory Requirements

BOSS/IX PARAMETERS (set by "vconf")	memory used for each param. (bytes)	param.'s for base config- uration system	total memory for base system (bytes)	param.'s for example system ?	total memory for example system (bytes)
buffers=	586	16	9376	28	16408
printers=	228	1	228	2	465
procss	284	23	6332	45	12780
tsegs=	20	16	320	28	560
sockets=	600	6	3600	6	3600
dsocket=	600	8	4800	12	7200
lanbuffers:	400	8	3200	10	4000
locks=	30	64	1920	84	2520
ibsize=	512		1024		2560
tbsize=	64		128		320
mfsys=	424	3	1272	3	1272
lus=	40	96	3840	182	7280
opens=	86	40	24944	76	32648
eventcalls:	20	64	1280	83	1660
TOTALS	--	--	39760 39 KB	--	67161 65 KB

Refer to chapter 6, table 6-1, which explains how the parameter values for the example system were derived.

INTRODUCTION

Menus are terminal screens that list a set of programs or processing options. The user is allowed to pick one of the choices in the menu. That choice is then executed.

Your system is delivered with a set of menus that organize the utilities. You can change or add to this set of menus. For example, if you purchase an application package, you might want to add that application to one of the existing menus. You can also define a new menu for the application.

Menus and their security feature can also be used to control the programs and features that the operators are allowed to execute.

CREATE AND MODIFY MENUS

The "menuedit" utility allows you to define new menus and/or change existing menus. This utility is described in the Utilities section of the MAI 2000 User Guide.

MENU SECURITY

Menu security works with the operator security codes to control the menus and menu choices that each operator is allowed to execute.

For example, in chapter 4 of this section, operators John and Martha were defined. John was assigned the security codes "xyz". Consider the menu below:

```
SAMPLE MENU:

1.  Accounts Payable      (Security code:  p)
2.  Accounts Receivable   (Security code:  r)
3.  Payroll               (Security code:  z)
```

This menu will allow John to execute only the "payroll" choice. He has not been assigned security codes "p" or "r", so he is not allowed to execute either the "accounts payable" or "accounts receivable" choices. Martha was not allowed access to the menu system at all, since she logged on to a BASIC program. If her status is changed to allow access to the menu system, she would have access to all menu items, since she is assigned no security codes.

The "menuedit" utility is used to set and change the security on the menus. The "oprinfo" utility is used to set security for operators. See chapter 4 of this document for more information on operator security.

INTRODUCTION

This section explains how to move the system console to another port.

Every MAI 2000 system must have a terminal that is defined to be the system console. Your system is initially set up with the system console on port 3 (fw0). The console can only be on port 0 (sc0, the first CMB port) or 3 (fw0, the first 4-way port.

If the system configuration is set for debug ("-debug") or verbose start-up ("-verbose"), the system console must be on port 0.

PROCEDURE

This is the step-by-step procedure for moving the system console:

STEP 1: Shutdown the system.

STEP 2: At the prompt "Press <RETURN> key to reboot ('^C' = alt-load, '^S' = self-test)", type <CTRL>+"z" ('^Z'). This will start the boot prom debugger with the prompt "<debug>"

STEP 3: Type "config" at the <debug> prompt.

STEP 4: The "config" menu and <config> prompt will be displayed:

Config Menu

command	description
read	read config NVRAM
display	display configuration read
terminal	change/specify terminal type
boot	change/specify boot device
console	change/specify system console
download	change/specify download port
printer	change/specify printer port
write	write/update Config NVRAM
debug	exit to the debugger
menu	redisplay this menu

<config>

(Refer to Chapter 11 for more information on the "config" program.)

STEP 5: Type the "display" command to display the current system configuration parameters. The display will look something like this:

```
terminal type  : evdt
boot port      : wd0
console port   : sc0 baud:9600 data bits:7 stop
                 bits:1 parity:odd flow:xon
download port  : sd  baud:9600 data bits:7 stop
                 bits:1 parity:odd flow:xon
printer port   : pit0
```

STEP 6: Use the "console" command to specify the new parameters for the system console. Answer the prompts like this:

```
console device:  fw  (for ports on the four-way
                      controllers, ports 3-14)
                 sc  (for ports on the CMB, ports 0 and 1)

console unit:  0  (must be 0)

baud rate:     your terminal's baud rate
data bits:     your terminal's data bits value
stop bits:     your terminal's stop bits value
parity:        your terminal's parity
flow control:  your terminal's flow control value
```

STEP 7: Use the "write" command to write the changes to the system NVRAM.

STEP 8: Type <ESCAPE> to end "config" and return to the <debug> prompt.

STEP 9: At the <debug> prompt, type the "reset" command. This will program the terminal controller for the new configuration of the system console.

STEP 10: If you need to move the current system console to its new port, or if you need to attach a terminal to the new system console port, do it now.

Type <RETURN> on the new system console terminal keyboard.

STEP 11: The <debug> prompts will now start appearing on the new system console terminal. The old system console terminal will no longer be active.

STEP 12: At the <debug> prompt, type the "boot" command to reboot the system. When you are prompted for the boot device and system file, type <RETURN> for the defaults.

STEP 13: When the system is booted, use the "configure" utility to adjust the terminal configuration to match the new configuration.

INTRODUCTION

You can change the number and size of partitions on the system disk drives, both fixed -disks and diskettes. A partition can hold a filesystem, or it can be used for swapping tasks between memory and disk. This section describes how to partition a disk.

If a second fixed disk is added to your system, its partitions must be defined before it can be used. The same procedure is used.

The "usb" command is used to define disk partitions. It is described in the Command language section of this manual. The process of partitioning also uses other commands and system features.

If you want to change the partitions on a fixed disk, we recommend that you get the assistance of an MAI Basic Four representative. We recommend that you DO NOT change the partitions on the first fixed disk, only on the second. Basic Four software packages depend on certain filesystems of a minimum size being present on the first disk (wdO).

PROCEDURE

Before partitioning a disk, you must backup all files from all filesystems. The partitioning process destroys the existing filesystems.

After backing up the disk, and with the assistance of a Basic Four representative, proceed with the following steps:

1. Read a description of the current partitions
2. Change the partition description file
3. Create/delete partition device definitions
4. Set the new partition description
5. Define the new filesystems for the new partitions
6. Restore the backup files and/or re-install software packages

STEP 0: Backup all files from all filesystems.

STEP 1: Read a Description of the Current Partitions

The first step in the partitioning process is to read a description of the current partitions on the disk.

First, shutdown the system and re-boot with the boot partition as the root filesystem.

As ADMIN, use the "usb" command to read the current partitions from the disk:

```
usb <device> desc=<description file name>
```

where <device> is the full pathname of the unbuffered device file for the whole disk device, such as V dev/rwd1" (the 2nd hard disk), and description file name> is the file where the partition information will be written (defined in the directory "/etc/diskdesc").

For example, the following command will read a description of the partitions on the second fixed disk, wd1, into the file "/etc/diskdesc/oldparts":

```
usb /dev/rwd1 desc=oldparts
```

A description of the disk is displayed while another description is written to file. Note that the two descriptions differ in format.

STEP 2: Change the Partition Description File

The second step of the partitioning process is to change to the partition description to reflect the new disk layout. Use the text editor, "ved", to edit the partition description file that you saved in step 1 ("/etc/diskdesc/oldparts").

This is an example of the partition description file contents as displayed by the text editor:

```
** volume id
BASIC FOUR INFORMATION SYSTEMS DIVISION    M A I 2000
?? class : hds : sects : bytes/sect : # cyls : blocks : rwc: wpc : defects
0:6:17:512:830:84354:400:400:20:
   partition # : starting block : length in blocks
1:0:4096:
2:4096:4096:
3:8192:76162:
```

NOTE

> The lines that start with "**" are comments that describe the following record(s). The last lines of the file describe the starting block number and size of each partition.

The example above shows a disk with three partitions. (Sometimes a "fourth" partition, 0, which is the entire disk, is refered to. This is shown in the display produced when "usb" is run, but not in the "diskdesc" file. Do not include it when you define partitions.) The first and second partitions contain 4 K blocks (2 MB) each, and the third takes the rest of the disk.

You should use "ved" to edit the partition description lines of the file. (In the above example, these are the last three lines of the file.) Change the number and size of the partitions to the values that you want. You can also change the ID, but do not change anything else.

For instance, to add another 2 MB partition after the second partition, add a line and change the last line in the file to this:

```
        .
        .
        .
1:0:4096:
2:4096:4096:
3:8192:4096:
4:12288:72066:
```

STEP 3: Create/Delete Partition Device Definitions

Each disk partition needs associated device file(s). If you are adding new partition(s), you should create the device files for them. If you are deleting existing partition(s), you should delete their device files.

The following table lists the major and minor device numbers that must be used for the partitions:

First fixed disk (wd0):		Major #	Minor #
partition 0:	the complete disk	14	0
partition 1:	first partition	14	1
partition 2:	second partition	14	2
:		:	:
:		:	:
partition 31:	last possible partition	14	31
Second fixed disk (wd1):			
partition 0:	the complete disk	14	32
partition 1:	first partition	14	33
:		:	:
:		:	:
partition 31:	the last possible partition	14	63
First diskette device (fd0):			
partition 0:	the complete floppy	7	0
partition 1:	first partition	7	1
:		:	:
:		:	:
partition 31:	last possible partition	7	31
Second diskette device (fd1):			
partition 0:	the complete floppy	7	32
partition 1:	first partition	7	33
:		:	:
:		:	:
partition 31:	last possible partition	7	63

Each disk should have two device files that refer to the entire disk: the buffered and unbuffered ("raw") versions. For example, the device files "wdl" and "rwdl" are needed for the second fixed disk drive. (The leading "rw on the second device name is a convention that denotes the unbuffered device.)

In addition, two device files (buffered and unbuffered) should be created for each partition if there is more than one partition on the disk (not counting 0, the entire disk, which does not contain a filesystem). These device files may be named as desired. For example, if you create a partition that will be used exclusively for accounting applications, you could name the device files "/dev/acct" (buffered) and V dev/racct" (unbuffered).

Device files for partitions should be created in the directory "/dev" on both the boot and root partitions. (The "makedev" command creates a device file. It is described in Section 3 of the Technical Reference Manual.

If you delete partitions from a disk, the associated device files should be deleted. The "delete" command can be used to delete a device file.

STEP 4: Set the New Partition Description

Next, you should set the new partition definitions that you created in step 2. The "usb" command is used:

```
usb <device> type=<description file name> -set -save
```

NOTE: <device> = full pathname of the unbuffered device file for the whole disk device, such as V dev/rwdl" for the second fixed disk.

description file name> = the file where the partition information is found. It must be in the directory "/etc/diskdesc".

For example, to set the partition description from the example in steps 1 & 2, use this command:

```
usb /dev/rwd1 type=oldparts -set -save
```

STEP 5: Define the New Filesystems for the New Partitions

The next step is to define the filesystems for the partitions that you created or changed.

If you did not change the starting location or the length of a partition, then you do not need to make a filesystem on it. Also, you do not need to make a filesystem on a non-filesystem partition, such as the swap partition.

To make a filesystem, use the "makefs" command. For example:

```
makefs /dev/acct
```

This will make a filesystem on the partition for the device "/dev/acct" from the example in step 3.

After making the new filesystems, you should copy the new device files to any partitions that will be used as the root partitions for an alternate load. You should also delete any obsolete device files from these partitions.

STEP 6: Restore Backup Files and/or Re-install Software Packages

As the last step, you should restore the files that were backed up. If you saved the "/dev" directory, the device files that you deleted will reappear, so re-delete them.

INTRODUCTION

This section describes how you can update the system attributes that are defined in .the NVRAM. For example, you can define the system console's attributes to match the terminal that you are connecting to the system console port.

PROCEDURE

To change or specify system attributes for the NVRAM, you first need to shutdown the system. At the "alt-load" prompt, enter CTRL + Z. The boot prom debugger will be executed. Type "config" to execute the NVRAM Hardware Configurator. The NVRAM configuration menu will be displayed followed by the config prompt:

```
   Config Menu

command   description

read      read Config NVRAM
display   display config
terminal  specify terminal type
boot      specify boot dev
console   specify system console
download  specify download port
printer   specify printer port
write     write config NVRAM
debug     enter Debug
menu      display menu

<config>
```

NOTE: "<config>" is the prompt for the "config" program.

A list of terms:

term	definition
cs	MCS (cartridge streamer)
fd	diskette drive
fw	four-way serial controller
NVRAM	Non Volatile Random Access Memory. This memory is used to store BOSS/IX parameters. It is not part of main memory.
pit	parallel interface and timer
sc	serial communications
wd	Winchester disk (fixed disk)

THE "read" COMMAND

The read command is used to read the current contents of the NVRAM.

THE "display" COMMAND

The display command is used to display the current configuration information. The configuration information is displayed in the following format:

```
terminal type  : evdt
boot port      : wd0
console port   : sc0 baud:9600 data bits:7 stop bits:1
                 parity:odd flow:xon
download port  : sd  baud:9600 data bits:7 stop bits:1
                 parity:odd flow:xon
printer port   : pit0
```

THE "boot" COMMAND

This command can be used to specify/change the boot device. You will be prompted for the Boot device (fd, wd, cs, sc, or fw) and the unit number.

THE "terminal" COMMAND

You can use this command to change the terminal type for the system console. Enter the terminal type (evdt, vdt7270, edt, or other).

THE "console" COMMAND

You will be prompted for the device, unit number, baud rate, number of data bits, number of stop bits, parity option (odd or even), and flow control (xon, none, dtr, xon dtr, modem).

The device may be either "sc" for serial communications (one of the two serial ports on the CMB board), or "fw" for a port on a four-way controller. Only unit number 0 is allowed for both sc and fw.

You must set the console attributes to correspond to the attributes of the system console terminal.

THE "download" COMMAND

The prompts for the "download" command are similar to those for the "console" command. The download port is currently used only for system diagnostics.

THE "printer" COMMAND

The printer can be either the parallel printer or a serial printer. The printer "pit0" is the default (The parallel printer device is "pit" and there is only one unit number: 0). You should enter "pit" to set the printer to the parallel printer.

THE "write" COMMAND

After you are sure that all of your changes are correct, you can use the "write" command to write them to the NVRAM. You can display your changes with the display command.

A NORMAL "config" SESSION

Each time you update the NVRAM you should begin by reading the current contents of the NVRAM (the "read" command). You should then display the contents (the "display" command). After determining the characteristics that you want to change, enter one of the change commands ("boot", "terminal", "console", "download", or "printer"). To view the changes that you have made use the "display" command. If you do not wish to save your changes, return to the system debugger (the "debug" command) without writing your changes. If.you do wish to save your changes write your changes (the "write" command) and then return to the system debugger. Once you are in the system debugger enter a "reset" to reset the system, and then type "bo" to boot. The "config" program is described in the Sorbus Service Manual.

MAI(R) 2000 TECHNICAL REFERENCE MANUAL

SECTION 3

COMMAND LANGUAGE

AUGUST, 1985

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

In this we describe the command language. The command language interface to the BOSS/IX operating system consists of two main parts:

1. the language itself, with both its vocabulary and syntax, and

2. the command interpreter, which is a program that interprets the commands you give.

An extensive introduction to using the command language is provided in Section 8 of the MAI 2000 User Guide. Here we provide full information, but in a briefer form.

CONVENTIONS

A few special notations have been adopted to explain the format of commands in this section. These notations are consistent with, but additional to, the conventions adopted for all MAI Basic Four, Inc documentation.

Symbols

The symbols used in the format examples are the following:

{} Options and parameters enclosed in braces (curley brackets) are optional. If the optional parameters are not entered, the system either does not use them or uses default values.

Braces may be nested, one pair occurring inside another. In this case, there are further optional parameters available if you choose to use the main (outside) parameter option.

All options and parameters not appearing in braces are required for the execution of the command.

| A vertical line occuring between parameters indicates that one or the other parameter should be entered, but not both. Usually, this notation occurs within braces, indicating that this parameter option has two or more possible values.

Parameter Abbreviations

There are a few abbreviations used to shorten and clarify the format of commands. The main abbreviations are listed below.

{options}	Often there are too many options for a command to list in the format without losing clarity. When this notation is used, see the list of options in the discussion of the command for the complete list.
{parameter=value}	Often there are too many parameters for a command to list in the format without losing clarity. When this notation is used, see the list of parameters in the discussion of the command for the complete list.
file1...filen dirl...dirn arg1...argn	This notation indicates a list of file or directory names, or of some other arguments. The list of files, directories or arguments may be any length greater than or equal to one. If the list is enclosed within braces, the entire list is optional.

THE COMMAND INTERPRETER

The command interpreter is a program that mediates the communication between you and the operating system. It interprets commands you enter on the terminal keyboard, and then directs the O.S. to execute the command.

While it is working, the command interpreter monitors your terminal keyboard. When you enter a command the command interpreter reads the input, interprets the input checking for syntactical correctness, and then passes the instructions on to the operating system for processing.

When the instructions given by the command interpreter are completed (or immediately if the process is run in the background), the command interpreter displays its prompt again, indicating that it is ready for another command.

Some processes executed by the command interpreter take an indefinite amount of time to complete. This is the case, for instance, when the BASIC interpreter is the process evoked. In this case, the command interpreter interprets the command "basic" and requests that BASIC command mode be started on your terminal. It then waits until you are finished with BASIC. The command interpreter prompt isn't displayed again until you are finished with BASIC.

HOW TO COMMUNICATE WITH THE COMMAND INTERPRETER

You communicate with the command interpreter by entering commands when the command interpreter prompt is displayed. The command interpreter prompt is usually made up of your user name followed by ">". For example:

```
john>
```

A special case is when you are logged on as the system administrator, in which case the prompt is "ADMIN>". Some commands, or modes of comma:ds, can only be executed by the system administrator.

The prompt may be modified by changing the user "environment," discussed below.

The command you enter consists of the name of an executable file, followed by any other information (arguments) needed to execute the file and other special symbols recognized by the interpreter. All BOSS/IX commands are executable files, as are the utility programs and several special applications programs.

To enter a command, type the name of the command and any additional information required, and then press RETURN. The command and arguments you type is called the "command line". RETURN is the command line "terminator".

While you are typing in the information, it is stored in a buffer. Only when you have pressed RETURN does the command interpreter read what you have typed and attempt to execute it.

RULES

There are several rules that must be followed when you enter a command for processing by the command interpreter.

Case

The BOSS/IX command interpreter is "case sensitive", which means that it recognizes upper- and lower-case alphabetic characters as distinct. All BOSS/IX command names are spelled in lower case letters only, and must be entered in lower case letters only.

Arguments

On a command line, there is usually a list of "arguments" following the command. The arguments may be processing options specific to the particular command, parameters, or files to be processed.

The options arguments always begin with a dash, "-", e.g., "-1". The dash is required by the command interpreter, or else it will mistake the argument for a file name. File names are not preceded by a dash.

The parameter arguments begin with an expression and an equal sign, followed by the value for the parameters, e.g., "t=15". This is used to supply special information to the command interpreter for execution of the command.

Delimiters

If a command line contains arguments, the arguments must be separated from the command and from each other by a space. The space serves as a delimiter, telling the command interpreter where the command or argument ends. More spaces are allowed, but one is required.

Terminators

When you have finished typing in a command, you must press RETURN. This terminates the command entry and tells the command interpreter that you are sending a command. Without this terminator, the command interpreter does not read what you have typed, but waits for you to enter a command.

CONTROL CHARACTERS

Control characters affect what is happening at the terminal. They are transmitted when the control key, CTRL, and some other key are pressed in a single keystroke.

Control characters are used either to correct typing mistakes, to stop and resume output or to stop and suspend a running process.

|CTRL| + |C| This command "kills", or unconditionally ends, the command being run. The command interpreter prompt is then displayed.

|CTRL| + |D| This command indicates either the end of the line or the end of the file, depending on the context.

If anything has been typed on the line prior to the command, it means "End-Of-Line". That line is sent to the system, and the cursor moves down to the next line on the screen. The command interpreter prompt is not displayed at this time.

When the command is typed at the beginning of a line, it means "End-Of-File". If there have been lines of data entered since the command interpreter prompt was last displayed, it signals the end of standard input (data input from the keyboard). If typed alone at the command interpreter prompt, it ends your session with the command interpreter.

|CTRL| + |Q| The command cancels the effect of the CTRL + S command (below), continuing the screen display.

|CTRL| + |U| This command redisplays the current input line. It is useful when input and output are mixed on the screen.

|CTRL| + |S| This command suspends output to the terminal, freezing the display information. If a program is generating output rapidly, this comamnd can be used to stop it while you examine a screen display.

|CTRL| + |Y| This command suspends execution of a command. The execution can be resumed using the "resume" command.

PROTECTING ARGUMENT FIELDS

There are a few symbols that are recognized by the command interpreter as having a special meaning. These are discussed below, but a note is needed here on how to use these symbols without their special sense.

Two of the special characters are "<" and ">", which are recognized by the command interpreter as commands to redirect input and output. When one of these, or any of the other special characters, occurs on a command line, it is processed with its special meaning.

To prevent the special symbols from being interpreted, enclose the message in either single or double quotation marks. This indicates that all enclosed symbols are to be treated "literally," and not interpreted.

In general, whenever an argument in a command line contains any special characters that you do not want interpreted, enclose that entire argument in single or double quotation marks.

COMMAND SYNTAX

For most commands, the characters "-" and "=" indicate an argument. In most cases, the order is unimportant. However, there are exceptions.

- Since the "write" command accepts an entire line, these characters are not interpreted within the line;
- Because "command" expects a command argument, all "-" and "=" arguments must immediately follow the command in order to be interpreted correctly.

If you want to refer to a dash or an equal sign literally, rather than as a flag, precede the argument with double dashes. For instance, if you have files named "-filename" and "filename=temp" that you want to delete, you need to use the double dashes:

```
@> delete -- -filename
@> delete -- filename=temp
```

Without the double dashes, you would receive an error message.

COMMAND FORMAT

There are several different formats used for entering commands. These are described in the following paragraphs. See chapter 5 of this section for the detailed descriptions of the formats applicable to each BOSS/IX o.s. command. Detailed format descriptions for other executable files can be found in the discussion of those files.

Command with No Arguments

Some commands do not require arguments. For some of these, arguments are optional. If arguments are optional, certain argument options are taken as defaults.

Format

```
@> command-name
```

Examples

1)

```
@> date
```

This command displays the current date and time. In this example there are no arguments used with the command.

2)

```
@> ls
```

This command displays the contents of a directory. Without a directory name as an argument, your current working directory is assumed as the default.

Command with Dash Arguments

Most commands have a set of optional arguments that must be preceded with a dash, "-". They have an effect on the details of how the command is executed, how much information is returned, and so on. There must be a space separating each argument from the command or previous argument.

Format

```
@>command -argumentl ... -argumentn
```

Example

1)

```
@>ls -a -l -rev
```

This command lists all ("-a") of the files in your working directory, including files whose names begin with a period, which are normally hidden. The display is in "long" form ("-1"), which includes the file attributes. Finally, the list is displayed in reverse order ("-rev").

Command with Equal Arguments

Some commands have either mandatory or optional arguments that must be entered following a term and equal sign (e.g. "file= "). No spaces should occur between the equal sign and the argument. A space must separate each argument expression from the command and previous argument expressions.

Format

```
@>command argument=argument
```

Example

1)

```
@>date time=043000pm
```

This command sets the time to 4:30:00 pm.

Command with Arguments

Many commands take optional arguments that are not preceded by a dash or equal sign. Host commonly these arguments are file names, though there are several possibilities.

Format

```
@>command argument1 ... argumentn
```

Examples

1)

```
@>ls /usr /usr/fred
```

This command lists the files and directories contained in the directories "/usr" and "/usr/fredn.

2)

```
@>write fred good morning
```

This command prints the message "good morning" on the terminal screen being used by Fred, if Fred is logged on the system.

Command with Combined Arguments

Most commands accept arguments of more than one kind. These arguments can all be combined in a single command line. Each argument must be separated from the command or the previous argument by a space. For most commands, the order of arguments following the command is unimportant. Options and parameters may come in any order, and may precede and follow file names. However, the command must occur first.

Examples

1)

```
@>ls -l /usr
```

or

2)

```
@>ls /usr -l
```

This command lists, in long form, the files and directories contained in the directory "/usr". The order of the arguments does not matter.

3)

```
@>write fred good morning
```

This command prints the message "good morning" on the terminal screen being used by Fred, if Fred is logged on the system. The name of the recipient must precede the message.

PATTERN MATCHING

The command interpreter recognizes a large variety of pattern matching characters. The pattern matching characters can stand for single characters, groups of characters, or the placement of a pattern on a line.

The pattern matching characters available are summarized in Table 1-1. There is a wide variety of pattern matching characters, so you can create nearly any pattern you need.

Table 1-1. Pattern Matching Characters

Symbol	Pattern Matching Function
Asterisk (?)	Matches any number of characters in this position. E.g. "a*z" matches "abz","abcdz", etc.
Question Mark (?)	Matches any single character in this position. E.g. "a?b" matches "axbn, "acb", etc., but not "axxb" or naxcb".
Pound Sign plus Expression (#XYZ)	Matches any number of occurrences of the character following the pound sign. E.g. "a#zb" matches "azb", "azzzb", etc.
Exclamation Point (!)	Allows alternative matching patterns. E.g. nab!ac!de" matches Mabn and "ac" and "de".
Braces or Curley Brackets ({})	Used to group pattern matches by precedence. Brackets can be nested. Patterns inside brackets have lower precedence than patterns outside brackets.
Square Brackets plus Dash ([-])	Used to specify a range of characters. E.g. "a[c-h]b" matches "acb", "adb", "aeb", ... , "ahb". The range is in ASCII order.
Caret plus character in brackets ([^C])	When followed by a pattern and enclosed in square brackets, the up arrow means "not". Anything not matching the pattern is a match. E.g. "a[^b]c" matches "aac", "acc", "a=c", etc., but not "abc".
Double Quotation Marks (")	If the pattern includes more than a single word or includes a space, enclose the entire pattern in double quotes.
Caret (^)	Matches the beginning of a line. It may be followed by a string to find it at the beginning of a line. This is used primarily in the text editor, "ved".
Dollar Sign ($)	Matches the end of a line. It may be preceded by a string to find it at the end of a line. This is used primarily in the text editor, "ved".
Backslash (\)	Used to specify that a pattern matching character or an unprintable character is to be matched.

PROCESS MANAGEMENT

The command interpreter allows you to have extensive control over the processes you are running. You can control where input comes from and where output goes. You can chain processes together to form complex commands. You can also run processes that require no interaction with the terminal as a background process.

INPUT/OUTPUT REDIRECTION

Most commands read data from standard input (the keyboard), write to standard output and standard error (the screen), and display error messages to standard error (the screen), unless otherwise directed. Input can be read from, and output and error messages written to, any input or output device to which you have access. This input and output redirection is accomplished with the commands in Table 1-2. Any time redirection is not specified, standard input and output is used.

Table 1-2. Input/Output Redirection Symbols

SYMBOL	DESCRIPTION
<	Redirects input from specified device or file,
>	Redirects output to the specified device or file, Writes over existing data.
>>	Redirects output to the specified device or file, Appends new output to the existing data.
%	Redirects error messages to the specified device or file. Writes over existing data.
%%	Redirects error messages to the specified device or file. Appends new output to existing data.

The format for using these redirection symbols is:

```
source_process  redirection_symbol  destination
```

For example, you can write the directory listing of the root directory to a file with this command line:

```
@>ls / > /file/name
```

The file is created, if it does not already exist, and the contents of the root directory are written to it. If the file already exists and you want the information appended to the already existing data, use the "» B redirection symbol.

PIPES

Pipes provide a special kind of redirection, allowing you to string several commands together, taking the output from one command and passing it on to the next as input. For instance, this command line lists out the contents of "/bin" one screen at a time:

```
@>ls /bin | p
```

The output from listing "/bin", instead of being directly displayed on the terminal screen, is piped to the command "p". It is then processed by "p", which paginates the listing as it is displayed on the screen. Without piping the listing through "p", the listing runs by too quickly to be read.

The pipe can be continued through as many processes as you want.

BACKGROUND PROCESSING

The command interpreter usually waits for completion of the command before it prompts you for another command. In the case of commands that take a long time to process, especially when no user interaction is required, this is inconvenient. To eliminate this inconvenience, the command interpreter can process commands in the "background."

When you execute a command in background, the command interpreter starts the process, echoes the process identification number (pid), and immediately repeats the prompt. You may then continue processing other commands. The background process continues to operate without interaction from your terminal. It is assigned a lower priority number, so may take slightly longer to run, but you are free to do other things.

To execute a process in the background, enter the command line as usual, and type an ampersand, "&" at the end of the line before press RETURN.

When execution is complete, the background process terminates as usual. Completion is not indicated unless as the last step of the process itself.

COMMAND FILES

The command interpreter can itself be used as a command. What this means is that you can call upon the command interpreter to execute a series of commands contained in a file, called a "command file". A command file takes the place of entering one command after another at the command interpreter prompt.

MAKING COMMAND FILES

Command files can be created using the text editor, "ved", just like any other text file.

Each line of the command file must be an executable command line.

EXECUTING COMMAND FILES

There are two basic methods available for executing command files. These are explained in the following paragraphs.

As a Non-executable File

When you create a command file using ved, the access modes usually include read and write access only. The file cannot be executed directly, but you can call on the command interpreter to execute it for you. To execute the command file, you only need to enter the command line:

```
@>command filename
```

The command "command" invokes the command interpreter for execution of a command file. The command interpreter opens the file, and then executes each of the commands in sequence.

As an Executable File

You can also execute a command file without invoking the command interpreter by "command", if you change the file access rights to include execution privileges. To do this, use the "filemodes" command:

```
@>filemodes +x +x filename
```

This adds execution privileges for the file's owner and all other users. The file may be executed just like any other executable file.

VARIABLES IN COMMAND FILES

The command interpreter recognizes a certain notation as representing variables. The command interpreter accepts file names and command parameters as values for these variables when a command is being executed. The acceptable variable names are:

```
$1, $2, $3, . . . ,$n, $r1, $r2, $r3, . . ., $rn
```

The variables $1, $2, $3i ? ? ? take single files, parameters or options as their values. $1 always takes the first file listed, $2 always take the second, $3 the third, and so on.

The variables $r1, $r2, $r3, . . . each take the file names from the number indicated to the end of the list.

INIT FILES

When you log on the system into the command interpreter, the command interpreter looks in your home directory for a command file ".init". If the file exists, the command interpreter attempts to execute it.

Since ".init" is simply a command file, you can design it to perform many of your routine log on procedures.

NOTE

In order for ".init" to execute when you log on, your operator information file must specify "/bin/command" as your initial program. No other initial program executes ".init" automatically.

POUND-SIGN COMMANDS

The command interpreter recognizes several special commands beginning with the pound-sign (#). The commands are summarized in Table 1-3. A thorough description of each command, including command formats, follows the table.

Table 1-3. Pound-sign commands

COMMAND	MEANING
#a	adds a new abbreviation usable anywhere in a command string
#b	adds a new abbreviation usable at the beginning of a command string
#d	deletes an abbreviation
#e	repeats the last executed command
#h	displays pound-sign command help file
#?	displays pound-sign command help file
#l	lists abbreviations
#p	pushes an abbreviation in the stack
#q	exits the command interpreter
#s	sets the default table for # commands
#x	displays and sets the environment
#v	displays and sets verbose mode

FORMATS AND USAGE OF POUND-SIGN COMMANDS

The formats of the pound-sign commands and descriptions of their functions are given in the following paragraphs. First, however, a few points about the commands in general must be noted.

Limitations

The pound-sign commands are all internal to the command interpreter itself, unlike most of the commands discussed in the next two chapters. As a result, they do not have the full functionality of regular commands. The restrictions are:

- pattern matching characters are not used as pattern matchers, but are taken literally
- the "#e" command does not repeat a preceding pound-sign command

Global and Local Abbreviations

Abbreviations are distinguished as "local" or "global."

Global abbreviations are kept in the file ".globals" in your home directory. Whenever you create a global abbreviation, it is stored in this file in your home directory, regardless of your working directory when created. With one exception, global abbreviations are available only to you, and can be used whatever your working directory is at the time. The exception is that, if another user has the same home directory as you do, then you share global abbreviations.

Local abbreviations are kept in the file ".locals" in the directory that was your working directory when you created it. They are available to you only when your working directory is set to the same directory as when they were created. They are also available to any user whose working directory is set to that directory.

Unless specified otherwise, abbreviations usually become global by default. The default can be changed to local with the "#s" command.

Format @>#a{gll} name expression

Description The "#a" command defines an abbreviation an expression that can occur anywhere in a command string. For instance, if you often change your working directory to a directory with a long path name, you can define an abbreviation for that path name.

The options specify either a global ("g") or a local ("l") abbreviation. The default is to global unless it has been changed.

If another abbreviation with the same name already exists, it is replaced. This is true regardless of whether the earlier abbreviation was created using "#a" or "#b".

Examples 1) @>#a jy install/doc/work/july

This command line sets the definition of "jy" as a global abbreviation. The abbreviation can be used to change your working directory to include this as part of the path as follows:

```
@>cd /usr/john/jy/manuals
```

Your working directory is now:

```
"/usr/john/install/doc/work/july/manuals
```

2) @>#al jy install/doc/work/july

This command line sets the same definition, but as a local abbreviation.

Notes "#a" differs from "#b" in that the abbreviation defined by "#a" can occur anywhere in the command string, whereas the abbreviation defined by "#b" can only occur at the beginning of a command string.

Format @>#b{g|l} name expression

Description The "#b" command defines an abbreviation for an expression that can occur at the beginning of a command string. For instance, if you frequently use the local area network system for remote command execution, an abbreviation can greatly reduce the length of the command line.

The options specify either a global (g) or a local (1) abbreviation. The default is to global unless it has been changed.

If another abbreviation with the same name already exists, it is replaced. This is true regardless of whether the earlier abbreviation was created using "#a" or "#b".

Examples 1) @>#b lan /bin/rcx s=ed -c

This command line sets the definition of "lan" as a global abbreviation. Remote command execution can now be achieved simply by entering "lan" followed by a normal command line, such as:

```
@>lan write -all the local area network is active
```

2) @>#bl lan /bin/rcx s=ed -c

This command line sets the same definition as a local abbreviation.

Notes "#b" differs from "#a" in that, whereas abbreviation defined by "#b" can only occur at the beginning of a command string, an abbreviation defined by "#a" can occur anywhere in the command string.

Format @>#d{g|l} name

Description This command deletes the abbreviation specified.

The options specify either a global ("g") or a local ("l") abbreviation. The default is to global unless it has been changed.

If the specified abbreviation is stacked (see "#p") only the top abbreviation in the stack is deleted.

Examples 1) @>#d jy

This command line deletes the global abbreviation wjy" created earlier.

2) @>#dl jy

This command line deletes the local abbreviation "jy" created earlier.

Format @>#e

Description This command repeats the last command you executed. It does not repeat a previous pound-sign command.

Examples 1)
```
@>date
Wed May 9 1984 14:47:54

@>#e
Wed May 9 1984 14:48:03
```

The first command line displays the current time and date. The second command line, "#e", repeats the first.

Format

```
@>#h

or

@>#?
```

Description

This command displays the complete help file for the pound-sign commands. It has the same effect as "#?".

Example

```
@>#h
#a{g|l} name value - add new abbrev
#b{g|l} name value - add new begin abbrev
#d{g|l} name - delete abbrev
#e      re-execute last non-# command
#h      display complete help file
#?      display complete help file
#l{g|l} {name} - list abbrevs
#p{g|l}{a!b} name value - push abbrev definition
#q      exit command interpreter
#s{g|l} set default table for # commands
#x{d}   {name=value} - display {export to} environment
#v      {on|off} - display {set} verbose mode
```

Format @>#l{g|l} {name}

Description This command displays the specified abbreviation. If no abbreviation name is specified, all abbreviations are displayed.

The options specify either a global ("g") or a local ("l") abbreviation. The default is to global unless it has been changed.

Examples 1)
```
@>#l
#b mt mount /dev/fdO /mnt
#a jy doc/work/July
#b now who;date
```

This command line displays all of the global abbreviations,

2)
```
@>#ll now
#b now date;who;ps;pwd
```

This command line displays the local abbreviation "now."

Format @>#p{g|l}{a|b} name expression

Description This command allows you to temporarily redefine your abbreviations. It does this by handling abbreviations in a "stack", as if one abbreviation is stacked on to of another. If you add an abbreviation with the same name as an existing abbreviation using "#p", the stack is pushed down and the new abbreviation temporarily takes its place. The "#a" and "#b" commands would permanently replace the old definition with the new one.

The options specify either a global ("g") or a local ("l") abbreviation. The default is to global unless it has been changed.

The pushed abbreviation can further be specified as a "#a" or a "#b" type abbreviation. The default is to "#a" type.

The "#d" command deletes the top definition in a stack.

Examples 1)
```
@>#1
#b mt mount /dev/fdO /mnt
#a jy doc/work/july

@>#pb mt mount /dev/boot /mnt

@>#1
#b mt mount /dev/boot /mnt
#a jy install/doc/work/july
```

In this example, all abbreviations are listed first, and then the original abbreviation "Ian" is temporarily replaced.

Format @>#q

Description This command exits the command interpreter. It has the same effect as pressing the CTRL and D keys at the same time. When you exit the command interpreter, you return to the process that called it or log off, depending on the context of the command.

Examples 1)

```
>!command

@>#q

 >
```

The command interpreter was called from BASIC. The "#q" command exits the command interpreter, returning to BASIC.

Format @>#s{g|l}

Description This command sets the default value to global or local for pound-sign commands that use the default. The default is to the current default. The default definition lasts until you logoff, at which time it reverts to global.

Examples 1) @>#sl

This command line sets the default to "local."

Format @>#x{d} {NAME{=value}}

Description This command displays and sets environmental parameters. The standard environment parameters are:

- PATH - the directory paths searched by the command interpreter when looking for the initial command to execute
- HOME - your home directory
- TERM - your default terminal type

Changes made to the environment last only for the duration of the log on session. The environment may be changed at logon by including this command in the user's ".init" file.

Examples 1)
```
@>#x
 PATH=:/bin:/util:/usr/bruce
HOME=:/usr/bruce
TERM=:evdt
```

The command alone displays the current values of the environment parameters.

2)
```
@>#x PATH=:/bin:/util:/usr/bruce:/usr/dennis
```

This command line adds another directory as a search path sequence. All previous paths must be repeated or they are deleted from the search path options. Repeat the command of example (1) to verify the change.

3)
```
@>#xd DBPATH
```

This command line deletes the environment parameter line "DBPATH" from the environment.

Notes The three parameters shown in the description are only examples. These three are used in the standard setup of operators on the 2000. Another, optional, parameter that is recognized by the O.S. is "PROMPT=", which defines your command interpreter prompt. Others may also be recognized by applications software, such as "DBPATH=M, which is used by some data base management systems.

Format @>#v {on|off}

Description This command displays the current status of verbose mode (on or off), or sets verbose mode as specified.

With verbose mode set to "on," all commands are displayed on the screen before they are executed. This includes each command in any command files you execute.

Examples 1)
```
@>#v
Verbose off.

@>#v on

@>#v
Verbose on.
```

In this example, the "#vn command is first used alone to display the current status of verbose mode. Verbose mode is then turned on. Finally, "#v" is used alone again to verify the status.

ENVIRONMENTAL TAILORING

Using the information covered thus far in the chapter, you can tailor the way the system appears.

MAKING YOUR OWN COMMANDS

On the preceding pages, two ways of creating your own commands have been discussed. Both command files and abbreviations can be used to do this.

Command Files

To create your own commands using command files, create a command file, using the text editor, "ved". Then change the access privileges to include execution access, using the command "filemodes."

You can execute your custom command by entering the name of the file at the command interpreter prompt. But, the directory path leading to that file must be included in your search path list. Depending on where you keep your customized commands, you may need to change your environment.

Abbreviations

Abbreviations can also be used to create your own commands. Command abbreviations should be made using the w#b" command, since they always occur at the beginning of a command line.

Global abbreviations are more useful for customized commands than are local abbreviations. As global abbreviations, you can make use of your command regardless of what your current working directory is. Local abbreviations only work when you have the correct working directory.

TAILORING YOUR ENVIRONMENT

As noted above, if you keep your command files in a particular directory, that directory must be included in your search path options in order to be executed simply by entering the file name. It may be convenient to keep these command files in a separate directory, in which case you need to add that directory to your search path list.

To change your environment to include specific directories, use the "#x" command as follows:

```
@>#x PATH=:/dir/path/one:/dir/path/two:/dir/path/three:. . .
```

All of the search paths you want included must be listed or they will be lost.

This change in your environment lasts until you log off the system.

INIT FILES

If you frequently make the same change to your environment soon after you log on the system, you should make use of your ".init" file. This file is executed automatically when you logon to the command interpreter.

For instance, if you keep your customized commands in the directory "/usr/john/commands", you will want to include this directory in your search path list as soon as you log on. Your ".init" file should include this line:

```
#x PATH=:/bin:/util:/usr/John:/usr/john/commands
```

When you log on, the command interpreter automatically executes your ".init" file, thus setting your environment to include your command directory.

You can also include other lines, to display the date, a message, the contents of your calendar, or anything else you set up.

CONCLUSION

The features of the command interpreter discussed in this chapter allow you a great deal of flexibility in how you use your 2000 system, and how it appears to you as you use it. The next two chapters cover the rest of the commands available for your use. Work with them, trying all sorts of arrangements, and your appreciation of the system will continue to grow.

INTRODUCTION

This chapter describes the entire set of BOSS/IX commands included in the base command set, the commands included in the EOS software package. The commands are described in alphabetical order. Full information on the format and argument options for each command are included, as well as a description of the command's function.

A summary of the commands is contained in Table 2-1, following which are the command-by-command descriptions.

LOCATION OF COMMANDS

Most of the commands are executable files in the directory "/bin". A few, however, reside within the command interpreter program itself, and so are not listed in any directory listing. These commands are:

ad
cd (chdir, cwd)
do
prompt
remark
resume
setmask
sync
wait

A few more are kept in the "/sys" directory, as noted under the command descriptions.

BOSS/IX COMMANDS

The BOSS/IX commands are listed in Table 2-1. They are grouped by function and listed in alphabetical order. A brief description of what each command does is also given.

Table 2-1. Commands Summary by Function

FUNCTION	COMMAND	DESCRIPTION
DIRECTORY FUNCTIONS		
	ad	adds alternate working directories
	cd	changes working directory
	chown	changes owner of a file or directory
	delete	deletes a directory, a file, or a link to a file
	diskusage	displays the number of blocks used by a directory
	fllemodes	sets access modes for files and directories
	Is	lists contents of a directory
	makedir	creates a directory
	mount	mounts a directory system onto a directory
	move	moves files or directories across directory systems
	pwd	prints working directory name
	unmount	unmounts a mounted directory system from a directory
FILE FUNCTIONS	addname	adds another name to a file
	advance	advances eventcount files
	cat	displays and concatentates files
	change	substitutes for a pattern in files
	chown	changes owner of a file or directory
	copy	make a copy of a file
	debe	moves files or pieces of files
	delete	deletes a directory, a file, or a link to a file
	diff	displays lines differing between files
	dump	displays binary contents of file
	filemodes	sets or changes access modes on files
	makeec	creates an eventcount file
	makesta	creates remote station
	match	finds occurences of a pattern in files
	move	moves or renames a file
	P	paginates text and filters non-printing characters
	pmask	prints the current file creation mode mask
	pr	paginates file
	setmask	sets file access modes mask
	ved	text editor, creates and edits text files
FILESYSTEM FUNCTIONS		
	fscheck	checks and fixes directory system structure
	makefs	defines and initializes a directory system
	mount	mounts a directory system
	move	moves files across directory systems
	space	displays amount of free space in a directory system
	unmount	unmounts a previously mounted directory system

Table 2-1. Commands Summary by Function (Continued)

FUNCTION	COMMAND	DESCRIPTION
PRINTER FUNCTIONS	lpmaint	maintenance program for printer queues
	lpq	displays status of printer queue
	lpr	enqueues files for printing
	lpstat	displays printer status
	pted	maintains printer tables
TERMINAL FUNCTIONS	makettymntbl	makes a terminal mnemonics table
	makettyxlate	makes a terminal translation table
	ttymntbl	sets terminal mnemonic table
	ttymodes	sets or displays terminal modes
	ttyxlate	sets terminal translation table
TAPE FUNCTIONS	mcscompare	compares files on tape
	mcslabel	labels an mcs tape
	mcslist	lists the contents of an mcs tape
	mcsrestore	restores files from mcs
	mcssave	saves files to mcs
SYSTEM FUNCTIONS	date	shows the system date and time
	devfmt	formats a device
	devstatus	reports a device i/o statistics
	errlog	initializes or displays an error log file
	fscheck	checks and fixes directory system structure
	install	installs a software package
	install_key	installs a public key for a software package
	kill	forcibly terminates a process
	login	signs onto the system
	makedev	creates a device file
	makeec	creates an eventcount file
	makefs	initializes a files system
	makesta	creates remote station
	message	allows or prevents writing to terminals
	pmask	prints the current file creation mode mask
	ps	displays process status
	resume	resumes execution of a suspended process
	shutdown	shuts down the system to single user mode
	suspend	suspend execution
	sync	updates disk with information from memory
	sysinfo	displays OS level and configuration
	usb	update super block
	vconf	maintain configuration file
	ved	BOSS/IX screen oriented editor
	who	displays users currently logged in

Table 2-1. Commands Summary by Function (Continued)

FUNCTION	COMMAND	DESCRIPTION
MISC. COMMANDS	admin	become the system administrator
	advance	advances eventcounts
	command	BOSS/IX command interpreter
	do	executes a command
	echo	displays arguments
	exec	executes a program
	prompt	prompts for a character before proceeding
	remark	introduces a non-executable remark line
	wait	waits for process to exit
	write	sends a message to a logged in user

Format @>ad directory1 ... directory8

Description The "ad" command changes the alternate search directories to those directories included in the argument list. The current working directory, along with directory 1 ... directory8 are then searched for a given file name or command.

At least one directory must be included in the argument list, although it may be the "null" directory, signified by "". In this case the directory list is cleared. Not more than eight directories may be included.

Examples 1) @>ad /usr/john /etc

This command line changes the alternate search directories list to include only "/usr/john" and "/etc".

2) @>ad /usr/*

If there are no more than eight directories in "/usr", they are all added to the search path. If there are more than eight directories, or if there is a file that is not a directory, an error message is displayed.

3) @>ad ""

This command line deletes all directories from the alternate directories list.

Notes Pattern matching characters may be used in specifying the directory list. All files matching the pattern must be directories, and they must meet the restrictions mentioned above, or an error is generated.

The "-a" option to the "pwd" command lists all alternate directories.

Format @>addname newname oldname

Description The "addname" command creates a "link" to a file, which is a directory entry referring to a file. The same file can have several links to it. There is no way to distinguish a link to a file from its original entry.

By using addname, a single file can be referred to by several different directory path names without creating multiple copies of the same file.

A link cannot be formed across directory systems. For example, addname cannot form a link between a file in the root directory system and a mounted directory system.

When you delete one of a group of linked names, only the link is deleted. The file itself is not deleted until the last link has been deleted.

When you write to a file with multiple names, all of the linked files reflect the new contents.

Examples 1) @>addname name2 file

This command line results in two different directory path names leading to the same file.

Notes Links should be created only on a temporary basis, for a specific procedure, and then deleted.

Although addname cannot create links across filesystems, it can create links between files in different directories within the same filesystem.

The copy command can be used to make copies of files in a different filesystem. The copy command creates a real copy of the file rather than creating a link. The two files are handled independently by the system.

Host programs that handle multiple files do not recognize linked files. In addition, the commands "change", "ved" and "copy" with the "-r" option, cause files to become unlinked. Changes made to a file using these commands are not made in the linked file(s).

The maximum number of names that any single file can possess is 127. The Is command, when executed with the "-links" option, shows how many names a file has. This is the number in parentheses before the owner of the file.

Format

```
@>admin
Password:
```

Description

The "admin" command changes your access privileges to those of the system administrator. When you execute admin, the command interpreter prompts you for the password,if there is one. You must enter the correct password before you will be given system administrator privileges.

Notes

Certain commands can only be executed by a system administrator. This command makes it more convenient for a user to gain system administrator privileges without going through the full log off and log on process.

When you press at the same time:

|CTRL| + |D|

after using "admin," your own command interpreter prompt is displayed and your usual access privileges are returned.

File Used

/etc/passwd To verify the system administrator's password.

Format @>advance ecnamel ... ecnamen

Description The "advance" command advances the first part of the specified event count files by one and displays the new event count value.

An event count is a special file type that simply keeps track of the number of times it has been advanced. In some circumstances, event counts can be used to perform the function of incrementing loops in programs.

Examples 1) @>advance /usr/test.ec
/usr/test.ec .1:4 .2:0

2) @>advance /usr/test.ec
/usr/test.ec .1:5 .2:0

Each occurrence of the command increases the specified event count by one.

Format @>cat filel ... filen

Description The "cat" command opens each of the specified files and reads all the characters found in each. It then writes the characters to standard output. If several files are specified, the output is the concatenation of the files in the specified sequence.

If "cat" is used with no arguments, it reads from standard input and writes to standard output.

Because "cat", unlike "p", simply reads the characters in the files without interpreting or filtering them, it can be used to merge several files.

The fact that "cat" does not interpret the characters causes it to give erratic displays if used on non-ascii files displayed to standard output.

Examples 1) @>cat file

Displays the contents of "file" on the terminal screen.

2) @>cat filel file2 > file3

Merges filel and file2, creating file3

Notes Avoid constructs such as:

@>cat filel file2 file3 > file3

This command line will cause the disk to be filled to capacity unless it is terminated.

Format

```
@>cd {/directory/path/name}

or

@>cd {partial/path/name}

Also:

@>cwd {/directory/path/name}
@>chdir {/directory/path/name}
```

Description

The "cd" command changes your working directory to the specified directory. You can specify either the full path name or the partial path name relative to your working directory. You must have execute permission to a directory in order to make it your working directory.

If no directory path name is specified, your working directory is changed to your home directory.

All directories in the directory path name must exist. The last name is the path name must be a directory.

Examples

1) `@>cd /usr/john/work`

Whatever your current working directory is, this changes your working directory to "/usr/john/work".

2) `@>cd work`

If your current working directory is "/usr/john," this changes your working directory to a "/usr/john/work." When a partial directory pathname is specified, your current working directory is prefixed to the specified directory.

Notes

".." may be used to designate the parent directory of your current working directory.

Format @>change {-v} pattern substitution {file1 ... filen}

Description

The "change" command substitutes the substitution string for the pattern string in the specified files.

All pattern matching characters are recognized by "change". If a pattern matching character is to be taken literally, it must be preceded by a backslash, "\". When pattern matching characters are used in the pattern string, enclose the entire string in quotation marks.

The ampersand, "&", is used in the substitution string to include what was matched in the substitution. Multiple occurrences are allowed.

The substitutions are made in place. If a file is specified, the changes are made in the file and its name remains the same. No backup (".bak") copy of the file is made.

If no files are specified, change operates on standard input and writes to standard output. Changes are made line by line. Input is ended with CTRL + D.

Change patterns are matched line by line, so a two-line chnage should be divided into two single-line patterns.

Options

-v Verbose. Lines are displayed before they are changed, once per change.

Examples

1) @>change recieve receive *.doc
Corrects the spelling of receive in all ".doc" files.

2) @> change "ax" "&b&"
axy
axbaxy

3) @>change "devspin" -- "dev=spin" chap.1

The dashes cause the equal sign to be interpreted literally.

Format @>chown newowner filel...filen

Description The "chown" command changes the current owner of filel through filen to the specified new owner. Only the system administrator can change the ownership of files.

All pattern matching characters are available for specifying the files.

Examples @>chown admin /bin/ps /bin/cpw

Changes "/bin/ps" and "/bin/cpw" to be owned by the system administrator.

Notes Some commands require ownership by admin to function properly (e.g. login, ps, cpw).

Only the system administrator is allowed to change a file's owner.

Files Used /etc/passwd To look up the user identification number of the new owner.

Format command {-c} {-v} {command/file/name} {argl ... argn}

Description The command "command" invokes the command interpreter. The command interpreter reads and supervises the execution of most user command lines. Inherent in this capacity is its ability to recognize abbreviations, special characters, pipes, i/o redirection and detached processes (see chapter 3 for descriptions of these).

Called as a command, the command interpreter can be used to interpret a command file. The command lines in the command file are executed in sequential order. The arguments following the name are passed as the values required by the file for execution.

Options

-c Indicates that argl is itself a command line. arg2 through argn then serve as arguments for argl. If argl, the command line, is compound, enclose it in either single or double quotation marks, but don't mix them.

-v Verbose mode. Echoes each command before it is executed.

Examples

1) @>command /usr/fred/testfile -rev /usr

Suppose /usr/fred/testfile is a command file containing the one command line: Is $1 $2.

This example directs the command interpreter to execute the command file, passing it an option and a directory name as arguments for the variables $1 and $2. The execution is exactly as if this command were entered:

```
@>ls -rev /usr
```

2) @>command -c "Is $1 $2" -rev /usr

This example executes exactly as does the previous example. Instead of executing a command file, the -c option indicates that the first argument is a command line. The command line is complex, so it is enclosed in quotes. The command line contains variables, and the following arguments supply the values.

Files Used

/etc/passwd	To determine the user's home directory.
.globals	For the user's defined global abbreviations.
.locals	In the current directory for local abbreviations.

Format

```
@>copy {options} {/source/file/name /target/directory/name}

@>copy {options} {/source/file/name /target/file/name}

@>copy {options} {/source/file/name1 ... /source/file/namen
        /target/directory/name}
```

Description

The "copy" command makes a copy of the source file(s). The copy can be placed either in the source directory or in another directory. If a single file is being copied, the name of the copy can be changed by giving the full destination file name rather than only the destination directory name. If a new file is not given, the current file name is kept.

If a source file does not exist or cannot be opened, or if the destination directory cannot be opened, an error message is displayed. If the second file already exists, it is overwritten.

Copying directories, devices, event counts and the like are permitted, but are treated differently by the o.s. Copying a directory just creates the target directory. Copying a device just creates a new raw device file. Copying an eventcount file just creates a new eventcount.

To copy multiple source files to a directory, list the source files on the command line followed by the name of the destination directory. The command interpreter verifies that the last named file is a directory before attempting the copy.

V en you copy a file or directory, the owner is changed to your user identification. The exception to this is if you are logged on as the system administrator and use the "-setowner" option. In this case the original owner is kept. In either case, the original user access modes are kept.

Options

-c Contiguous. This option stores the new copy of the file on contiguous blocks.

-i Interactive. This option causes "copy" to prompt:

From:
To:

until you type:

|CTRL| + |C|

anywhere on one of the prompt lines, or type either:

|CTRL| + |D|

or:

|RETURN|

alone on one of the prompt lines.

-nc Non-contiguous. This options explicitly allows the new copy of the file to be stored on non-contiguous blocks.

-o Over. This option is used only with the n-r" option. If the directory being copied already exists in the new location, a new copy of the directory is not created. Instead, the contents of the source directory are copied over the contents of the destination directory.

-q Query. Requests confirmation before each file; "Y", "y", "YES", or "yes" to do the copy, anything else to skip the file.

-r Recursive. If the source file is a directory, "-r" creates the target directory and all subdirectories, and copies the files contained in them. Every directory and file from the source directory to the end of the tree structure is copied.

-setowner Setowner. This option copies the files keeping the original owner (only the system administrator can use this option).

-v Verbose. This option reports the result of the copy.

Examples

1) @> copy file-name new-file-name

This command line creates two identical (except for the name) files in the same directory.

2) @> copy file directory

This command line creates a file with the same name in specified directory.

3) @> copy -i
From:

The command line starts "copy" as an interactive process which prompts you for source and destination files.

4) @> copy -r directory newdirectory

This command line creates two identical (except for the name) directories with the same files.

NOTES

When copying a directory using the "-r" option, if the target directory already exists, a subdirectory with the same name as the source directory is created and used as the destination for all the files. If the target directory does not exist, "copy" first creates the target directory and then copies all the files to it.

Format

```
@>cpw {user}
Enter current password:
```

Description

The "cpw" command creates, changes or eliminates your password. Simply type "cpw" and then press RETURN. The program prompts you for your old password and then twice for your new password. Your current and new passwords are not echoed on the screen as they are entered.

The system administrator may also change the password for other users by specifying the user name.

A password may be up to eight characters long.

Examples

1)
```
@> cpw
Enter current password:
Enter new password:
Again:
```

Files Used

/etc/passwd	To check the current password, and then modifies it to contain the new password.

Format

```
@>date {options} {parameter=value}

@>date hhmm{ss} {mm{/}dd{/}yy}
```

Description

The command "date" sets and displays the system date and time. Date and time may be displayed in BOSS/1X format (Day-Name Month-name Day-number Year-number hh:mm:ss) or in BFS format (hh:mm:ss MM/DD/TT).

The command "date" can also be used to change the time and date display formats.

Examples

1)
```
@> date
Fri Mar 23 1984  12:32:41
```

This command line, including no arguments, displays the current system date and time in BOSS/IX format.

2)
```
@>date -bfs
03/23/84 04:41:59 PM
```

This command line displays the current system date and time in BFS format.

3)
```
@>date -mdy time=0200 date=03/23/84 a=AM
Date format is 'mm/dd/yy'. Time format is 'hh:mm:ss AM
or PM (12 hour clock)'
Current date and time:  02:00:00 AM 03/23/84
```

This command line sets the date format to "mm/dd/yy" and sets the system date and time.

4)
```
@>date -i
02:42:34 PM, 03/23/84.  Update clock: hhmmssxx mmddyy
```

Options and Parameters

time=hhmm{ss}{xx}	Set time in hour, minute, second, AM/PM format. Seconds and AM/PM indicators are optional.
date=xx{/}xx{/}xx	Set date in current BFS format
-i	Interactive. You are prompted for the date and time.

The remaining options only apply to BFS date and time formats.

-mdy	Changes to month/day/year date format.
-dmy	Changes to day/month/year date format.
-ymd	Changes to year/month/day date format.
-24	Changes to 24-hour clock format.
-12	Changes to 12-hour clock format.
-sec	Displays seconds.
-nosec	Does not display seconds.
b=xx	Two characters to indicate evening.
a=xx	Two characters to indicate morning.
-default	Sets date and time formats to the system defaults.
-bfs	Prints date and time in BFS format.
-fmt	Displays the current date/time format.

Notes

Only the system administrator can change the time, date, or formats.

Format @>debe {options} {parameter=value{b|k}}

Description

The command "debe" copies the specified input to the specified output, performing the requested conversions. The standard input (terminal keyboard) and standard output (terminal screen) are used by default.

When numbers are specified, they are assumed to specify the number of bytes in the parameters "iskips", "oskip=", "ibs=", "obs=", "bs=n, "ivsizes", "ovsizes", "ivskips", novskip=", "skip=" and "counts". However, "b", meaning "blocks" (512 bytes), and "k" meaning "1024 bytes", can also be used immediately following the value (e.g. oskip=10b bs=1k).

The "debe" command is one of the most powerful commands available, and consequently also has great destructive power. Extreme caution should be exercised when using "debe" any time it is being used to copy data to a device. In these cases, existing data is not respected.

Volumes are created by specifying their size using the "ovsizes" parameter. This is used when the output must be broken into pieces, as when copying a hard disk partition to floppy diskette. When the input file is in volumes, the "ivsizes" parameter must be used.

Options and Parameters

-fill	Pads every output record with zeros to the requested size (ovsize=#).
-pg	Displays a dot each time a record is written to indicate progress.
-noerror	Continues transfer despite errors. Error messages are displayed on the screen.
-notrunc	Does not truncate an already existing output file, enables copying one file into another.
-swab	Swaps bytes (except for the last byte in odd block sizes and odd transfers due to EOF).
iskip=	Skips the number of bytes of the input file before beginning transfer.
oskip=	Skips the number of bytes of the output file before beginning transfer.
skip=	Combines the functions of "iskip=" and oskip=#.
ivskip=	Skips the number of bytes at the beginning of each input volume. The "iskip=" option still works, but only applies to the first volume, where it adds its value to the value of ivskip.

ovskip= Skips the number of bytes at the beginning of each output volume. The "oskip=" option still works, but only applies to the first volume, where it adds its value to the value of "ovskip."

ivsize= Specifies the maximum length, in bytes, of each input volume. When the end of the volume is reached, "debe" asks for the next volume to be mounted. When you are ready to resume copying, type "yes" and press RETURN.

If you type "no" and press RETURN, it is treated as an end of file condition, and "debe" writes any buffered data to output and exits.

ovsize= Specifies the maximum length, in bytes, of each output volume. At the end of each volume "debe" prints a message requesting that the next volume be mounted. When you are ready to proceed, type "y" and press RETURN.

If you type "n" and press RETURN, "debe" stops and displays statistics showing that more data was read than written.

ibs= Input block/buffer size in bytes.

obs= Output block/buffer size in bytes.

bs= Both input/output block sizes in bytes.

if= Input file name.

of= Output file name.

count= Transfers the specified number of input records or until EOF.

Examples 1) @>debe if=/usr/file1 of=/usr/file2

This command line copies "/usr/file1" to "/usr/file2". This is not a recommended use for "debe".

2) @>debe if=/dev/boot of=/dev/fd0 ovsize=1280b -pg

This command line copies the boot partition to floppy diskettes. Since the boot partition is larger than a floppy, the data is broken up into volumes of 1280 blocks each, the size of the floppy. The "-pg" option is used to indicate progress while "debe" is working.

Notes

When "debe" copies data to a new file, the file is created with read and write access modes only, regardless of the access modes of the source file.

Extremely large buffer sizes ("bs= ") that work in the single volume case can cause the error message:

```
debe:  cannot allocate storage
```

if an "ivsize=" or an "ovsize=n is specified that is not a multiple of the buffer size.

When using "debe" with multiple volumes, the following message is printed:

```
debe:  Done with input volume 1.  Do you want to
continue:  If so, mount volume 2 on /dev/rfd0 before
answering.  Answer yes or no:  y  Working on input
volume 2.
```

Format @>delete {-i} {-q} {-r} {-s} file1...filen

Description The "delete" command removes the specified files from a directory. The files may be directories. If a file has multiple names, delete removes only the names included in the list of file names.

Options

-i Interactive. You are prompted for the files to delete. If "-i" is used with file names, the interactive option is ignored and the file is deleted.

-q Query. You are asked to verify that a file should be deleted. A response of either "yes" or "y" deletes the file; any other response is interpreted as "no", and the file is skipped.

-r Recursive. When a file to delete is a non-empty directory, delete with the "-r" option recursively deletes all files and then the directory itself.

-s Silent. Entries are removed without reporting the names of files being deleted.

Examples

1) @>delete test.j

This command line removes the file test.j

2)
```
@>delete -i
delete>x.j
delete>y.j
delete>^D
```

This command line prompts you for a file to delete, and continues to prompt for file names until you type either;

|CTRL| + |D|

or

|RETURN|

alone at a prompt.

Notes You must have write permission to the directory containing the files in order to delete them.

Format @>devfmt {options} {parameter=value} raw_device_name

Description

The "devfmt" command formats the specified "raw" (unbuffered) floppy drive for the number of tracks specified by "n=" starting with the track specified by "s=". If no starting track or number of tracks is specified, default values are used. Specifying a starting track without specifying the number of tracks (or vise versa) is allowed (see examples below).

Verification of tracks formatted can be accomplished in three ways:

1. The "-v" option does a read comparison of the data field,

2. The "-rw" option does a read-write-read comparison of the data field with a different data pattern.

3. The "-a" option, which requires the disk to be formatted before analysis can be done, performs a read comparison of the data field.

If a bad block is found during formatting with verification, the track is reformatted. The verification continues with a check of the same block. If that block fails again, it is recorded as a bad block. Note here that you are able to set the format retries to any positive number you like via the "r=" option. This enables you to reformat a bad track as many times as necessary.

Options and Parameters

s=	starting track number. Default is track 0
n=	number of tracks to format.
r=	number of format retries. Default is 1 (-v or -rw must be used).
-v	read only verify.
-rw	read-write-read verify.
-a	analyze format (tracks must be already formatted)

Examples

1) @>devfmt s=10 n=30 /dev/rfd0

This command line formats the 30 consecutive tracks starting at track number 10.

2) @>devfmt n=3 r=0 -v -a /dev/rfd0

This command line performs a read only verification of the first 3 tracks with no format retries. Those tracks must already have been formatted.

Notes

Care must be used in specifying the options. Any time the "-a" option is used, no formatting will occur unless a bad block is found and the number of retries is greater than 0.

Format @>diff filel file2

Description The "diff" command shows what lines differ between two string files.

Lines are noted as deleted, created or changed in the order the files are specified. No output is generated if the files are the same.

Examples The following examples use these three files:

filel	file2	file3
aa	bb	bb
bb	cc	ce
cc	dd	dd

1) @>diff filel file2

```
-------- 1 line deleted at 0:
aa

-------- 1 line added at 3:
dd
```

2) @>diff file2 file3

```
-------- 1 line changed at 1 from:
cc
-------- to:
ce
```

Format @>diskusage {-v} {file1 ... filen}

Description The "diskusage" command shows the number of blocks on disk that are being used by the specified files or directories. If no files are specified, the disk usage of your current working directory is given.

With the n-v" option, the names of each directory (including normally hidden subdirectories) and file, and the number blocks used by each are displayed. On the final line, the total disk usage is given for all the specified directories.

Options -v Verbose. The name of each directory and subdirectory is displayed along with its disk usage.

Examples 1) @>diskusage /etc
```
        11 /etc/maillists
         7 /etc/mailboxes
        14 /etc/ttydesc
     Total blocks: 238
```

The disk usage in blocks for each subdirectory in "/etc" is displayed, followed by the total disk usage for "/etc."

Notes Since diskusage only looks at the size of the file and not at the bit map describing each block, it can only give an approximation of the number of blocks used.

Since it does not take into account fielsystem indirect blocks, diskusage underestimates the size of large files. In addition, if there are any linked files in the subdirectories, diskusage cannot accurately reflect the true usage.

Format

```
@>do "command string"
```

Description

The "do" command executes a command string. If the command string has one or more arguments, enclose the entire string in single or double quotation marks.

By using variables in the command string, "do" can be useful in defining abbreviations.

Examples

1)

```
@>do "diskusage /usr/jws /usr/sdm"
  10 /usr/jws
 260 /usr/sdm
total blocks: 270
```

This command line executes the command string with the single command diskusage with two arguments.

2)

```
@>#b du do "diskusage $r1"
@>du /usr/jws /usr/sdm
  10 /usr/jws
 260 /usr/sdm
total blocks: 270
```

This pair of command lines first defines an abbreviation with a variable, and then uses the abbreviation to process the specified files.

Format @>dump {options} {file {starting-address count}}

Description The "dump" command displays the contents of the file in 16-bit words unless otherwise specified (bytes or characters). It dumps the data in hexadecimal unless otherwise specified (character, decimal or octal).

dump assumes the starting address and the count are in decimal unless there is a leading 0 -- interpreted as octal, or a leading OX -- interpreted as hexadecimal (as in C).

Options

-a	Displays the contents in hexadecimal along with the alphanumeric string.
-b	Displays the content in bytes.
-c	Displays the contents in characters.
-d	Displays the contents in decimal (16 bit words).
-k	Displays the keyed portion of a keyed file.
-o	Displays the contents in octal.

Examples 1) @> dump /bin/dump 0 0x20

This command line displays the first 32 (0x20) bytes of the dump program file in hexadecimal (by default).

Format @>echo {-nnl} {-nl|-fl} message

Description The "echo" command displays the specified message. It can be used to display a message to the terminal screen, for instance, when inserted in a command file, to display remarks reflecting the progress of the program.

Pattern matching characters are accepted in the message.

Output is to your terminal unless it is redirected elsewhere.

Options -nnl This option suppresses a line feed at the end of each message.

-nl or -fl This option inserts a new line at the end of each message. It is useful for creating filelists.

Examples 1) @>echo good morning
good morning

This command line simply displays the specified message.

2) @>echo /bin/d»
/bin/date /bin/debug /bin/deldir /bin/delete
/bin/diskusage /bin/dump

This is a simple way to list the names of all the files that match a pattern.

3) In a command file:

```
Is /usr/dir > /usr/dir/list
echo "V usr/dir* has been listed to '/usr/dir/list'"
move /usr/dir/list /usr/records
echo "file '/usr/dir/list* has been moved to V usr/
records'
```

In this example, "echo" is used to notify the user of the progress of a command file.

Format @>errlog {file} {options} {parameters}

Description The "errlog" command initializes and displays the contents of the system error log file. When a system error occurs, it is logged to the file "/etc/error.log" by "errlog", which is run in the background.

If the "errlog" command is used without specifying a file, options or parameters, the contents of the active error.log file are displayed.

Options and Parameters

-quiet Suppresses display of error log.

-initial Initializes the error log file. The file must be specified.

size= Sets the maximum number of errors an error log may have. The default is 100. The size can only be specified when initializing an error log.

errors= Sets the current number of errors an error log begins with. The default is 0. The number of errors can only be specified when initializing an error log.

Examples 1) @>/sys/errlog /etc/error.log -initial errors=10

This command initializes the system error log, keeping the last 10 errors.

Notes The "errlog" command is in the "/sys" directory.

Format @>exec {options} {parameter=value} program

Description The "exec" command starts a program running. The program can be run in foreground or background mode, or on another terminal that is not in use. Input, output and error messages can be redirected.

Options and Parameters All options except "ceils" and "floors" may be abbreviated by their first letter.

alias=	This parameter specifies a name for the process to be used in the "PROGRAM" column of the "ps" display.
ceils= floors=	These parameters specify the upper and lower bounds for the priority of the process. They may be any number between 0 and 9.
i=	Input. This parameter redirects standard input to take input from the specified device or file. Default is the keyboard.
o=	Output. This parameter redirects standard output to the specified device or file. When the "-combine" option is used, the output device must be specified. Default is the terminal screen.
e=	Error. This parameters redirects standard error message output to the specified device or file. Default is the terminal screen.
t=	This parameter specifies the terminal device on which the program is to be run.
-append	This option appends output and error messages to the devices specified by "osoutdev" and "eserrdev" without writing over the data already in those files.
-combine	This option combines the output and error message devices. The device or file must be specified by the "osoutdev" option.
-detach	This option executes the program in background (detached) mode.
-notify	This option reports completion of the program with a message. The "-notify" option runs the process in background mode.

-silent This option suppresses printing of the child process number when the "-notify" option is used, and the message displayed when the "-wait" option is used.

Examples

1)
```
@>exec t=/dev/tty2 -detach basic

@>
```

This command line starts BASIC on terminal tty2. BASIC runs in foreground on tty2 but in background from the host terminal.

2)
```
@>exec i=/dev/null -notify -combine -detach o=prm.log
program
223
.
.
.
Process 223 completed due to normal completion
returning 0.
```

This command line starts the program "program" in background mode. Output and error messages are redirected to the file "prm.log." Input is taken from the null device. The user will be notified upon completion of the program.

Format filemodes {+|-}{rwx} {+|-}{rwx} {-m} file1...filen

Description The "filemodes" command establishes the access modes of the specified files, permitting access to the owner and to all other users. The letters stand for read ("r"), write ("w") and execute ("x") permission. The first set applies to the owner, the second set to all other users. Only the owner or the administrator can change the nodes of a file.

Options

+	Used with a "r", "w" or "x", adds that mode.
-	Used with a "rn, "w" or "x", subtracts that mode.
r,w,x,	Without + or - gives only that mode.
-m	Changes the modes for all other users to be the same as those for the owner.

Examples 1) @>filemodes rw -w test

This command line changes the modes to allow read and write access only to the owner of test, and to disallow write access to all other users.

Notes Options must be specified in the order given. A mode string must be given for both the owner and for all other users. If no access is desired, it can be specified as a null string and represented as a pair of quote characters, either "" or

Format @>fschk device_name {options}

Description The "fschk" command checks and optionally fixes the filesystem on the specified device.

An exit code is also returned by "fschk." The exit codes are described in table 2-1.

Options The following options can be abbreviated to the first letter.

-i Inode. If this option is followed by an inode number, the name of the file using that inode is returned.

-b Block. If this option is followed by a block number, the name of the file using that block is returned.

-fix Fixes repairable filesystem errors on an unmounted filesystem.

-quiet Displays no output. This option is useful only when "fschk" is started by a process that uses the exit code returned by "fschk." This option is ignored when the "-u or -utility" option is used.

-utility Displays output using the utility program message files.

Examples

```
@> fschk /dev/fd0
```

Checks for filesystem errors on "/dev/fd0" but does not attempt to fix them.

```
@> fschk /dev/fd0 -u -f
```

Checks for filesystem errors on "/dev/fd0" and attempts to fix them automatically. Uses the utility program.

```
@> fschk /dev/boot -i 23

Inode 23 is file /bin/makedev.
```

Returns the name of the file on "/dev/boot" using inode 23.

Notes If you want more control over the examination or repair of a fiesystem, you can use the standard utility program "/util/fsdbg."

A program that spawns "fschk" and uses the "-quiet" option should be able to interpret the following exit return codes and produce appropriate error messages when necessary.

Table 2-1A. "fschk" Exit Codes

Exit Code	Description
< 0	System error number that caused "fschk" to terminate prematurely.
0	Normal termination with no file system errors detected.
100	User chose to abort during normal exectution.
201	No device name was specified in command line.
202	Device name incorrectly specified.
203	Error by "fschk" in parsing command line for options.
204	Unknown option found on command line.
205	Error by "fschk" in parsing command line for options.
206	Could not open specified device for reading.
207	Could not open specified device for writing and so could not fix the filesystem on that device (e.g. device may be mounted).
208	Attempt to use "-fix" option by some account other than "admin".
209	User chose to abort after a file i/o error, This should occur only when the "-utility" option is used.
300	Normal termination. The exit code is 300 plus the number of errors detected.

Files Used

The following files are required when the "-utility" option is specified:

/util/utmsg.txt
/util/utmsg.ind
/util/uthelp.txt
/util/uthelp.ind
/util/stderror

Format @>install device PID1 {PID2 ... PID3} {options}

Description The "install" command installs software products from either diskette or magnetic cartridge streamer release sets. Installation from diskette is specified by entering "fd0" or n/dev/fd0w for the device, and installation from cartridge by entering either "res" or "/dev/rcs." The Product Identification Code (PID) is a three letter code associated with each software product.

Installation of several software products at one time is allowed only for installation from cartridge. In this case, all the products must be on the same cartridge. For installation from diskette, only one product can be specifed in each command line.

Options and Parameters

to=path/name	This option allows you to specify a directory path to be prefixed to the files in the software product release set. This allows a product to be installed onto a mounted filesystem.
-query	This option only applies to installation from cartridge. The release level file for each product being installed is displayed, and you are asked to verify that this is the correct product.
-verify	This option causes "mcscompare" to be run after the files have been installed.

Examples 1) @>install /dev/fd0 EUT

This command line installs the EUT product, the 2000 utility set, from diskette.

2) @>install fd0 EUT to=/out

This command installs the EUT product from diskette to the directory '/out'.

3) @>install cs EUT EBS EIT EDB EDS

This command line installs the five software products specified from magnetic cartridge steamer. Release level files are displayed before installation, but verification is not requested prior to installation.

Format @>install key /absolute/file/path/name

Description The "install_key" command opens the specified file and reads the current "public key" from the file. You are then asked to type the new public key or press RETURN for no change. If you enter a new public key, you must enter it twice. After being correctly entered twice, the new public key is written to the file.

Examples 1) @> install_key /etc/level/EUT
Current Public Key: 12345678

Input 8-character public key (cr=no change):

The command line displays the current public key of the EUT software product and prompts you for the new key.

Format @>kill {signal=#|-#} pid1 ... pidn

Description The "kill" command forcefully terminates each of the currently existing processes specified. If the process does not exist, this message is displayed:

```
kill: process pid does not exist
```

Only the process owner or the system administrator can kill a process.

Processes that have terminated (EXIT state) are unaffected by "kill". Only a process that waits for the child can clear the entry. A user program that does a fork/exec but does not ever wait for the child might cause this state.

The "signal=" parameter can only be used if the "/include/signal.h" file is present. It is included in the optional Program Development Package (EDS).

Parameters signal=# or -# This parameter is not used at this time, release 7.2A.

Examples @> kill 104 296

This command terminates the two processes, 104 and 296.

Format @> kychk filename {-data} {-listkeys} {-fix}

Description The "kychk" command checks and optionally repairs the keyed file "filename." Keyed files include the "direct" and "sort" file types.

2000 keyed files are managed by C-ISAM file organization processes, which are based on b-tree structures.

Options All of the options may be abbreviated by their first letter.

-data This option specifies that each data record is to be examined.

-listkeys This option displays information about each b-tree node along with detailed information about each entry in the node. (See notes.)

-fix This option fixes repairable keyed file errors.

Examples @>kychk test -1

The file "test" is checked for errors, but no attempt is made to repair them. Any errors found are reported.

@>kychk test -d -f

The file "test" is checked for errors, including errors in the data record. An attempt is made to repair any error found.

Notes The following information is included on each entry in a node if the "-listkeys" option is used:

- flag value
- total length
- key length
- duplicated number
- lead count
- trail count
- key value

Format @>login {name} {-quick} {t=time}

Description The "login" command executes the logon procedure program. When the system goes from single to multi-user mode, it automatically runs this program for every terminal that is specified in the "/etc/ports" file.

To log on the system, type your user name (which is defined in the "/etc/passwd" file) on any terminal that displays the prompt:

Account name:

If you have a password assigned, the program prompts you to enter it:

Password:

After a successful log on, the login file is updated and the user is informed of the message of the day.

Options and Parameters

-quick prevents displaying of the message of the day.

t=time specifies a time limit in seconds for the user to log on the system. The default is 3 minutes.

Examples

```
@>login fred -quick
password:
```

Logs Fred onto the system, bypassing the message of the day. Fred has a password assigned, which must be entered at the additional prompt.

Notes Typing:

|CTRL| + |D|

logs you out.

Format @>lpmaint {printer} {options}

Description The "lpmaint" command performs three functions:

1. it changes the status of the entries in the printer queue,

2. it fixes any inconsistent printer queue,

3. it signals a printer that a form has been changed.

Any user can execute the third function. The user who submitted a job and the system administrator can execute the first function. Only the system administrator can execute the second function.

Options Options may be abbreviated by their first letter.

-kill #1 ... #n	Terminates the print jobs specified by job request numbers. If the job is in the queue but not printing, it is removed. If the job is printing at the time of the request, it is stopped and then removed.
-stop #1 ... #n	Stops the print jobs specified by job request number. The jobs remain in the queue with the status "stopped".
-resume #1 ... #n	Starts the print jobs specified by job request number. The job status is changed from "stopped" to "waiting". The jobs will be printed in turn. Unless specified otherwise, printing is resumed at the top of the page on which the job was stopped.
-fix	Checks the printer queue and returns it to a consistent state. This should not be used while despooling is running. It verifies that the linking between print jobs is correct so that system errors do not occur.
-fc	Signals the printer that a new form has been mounted and that printing can be resumed.
-fp	Signals the printer to print the new form for alignment purposes. This may only be done when a job is in a form change state.

-priority job# pri#	Changes the priority of the print job to the new priority specified.
-unit job# device	Changes the name of printer on which the specified job is to be printed.
-form job# form	Changes the form used for the specified print job.
-begin job# page	Changes the first page to be printed of the specified print job.
-end job# stop page	Changes the last page to be printed in the specified print job.
-copies job# copies	Changes the number of copies to be printed by the specified print job.
-dn job#	Turns the delete option for the specified print job to "on".
-df job#	Turns the delete option for the specified print job to "off .
-nn job#	Turns the notify option for the specified print job to "on".
-nf job#	Turns the notify option for the specified print job to "off".
-qn job#	Turns the requeue option for the specified print job to "on".
-qf job#	Turns the requeue option for the specified print job to "off".
-rn job#	Turns on the "raw" option for a job. When the job is printed, all mnemonics processing and character translation is bypassed. If this option is set, starting and stopping page numbers are ignored. The file is printed from beginning to end.
-rf job#	Turns off the "raw" option for a job.
-wn job#	Turns on the "wait" option for a job. This causes the despooler to delay printing the file until all processes have closed their access to this file.
-wf job#	Turns off the "wait" option for a job.

-time time/date	Delays despooling until the specified time/date The format options for the date are: hhmm MMDDhhmm MMDDhhmmYY where hh is hours, mm is minutes, MM is months, DD is days and YY is years.
-version	Displays the version level of the spooler software.

Examples

1) @>lpmaint stop 2 4 -kill 1 5

This command line stops print jobs 2 and 4, and kills jobs 1 and 5

2) @>lpmaint -begin 4 3 -copies 4 5

This command line changes print job 4 so that printing begins on page 3 and 5 copies will be printed.

Files Used

/etc/printers	For the location of the printer queue.
/etc/_queues/lpq.printer.killev	For the printer's kill event count file name.
/etc/_queues/lpq.printer.info	For the printer's name and job number.
/etc/_queues/lpq.printer.form	The form event count file.

Format @>lpq {printer} {-version}

Description The "lpq" command displays the entries in the queue of the specified printer. If no printer is specified, then all printers in the default queue are displayed. The following information is displayed.

Job — Print job number. This is the number that the print job was given when it was submitted to the queue.

Pri — Priority. The priority can be any number from 0 to 9. Priority 0 jobs are not printed. Priority 9 is the highest priority.

State — Current state. The five states are:

- working (currently being despooled)
- waiting (waiting to be printed)
- fwait (waiting for a form change)
- error (an error occurred during printing)
- stopped (suspended, not printing)

User — The name of the user who submitted the print job.

File/Alias — The name used to notify the user when the job is completed. If none is specified, then the full path name of the file is used.

Pr — The name of the printer where the file is to be printed.

Form — The name of the form on which the job is to be printed. The default form is "stnrd".

— The number of copies to be printed.

D — The delete option. "Y" specifies that the file will be deleted from the system upon complete, "N" specifies that it will not be deleted.

N — The notify option. "Yn specifies that the user will be notified when the job is completed, "N" specifies that he will not.

R — The requeue option. "Y" specifies that the job will be requeued when it is completed, "N" specifies that it will not. Jobs are requeued with priority 0.

RP — Restart page. The first page of the file to be printed. If the job is stopped and restarted, printing resumes at this page.

	SP	Stop page. The last page to be printed. If this is 0, then the entire file is printed.
	H	Hold priority. -The hold priority can be any number from 0 to 9.
Options	-version	Displays the version level of the spooler software.
Examples	1) (See Figure 2-1.)	

```
@>lpq
Printer queue:                                          rev = ESP 7.2A
Job Pri State    User     File/Alias      Pr Form    # D N R RP  SP H
 10  9  working  dianne   /tmp.lpr0.128   P1 stanrd 5 Y Y N        0
  3  4  stopped  jeff     /src/cmds/lpq.c P1 stanrd 1 N Y N  2   5 0
 12  5  working  alien    /usr/allen/dved P0 stanrd 1 N N Y        1
 11  2  waiting  szy      /usr/szy/format P0 stanrd 1 N Y N  5     1
 13  2  fwait    jon      /usr/jon/payrol LP checks 1 N Y N        1
```

Figure 2-1. Example of "lpq" Command Usage and Display

With no printer specified, the queues of all printers are listed.

Notes

A "working" state indicates that the despooler is attempting to print the job. Printing may be interrupted, however, and the state be "working" if the printer is offline, out of paper or ribbon, in use by redirected output or BASIC, or if admin has manually "killed" the despooler while it was running. To check the reason for a job not printing while the state is "working," use "lpstat."

lpq (cont'd)

Files Used	/etc/printers	For the location of the printer queues
	/etc/_quesues/printer.killev	The print queue kill event count file.
	/etc/_queues/printer.info	Name and packet number for the kill request.

Format @>lpr {options} {parameter=value} filel ... filen

Description The "lpr" command places string files in printer queues for printing. If there are no files given, it reads from standard input into a temporary file, and then places this file into a queue for printing.

Options and Parameters All options may be abbreviated by their first letter.

alias=name	Specifies a name (the alias) to be used instead of the file name when notifying the user of completion. This name is displayed as the file name in the "lpq" report.
class=name	Specifies the name of a print job class contained in the file "/etc/class." Print job parameters that are not specified are supplied from the class definition.
copies=#	Specifies the number of copies to be printed.
form=name	Specifies the name of the form on which the job is to be printed.
list=printer	Prints the file on the specified printer.
priority=#	The print job priority. The priority may be any number from 0 to 9. 9 is the highest priority. Jobs with priority 0 are not printed until the priority is changed to above the hold priority.
-delete	Deletes the file when the job has been completed.
-notify	Notifies the user when the job has been completed.
-off	Turns spooling off. The job is printed directly without going through a print queue.
-raw	When the job is printed, all mnemonics processing and character translation is bypassed. This flag would be used to print a file that was generated by the graphics system.
-requeue	Places the file at the end of the queue with priority 0 after the current print request has been completed.

-spool	This option forces the job to be spooled, overriding the "/etc/defaults" file when it specifies that spooling is off.
time=time/date	This causes the despooler to delay printing of the file until the date specified. The format options are: hhmm, MMDDhhmm and MMDDhhmmYY.
-version	Displays the current level of lpr.
-vfu	This option unconditionally sets the electronic vfu (top of form) when spooling is turned off ("-off"). Make sure the paper is set to top of form before executing this option.
-wait	Spools the entire file before starting to print the job.

Examples

1) @>lpr temp .globals Makefile

This command line prints the files "temp," ".globals," and "Makefile." The files are submitted to the spooler which then places them in a print queue.

2) @>lpr -notify alias=expense_report copies=2 expenses

This command line prints two copies of the file, expenses, and notifies the user upon completion. The notification message refers to the print job as "expense_report."

Notes

The names of printers are contained in the file, "/etc/printers." The printer queue is the file "/tmp/_queues/xxx.que," where xxx is the printer name.

Files Used

/etc/_qtemps/lpr*	For temporary files ("?" is used as a pattern matching character).
/etc/printers	To find the location of the printer queue.
/etc/forms	To verify form definitions.
/etc/class	For the printer class information.
/etc/defaults	For system printer default values.

Format @>lpstat {printer} {-version}

Description The "lpstat" command reports the status of all printers on the system. With the "-version" option, the level number of lpstat is displayed. The information displayed by lpstat is:

Printer — The printer's name.

PID — The process identification number of the printer's despooler.

Job# — Print job number.

Form — The current form installed on the printer.

Copy — Number of the current copy being printed.

Page# — Page number currently being printed.

Status — Current status of the printer. The possible printer statuses are described in the following table.

Table 2-3. Printer Status Options

STATUS	MEANING
idle	Despooler is not processing a print job
printing	Despooler is printing a file
chng form	Despooler is waiting for a form change
offline	The printer is off line
no paper	The printer is out of paper
mnem err	A mnemonic error was encountered while printing.
not init	Printer has not been initialized.
i/o error	An input/output error occurred while printing.
no ribbon	The printer is out of ribbon

Examples

```
@>lpstat
Printer   PID   Job#    Form    Copy#  Page#     Status
   p0      26    23     stnrd     1      15      printing
   lp      25    14     stnrd     1      10      no paper
```

Files Used /etc/printtab The printer status file.

Format ls {options} {file1 ... filen}

Description The "ls" command lists the file names and additional information about the files specified. If no file names are given, the current directory is assumed. Otherwise for each argument, "Is" determines if it is the name of a directory. If it is a directory, "Is" lists its contents unless the "-dir" option was used. If a file was specified, only its name is printed. Additional information can be requested by using the options described below.

Options

-a All. This option shows all files, including files whose names begin with '.' which are normally considered "hidden" files.

-b BFS. This option displays additional BFS information.

-blks Blocks. This option displays the number of blocks used by the file.

-date This option displays the latests file creation/ update date.

-dir Directory. This option lists the specified directory entry instead of the files in it.

-fileno File number. This option displays the file number, which can be used for debugging purposes.

-1 Long. This option shows the length of the file in bytes, the access permission, the owner, the modification date, and the name of the file. If the file is a directory, a DIR precedes the listing. If the file is a device file, a "U" or "B" (Unbuffered or Buffered) or "R" (remote) is displayed followed by the major/minor device numbers. If the file is an event count, the word "EVENT" appears. If it is a system event count, the word "EVENT" is followed by the event count number. If the year the file was created or modified is different than the current year, it is included in the listing.

-links Links. This option shows how many directory path names, or links, a file has. Links are made with the "addname" command.

-modes This option displays file access modes.

-owner This option displays the name of the file's owner.

-P Path name. This option shows full directory path names rather than just the entry name. This only works if the file arguments are full path names.

-r Recursive. This option lists all directories and files from the specified directory to the end of the directory structure.

-rev Reverse. This option lists the files is reverse order. When used in conjunction with "-time," displays modification from newest to oldest, otherwise the sort is reverse alphabetical order.

-size This option lists the size of the file in bytes.

-t Type sort. This option sorts the listing by file type.

-time Modification or Creation Date. This option lists the files in order of modification, oldest to newest.

Examples

1) @>ls stuff junk trash

This command line lists "stuff," "junk," and "trash" if they are files, or their contents if they are directories. If any don't exits, they are skipped.

2) @> Is -1 work
365 rw. rw. john Aug 20 13:40:06 work

This command line displays, in long form, the file "work". The information included in the "long" display is:

1) size in bytes
2) read-write-execute permission owner others
3) owner
4) modification date
5) file name

3) e>ls -time «.c

This command line lists all files matching the pattern "*.c" in order of their creation or most recent modification time.

Notes

You must have read permission to a directory in order to list its contents.

Format @>makedev name {u|b} major minor

Description The "makedev" command creates a device file entry. The first argument is the name of the entry. The second is either "u" for unbuffered (direct access to the device driver) or "b" for buffered (access through the system buffers). The last two numbers specify the major and the minor device types (e.g. unit, drive, or line number). These numbers are assumed to be decimal.

For buffered devices there is usually an associated unbuffered device, called the "raw" device. The major and minor modes for both devices is the same, although they have different names and one is buffered and the other unbuffered. For example, a listing of your "/dev" directory will probably show both of these device entries:

```
Dev U 14,0   rw. rw. admin date time rwdO
Dev B 14,0   rw. rw. admin date time wdO
```

The first entry is for the raw device, and the second is for the buffered device.

Some commands that work on devices can process a raw device more quickly than they can process buffered devices.

Options u Unbuffered

b Buffered

Examples 1) @> makedev /dev/null u 0 0

This command line creates the null device entry, an unbuffered device.

2) @>makedev /dev/fdO b 7 0

This command line creates the floppy drive 0 entry.

Notes The assignment of major and minor device numbers is specific to your system.

Although device files can exist in any directory, by convention they are in "/dev", or in "/sta" for remote devices. Device files can have multiple file names, but it is not recommended.

Format makedir directory1 {... directoryn}

Description The "makedir" command creates new directories with the given path names. All directories in each path name except the last one must already exist. The last name in the path name must not exist.

Examples 1) @> makedir /usr/john/jdoc

This command line creates the subdirectory "jdoc" in the directory "/usr/john." Both directories, "/usr" and "/usr/john," must already exist.

Format @>makeec {sys= } ed {... ecn}

Description The command "makeec" makes an eventcount file.

Parameters

sys=# or s=#	Creates the specified eventcount files as system eventcounts. It is maintained for the use of system functions rather than for the use of special programs.

Examples 1) @>makeec s=1 /dev/clock

This command line makes the system clock entry.

2) @> makeec /usr/jeff/syncec

This command line creates a normal user eventcount.

Notes Making an eventcount with the same name as an existing file removes the file entry and its value and replaces it with the new eventcount entry.

Currently defined system eventcount numbers are:

sys=1 System Clock

This is advanced at fixed intervals (1/10 second).

sys=2 Child exits

This is advanced whenever a child process exits or suspends. The eventcount is advanced once per process.

Only the "/dev/clock" system eventcount is available to users.

Eventcounts can be deleted with the delete command.

Format @>makefs device {size} {file=value}

Description The "makefs" command creates a filesystem on the specified device. This is done by writing the following information at the beginning of the device:

o the filesystem header

o file descriptors

o the bit maps.

The first six blocks of the device are skipped by "makefs". The number of file descriptors is rounded up to fill out a block.

Any block found to contain an earlier filesystem header is erased.

Options and Parameters

size The number of blocks in the device. If size is not specified, the total number of blocks in the partition is used.

file= This option specifies the maximum number of files to be allowed in the filesystem. If no number is specified, 9/64 of the number of blocks is the number used.

Examples

```
@>makefs /dev/rfd0 1280
making freemap
making fdmap with 184 fds
making new fds
 1248 blocks available, writing header
```

This command line creates a filesystem on the diskette in the diskette drive fd0. There are 184 file descriptors and 1248 blocks available for use.

Notes Specifying the "raw" device makes the process slightly faster.

Format @>makesta name network# station#

Description The "makesta" command creates a remote station file in two parts, given a network address consisting of the network address number and the station number.

This is used with the local area network system. Further description of the network addresses can be found in the documentation for that software.

Example @>makesta /sta/payroll 0 17

Format @>makettymntbl terminal name

Description The "makettymntbl" command produces a table which is used by the operating system to implement mnemonics on the specified type of terminal. The parameters required by "makettymntbl" are supplied by the user in a file.

The parameter file must be in the V etc/ttymntbl" directory and must be called "terminal_name.mntbl" (the terminal name with ".mntbl" appended). The name of the terminal must also appear in the file "/etc/terminals".

The file must have one line for each function. The function name must be followed by hexadecimal encoded bytes separated by spaces. Any functions not included in the list are assumed to be non-functional on the specified terminal.

The following functions are available. If a function is not present, it is set to null.

Table 2-4. Terminal Mnemonics Functions

FUNCTION	DESCRIPTION
home	cursor home
clear	clear screen
eos	clear to end of screen
eol	clear to end of line
left	cursor left
right	cursor right
up	cursor up
down	cursor down
ic	insert character
dc	delete character
il	insert line
dl	delete line
b_normal	bright normal
b_underline	bright underline
b_blink	bright blink
b_blink_underline	bright blink and underline
b_reverse	bright reverse
b_reverse_underline	bright reverse and underline
b_reverse_blink	bright reverse and blink
b_all	bright reverse, blink and underline
d_normal	dim normal
d_underline	dim underline
d_blink	dim blink
d_blink_underline	dim blink and underline
d_reverse	dim reverse
d_reverse_underline	dim reverse and underline
d_reverse_blink	dim reverse and blink
d_all	dim reverse, blink and underline

Table 2-3. Terminal Mnemonics Functions (Continued)

FUNCTION	DESCRIPTION
xy	move cursor to x,y
read_cursor	read cursor position
clear_foreground	clear foreground
expanded_print	expanded print
start_protect	start protect
end_protect	end protect
transmit_screen	transmit screen
page mode	print page

Examples 1) @>makettymntbl evdt

This command line creates the file "/etc/ttymntbl/evdt." Parameters are read from the file "/etc/ttymntbl/evdt.mntbl." The file "evdt.mntble" the name of the function and, to the left, the hexadecimal code for that function. The first few lines of this file are shown in the following figure.

```
home                    IB 5B 48
clear                   1B 2A
eos                     1B 59
eol                     1B 54
left                    1B 5B 44
right                   1B 5B 43
up                      1B 5B 41
down                    1B 5B 42
ic                      1B 51
dc                      1B 57
il                      1B 45
dl                      1B 52
```

Figure 2-2. Sample Mnemonic Table File

Notes "Delay" inserts about a 50 ms delay after mnemonics which scrool the screen, if set to a non-zero value. This is used for termianls which do not support flow control.

"Start-narrow" begins normal screen width mode.

"Start-wide" begins 132 column mode, which is supported only on the EDT terminal.

Format @>makettyxlate table name

Description The "makettyxlate" command is used to create and modify translation tables used by the command "ttyxlate". The translation table is a 256 byte long record, arranged as a 16-by-16 (00 through FF) position matrix, which specifies how each numeric representation is to be translated.

	00	10	20	30	40	50	60	70	80	90	AO	B0	CO	DO	E0	F0
00	00	10	20	30	40	50	60	70	80	90	AO	B0	CO	DO	E0	F0
01	01	11	21	31	41	51	61	71	81	91	A1	B1	C1	D1	E1	F1
02	02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2
03	03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3
04	04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4
05	05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5
06	06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6
07	07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7
08	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
09	09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9
OA	OA	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA
0B	0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB
0C	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC
0D	OD	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD
0E	OE	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE
OF	OF	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF

Figure 2-3. Translation Table Matrix

The hexadecimal code for a character in position XY in the table specifies which character is to be displayed when the code XY is transmitted to the terminal. So, if position 7A (hexadecimal) in the table contains the code 41, the ASCII code for "A", the character "A" will be displayed when the hexadecimal code n7An is transmitted to the terminal.

When the "makettyxlate" command is executed, the 16 by 16 matrix of hexadecimal numbers is displayed. To change the code at any position, move the cursor to that position and type over the current code. Cursor movement is done with these commands:

|CTRL|+|P| moves the cursor up one line

|CTRL|+|N| moves the cursor down one line

|CTRL|+|H| moves the cursor left one character

|CTRL|+|F| moves the cursor right one character

|CTRL|+|C| ends the editing session

When a new table is first created, by executing the "makettyxlate" with a new table name, the positions are filled with codes 00 through FF, so no translation occurs.

Translation does not occur until the "ttyxlate" command is executed.

Notes

The translation is affected by the selection of terminal modes with the ttymodes command. The operation of mnemonics is affected when translation is in effect. Before using the translation tables, the user should consider mode setting changes resulting from use of the commands ttymodes, ttyxlate, makettymntbl and makettydesc.

Format @>match t-s} {-n} {-not} {-v} pattern {file1 ... filen}

Description The "match" command searches each line of the named files for all occurrences of the given pattern. If no files are specified, "match" searches lines read from standard input. The pattern can consist of literal characters and pattern matching characters.

All lines that match the pattern are output to standard output.

Only one pattern string can be specified for each match, but there is no limit to the complexity of the string.

Options

-s Suppresses the printing of the filenames.

-n Includes the file line number of each input file.

-not Prints all lines that do not match.

-v Verbose.

Examples

1) @>match printf *.c

Displays all lines containing "printf" in the ",c" files.

2) Is -l | match admin

Prints only the lines of the directory listing that contain the word "admin".

Notes See "change" for changing a pattern in a file.

To take pattern matching characters literally, enclose them in quotation marks to prevent interpretation by the command interpreter.

Format @>mcscompare {options} {parameters} {savesetl ... savesetn}

Description The "mcscompare" command compares files on an MCS cartridge to files on disk, and reports both files that are common to both media, and files that are different.

Options and Parameters

filesystem=	This parameter causes "mcscompare" to open the specified filesystem device rather than the filesystem device in the saveset. This allows the user to compare a filesystem after a restore, when the filesystem restored was one other than the one specified in the saveset file.
ip=pathname	If the files were saved with the ip option then the part of the path name stripped off will have to be given so that the file can be opened on disk.
-all	This option compares all savesets on the cartridge.
-errors	This options specifies that only name of files that miscompare, i.e. that are not the same, are displayed.
-fn	This option specifies that only the file's existence on tape, and not the data, is to be verified.
-query	This option specifies that the user is to be asked before the compare is begun for each saveset.
savesetl ... savesetn	This is a list of savesets to compare.

Examples 1) @> mcscompare dev=/dev/cs name=backup

This command line compares files in save set backup.

Format @>mcslabel {options} {parameters}

Description The "mcslabel" command allows a user to read and write a tape label using the specified magnetic cartridge streamer device.

Options and Parameters

-brief — Specifies a brief presentation of the tape label. On one line, the set name, date of creation, and date last written is displayed.

ser=number
set=setname
id=name — When any of these options are specified, it causes the new label to be written to tape. The entire tape is erased, the label is written, and the new label id displayed.

The maximum length for each of the fields is eight (8) characters. Anything longer is truncated.

-query — Query before writing a tape label. The current label will be displayed, and the user will be asked if this is the desired tape. A response of "y" will allow a new label to be written, "n" will stop the label process.

Examples

1) @>mcslabel

This command line displays a complete tape label.

2) @>mcslabel set=backup id=source -q

This command line displays the old label and prompts you for verification that the right tape is in the drive. If you verify it, the new label is written.

Format @>mcslist {-all} {-long} {-query} {setname1 ... setnamen}

Description The "mcslist" command displays the contents of the specified savesets on the MCS cartridge.

Options

-all This option displays all the savesets on the tape.

-long In addition to the file name, file information will be included in the output. As appropriate for each file, information includes:

- o file type
- o number of records
- o record length
- o key length
- o major/minor numbers
- o modes
- o date of last modification

-query Before a saveset is displayed, the tape label is displayed. When the query option is selected, the user will be queried to determine if this is the desired tape. If "y" is the response the search and presentation of a saveset will begin. When the response is "n," the process is stopped, unless the -all option is used.

setname A saveset is a group of files that are backed up. A saveset file, containing a list of the files in the saveset, precedes the backed up files. There can be more than one saveset on a tape. The user can request to view the contents of one or more of these savesets on the tape. When none are specified, only the first is displayed.

Examples 1) @>mcslist -1

This command line displays the tape label and, in long format, the contents of the first save set on tape.

Notes When an entire filesystem is backed up, the save set will have only one entry. That entry will be the name of the filesystem. An individual accounting of the contents within that filesystem is not available.

Format

mcsrestore {options} {parameters} {file1 ... filen}

Description

The "mcsrestore" command allows a user to transfer files and filesystem images from the MCS cartridge to disk.

Only the system administrator can restore a filesystem.

Options and Parameters

before=date
Only files found in the saveset with a date earlier than the specified date will be restored.

-contiguous
Files will be restored to disk in contiguous blocks.

dups=
This parameter specifies the action in case the file to be restored is already on disk. The values for this parameter are:

- o "replace", which replaces the original file with the restored file
- o "skip", which skips the restore file and goes on to the next file
- o "query", which prompts you for the specific action to take at each occurrence

-ignore
When a continuable tape read error occurs, a message will appear on standard error, but no prompt will be given to continue. The rest of the file will be skipped and the next file will be restored.

-filesystem
Without this option, it is not possible to restore a filesystem. Additionally, both a saveset name or a filesystem name must be specified.

list=filelist
The files in the file list that match those in the saveset will be recovered.

name=setname
The files from this specific saveset will be restored. When a save set name is not given, the first is used.

-ns
This option restores files with the current user as owner, and default access rights.

-query	The user will be asked for each file, whether or not to restore it.
-recursive	When a directory-is specified as a file to restore, all subdirectories and files found in the saveset will be restored.
since=date	Only files found in the saveset dated later then the specified date will be restored. Date formats are hhmm, MMDDhhmm or MMDDhhmmYY.
sf=file	This specifies the "start file." Files in the list are skipped until this file is encountered. This allows for starting to restore files from where the process was interrupted earlier. The full path name must be specified.
ssnum=number	This parameter specifies the saveset by position on tape (starting with 1).
-stat	This option causes the tape statistics (totoal blocks read from tape and total blocks retried) to be displayed at the completion of the command. The "-verbose" option implies this option.
-tension	The tape will be retensioned before files are restored.
to=directory	Files will be restored into the specified directory instead of the name of the file on tape. This is analogous to the "copy" command.
-verbose	After a file is restored a message is displayed that the file has been restored. This option implies "-stat".
-verify	This option causes "mcscompare" to be run after the files have been restored, to verify the files have been properly restored.

Examples 1) @> mcsrestore / -i -r -t -v

This command line first retensions the tape. All directories and files are then recursively restored from the directory in the first saveset. Tape read errors are ignored. Finally, the names of files that have been successfully restored are displayed.

2) @> mcsrestore -f /dev/root n=rootbackup

This command line restore the filesystem image /dev/root

3) @> mcsrestore n=saturday /src/bin/* -v

This command line looks for the 'Saturday' saveset, restores files found in /src/bin, and lists the names of the files that are recovered.

4) @> mcsrestore dev=/dev/cs /usr/rich/#.c to=/usr/temp -v

This command line restore files from the first save set, from the directory "/usr/rich" to the director V usr/temp", and list the files that are restored.

Notes

Either selected files, a save set, or both must be specified. When no files are given, all files in the specified save set are restored.

If pattern matching characters are used as part of the file names in the command line, the entire file name must be in quotes (e.g. "/bin*").

Format @>mcssave {options} {parameters} {file1 ... filen}

Description The "mcssave" command copies individual files, files found in filelists, and filesystems to the MCS cartridge. Files are written onto tape in the order they are found on the command line, i.e. they are not sorted.

Whenever a backup to MCS is done, a file called a saveset file is created. This file contains the list of all files that will be placed on tape and file information for each. The saveset is written to tape before the files are saved.

Only the system administrator can save filesystems to magnetic cartridge streamer tape.

Options and Parameters

-append	Appends a new saveset at the end of data on the tape.
before=date	Only those files dated earlier than the specified date will be written to tape. Date takes the format of hhmm or MMDDhhmm or MMDDhhmmYY.
-filesystem	Using the first file as the name of a filesystem, an entire filesystem image will be written to tape.
-ignore	Rather than stopping on continuable file errors and prompting for abort or continue, this option will cause the backup to report file errors and continue with the backup of the next file.
ip=path	This option causes the specified path to be stripped from the absolute path name of each file.
list=filelist	The user may specifiy a filelist of files to backup.
name=setname	Each saveset on a tape has a name. It can be up to 64 characters in length (spaces may be used if the name is enclosed in quotation marks). When a name is not given a default name of "operatorname-MM/DD_hh:mm" will be assigned.
-query	The user is queried before each file is placed into the saveset. Responding with "Y" will cause the file to be placed in the save set. "N" will cause the next file to be given.

sf=file This specifies the "start from" file. Files in the list are skipped until this file is encountered. This allows for starting to save files from where the process was interrupted earlier. The full path name must be specified.

-recursive When a directory is encountered in the list offiles to backup, all subdirectories and files in the directory are also placed in the save set.

since=date Only those files dated after the specified date will be written to tape. Date takes the format of hhmm or HMDDhhmm or MMDDhhmmYY.

-ss This option specifies that the files are to be written to tape in start/stop mode (tape motion stops after each file).

-stat This option specifies that the tape statistics (total blocks written, total blocks re-written) are to be given at the completion of the save. The "-verbose" option implies "-stat".

-tension This option, used with "-append", causes the tap to be re-tensionsed before writing files.

-verbose After each file is written to tape, a message will be displayed that the backup of the file completed.

-verify After all specified files have been saved, this option verifies that the data on tape matches the data on disk.

Examples 1) @>mcssave /usr/dennis /usr/dave -r -v

This command line recursively backs up all files in the specified directories and outputs the names of the files backed up.

2) @> mcssave -f /dev/root

This command line makes an image backup of the filesystem "/dev/root".

3) @>mcssave /usr/rich/src/#.c since=12230000

This command line performs a backup of all ".c" files in the specified directory that have changed since midnight, December 23»"d.

Format message {on|off}

Description The "message" command inspects, sets or clears the permission bits which determine whether other users can write to the terminal.

If "message" is entered alone, the current state of message permission is displayed. If "message" is entered with either "on" or "off", message premission for that terminal is set accordingly

Examples @> message on

The terminal can now receive messages.

@> message
Messages are currently allowed

Displays the current message permission for the terminal.

Notes The access permission on your terminal entry controls message reception. When you log on the system, you are given ownership of your terminal. Thus, you can change the access modes through the message command. When you first log on, message reception is allowed.

Format @>mount {-readonly} {-oride} {deviceidirectory}

Description The "mount" command incorporates independent filesystems into the root filesystem. Mounting simply makes an association between the root directory on the device and the directory node in the existing filesystem.

The contents of the mount point directory is inaccessible while another filesystem is mounted to it. Thus the directory should be one that is empty and reserved for this special purpose. The directory "/mnt", for instance, is available for this purpose.

Since the software detects no difference between a mounted filesystem and the "static" one, all normal file handling programs operate as usual on mounted filesystems. You can transfer files between mounted devices with the copy commands; you can change your working directory to any directory on the device. You can do backups (using "copy") to other filesystems, including floppy diskettes.

If you attempt to mount a filesystem that was not properly unmounted, the "mount" command rejects the request unless the "-oride" option is specified. The "-oride" option allows you to mount the filesystem, but you should repair it using the "fschk" command.

Typing mount, without any arguments, displays the currently mounted devices and directories on which they are mounted.

Options The options may be abbreviated by their first letter.

-oride This option overrides the mount flag if the filesystem had been left mounted.

-readonly Mounts the filesystem with read access only. You can not modify a read-only filesystem. The "mount" command will reject any attempt to create, delete, change the owner or change in any way, any file or directory in the filesystem. If a floppy diskette is write protected, this option must be used to mount its filesystem.

Examples @> mount /dev/fd0 /mnt

The filesystem on device "/dev/fd0" is mounted onto the directory "/mnt."

Format

1. @>move -i {-r} {-setowner} {-v}

   prompts for source and destination file names.

2. move {-q} {-r} {-setowner} {-v} source_file destination_file

   moves a single file giving it the name "destination_file".

3. move {-q} {-r} {-setowner} {-v} file1 ... file2 directory

   moves multiple files to the destination directory.

Description

The "move" command renames a file by changing the source file name to the destination file name. The source file name is deleted and the destination file name is either created or, if it already exists, is overwritten.

If the move crosses filesystem boundaries, then it is implemented as if the following sequence of commands had been given:

```
@>copy file1 file2; delete file1
```

If, however, the destination filesystem has no space, the file is not deleted, although the destination entry might be partially created.

If the second argument is a directory, the new file will have the same name in the new directory as in the old.

To move multiple source files to a target directory, list the source files on the command line followed by the name of the target directory. Move verifies that the last named file is a directory before attempting the move, and reports an error if it is not.

Options

-i Interactive. Causes move to prompt:

From:
To:

until you type:

iCTRLi + iCi

at any time, or either:

!CTRLi + iDi

or

iRETURN!

alone at a prompt.

-q Query. Request confirmation before moving each file (type "y" or "yes" and press RETURN to do the move, anything else to skip it).

-r Recursive. If the source file is a directory, the option creates the target directory, and recursively moves all files and subdirectories in the source directory.

-setowner Sets owner. Attempts to retain the original user id. Only the system administrator can do this.

-v Verbose. Identifies each file as it is moved.

Examples

1) @> move alpha beta.

This command line changes the name of the file from alpha to beta. The file remains in the same directory.

2) @> move /usr/jeff/games/newgame /games

This command line moves the file newgame from the directory "/usr/jeff/games" to the directory "/games." The name of the file is unchanged.

Notes

If the new name already exists in the destination directory, move first tries to delete it.

Format @>p {length=#} {width=#} {-silent} {-raw} {file1...filen}

Description The "p" command reads its argument files or, if no file names are given, it reads from standard input (the keyboard), and displays them on standard out (the terminal screen). It paginates, stopping at the end of each screen with the name of the file, the percentage of the file so far displayed, and the question "MORE?" The recognized responses to the prompt are described in the following table.

Table 2-5. Pagination Prompt Responses

RESPONSE	MEANING
y, Y, RETURN, SPACE	Yes. The next screen of text is displayed.
h, H	Half. The next half screen of text is !displayed.
q, Q	Quarter. The next quarter screen of text lis displayed.
1, L	Line. The next line of text is displayed.
1-9	iLines. The next 1-9 lines are displayed.
n, N	No. No more of this file is displayed. iGo on to the next file.
s, S	Stop. Exits the program.
other	Any other response causes the terminal to beep.

At the end of file, p moves to the next file, and prompts the user for more.

The "p" command also interprets unprintable characters, such as control characters, by printing "^" followed by a transformed version of the character, for example "^C". Note that spaces and tabs (which are defined at every eighth column) are treated normally. When lines exceed the specified width, they are wrapped around and continued on the next line of the screen.

Options and Parameters

All options can be abbreviated by their first letter.

length= Sets the number of lines on a page. The default is 22.

width= Sets the number of characters in a line. The default is 79.

-raw Raw mode. Control characters are not expanded.

-silent No prompts are given to the user. Everything is printed. This is used when the output is being re-directed into a file.

Examples

1) @>p /usr/fred/sample

This command line paginates and displays the file w/usr/fred/sample" on the screen.

2) @>p -silent filename i lpr

This command line prints a "hard" copy of the file "filename" in the current working directory ("|lpr" pipes the file to the "lpr" command for printing). There is no pause or message displayed between pages and control characters are filtered out.

@>ls / | p

This command line paginates the listing of the contents of the root directory. Execution of the "Is" command is piped through the "p" command.

Notes

If you type "p" with no arguments and do not redirect its standard input, "p" will read from standard input. It will appear that the system is not responding. To exit this situation, press CTRL and either D or C at the same time, or press RETURN several times, until the MORE? prompt is displayed, and then press N.

If the eighth bit of a character is "on" on your terminal, the tilde character, "~", is displayed before the character, e.g., "~C". If the character has the eighth bit "on" and is also considered a control character, then the tilde will precede the escape control character sequence, e.g., ~^C.

Format @>pmask

Description The "pmask" command prints the current file creation mode mask. This mask determines the actual modes of a created file.

Examples 1) @> pmask
rwx rwx

All modes are currently allowed.

Notes To change the mask, use the "setmask" command.

Format @>pr {options} file1 ... filen

Description The "pr" command produces a formatted version of one or more files on standard output. The output is separated into pages headed by the date, the name of the file and the page number.

The options apply only to files following the option on the command line, and can be reset between files.

Options

-b	Brief. Prints one line each of the header and footer. Counting blank lines, only six lines are subtracted from the total page size.
c=author	Places "Copyright (c) year author" in the footer.
-F	Places a formfeed after the file.
f1=footer	First line of footer.
f2=footer	Second line of footer.
f3=footer	Third line of footer.
h=argument	Uses the specified argument as the header, instead of the file name. (If you use "pr" as a filter, (e.g. "Is i pr"), it will not have a name unless you use the "h=" option.)
l=#	Takes page length to be the number of lines instead of the default 66.
-m	Prints all specified files simultaneously in separate columns.
-#	Produces numbered column output of each file sequentially. Any number between 1 and 69 can be specified. Columns are produced for the page length ("l=").
+#	Begins printing with page number.
-nb	No brief. Prints three lines of the header and footer. Counting blank lines, a total of ten lines are printed. This is the default.
-nh	No header or footer.
s=c	Separates columns by zero, one or more characters instead of white space. To specify no characters, types s= "". The default character is TAB. Do not use control characters as delimiters.
w=#	Takes the page width to be the number instead of the default 132.

Examples

```
@> ls | pr w=80 -5 l=1 -nh
```

Produces a five-column listing of the current working directory's files.

Notes

When using "pr" to display text on the screen, specify the width as 80, and the length as 24.

If you use arguments that consist of several words, enclose them in double quotes.

Format @>prompt {character} {message}

Description The "prompt" command causes the process to wait until a key is pressed. The "prompt" command is useful in building command files for which it is basically an "if" statement you can use with the command interpreter. In a command file, "prompt" can be used to wait for a key to be pressed.

If a character is specified, prompt acts like a condition. If the specified character is pressed, execution of the command file ends. If any other character is pressed, the remaining commands are processed.

If "prompt" is used without a character, no message is printed and execution of the command file continues.

If both a character and a message are specified, the character must precede string. The message is displayed if the specified character is press. A string is specified only if a character is specified also.

Examples 1) In a command file:

```
echo -nnl "Hit the ESC key to exit, any other key to
continue"
prompt ^[ "Exiting now..."
```

If the ESC key is pressed, the message "Exiting now..." is displayed and the command file is exited. If any other key is pressed, the command file continues execution. (See "ved" for instructions of entering unprintable characters in a file.)

Format @>ps {options} {parameter=value}

Description The "ps" command displays information about the currently active processes. The following information is given:

PID The process identification

USER The name of the user who initiated the process

FLG A three character code indicating the state of the process. The codes are in hexadecimal, and may be added. The codes for each position are as follows:

8-- uninterruptable wait
4-- three segment storage format
2-- defer signals and eventcalls
1-- locked in memory by user
-8- being traced
-4- being debugged
-2- interruptable wait
-1- locked in memory by kernel
--8 raw i/o
--4 kernel process
--1 swapped out

STAT The state of the process:

WAIT	process waiting for an event
RUN	process executing
RDY	process waiting in the run queue
STOP	process stopped
SSPD	process suspended
EXIT	process has exited

PR Current priority. Priorities range from 0 to 8. The greater the number, the higher the priority. Processes with the same priority are dispatched on a first-in-first-out basis.

0	background processes
1 - 5	interactive processes
6 - 8	system processes and interactive processes holding system resources

WAITFOR What the process is waiting for. Recognizes file locks as wait events. (Prints "lock".) Locking changes the size of file descriptors.

Kb The size in Kbytes of the core image of the process

SYS	The amount of system time used in seconds and tenths
USER	The amount of user time used in seconds and tenths
PARENT	If the process has a parent, then the parent's address and process identification are displayed (-long)
PROGRAM	Program name with its arguments, "-" means command interpreter, "=" means the user has executed "admin".
text	Beginning and ending physical address of the text segment (-loc)
data	Beginning and ending physical address of the data segment (-loc)
stack	Beginning and ending physical address of the stack segment (-loc)
MAX	Ceiling (maximum) priority (-long)
MIN	Floor (minimum) priority (-long)
CHILD	Child system time in seconds and tenths (-long)
TIMES	Chile user time in seconds and tenths (-long)
AGE	Time in seconds and tenths that a process has been on its present queue (-long)
LU	Number of open logical units (-long)
INT	Outstanding process interrupts (-long)

80	The process has completed it time quantum
40	The process will exit
20	The process will suspend
10	The process wil be preempted
8	Flag used by kernel tracing routines

SIG	Outstanding signals. BOSS/IX signal numbers correspond to bits in this word (-long)

If no names are specified, "ps" prints the process status of the user. If names are specified, "ps" prints the status of those users.

Options	-all	Prints out the status of all the programs on the system.
	-loc	Prints the beginning and ending locations in memory of the text, stack and data sections. Prints the addresses in hexadecimal.
	-long	Prints out the long form, all information including the complete command.
	pid=#	Prints the process status of a specified process. It will only accept one process id.
	-pri	Prints the floor and ceiling priorities as well as the current run priority.
	-sys	Prints only the system processes (kernel processes). This does not list caller's processes by default (if no names given).
	user names	Prints the process status of the specified user(s)

Examples 1) See figure below.

```
@>ps -a
 PID USER   FLG STAT PRI WAITFOR Kb  SYS  USER PROGRAM
   1 admin  C21 WAIT  99 child   45  5.1  0.5 /etc/start
  75 dennis C20 WAIT  99 child   54  1.6  0.3 - (/dev/tty2) /bin/command
  21 admin  C20 WAIT  99 event   65  1.1  0.1 /sys/lpd lp
  12 admin  420 WAIT  99 timeout  4  1.6  0.0 /* system update program */
  14 admin  C21 WAIT  99 34A0E   35  0.4  0.0 /* system error logger */
  16 admin  420 WAIT  99 3D940   35  1.6  0.1 /lan/lan_rsm
  17 admin  421 WAIT  99 3D8F0   82  0.2  0.0 /lan/lan_evtmtor
  22 admin  421 WAIT  99 timeout 31  0.6  0.0 exec -w t=/dev/tty4 /bin/logi
  23 admin  421 WAIT  99 timeout 31  0.5  0.0 exec -w t=/dev/tty4 /bin/logi
  24 bruce  C20 WAIT  99 child   53  1.5  0.4 - (/dev/ttyO) /bin/command
  92 bruce  400 RUN   67         35  1.4  0.5 ps -a
  88 dennis 420 WAIT  99 tty     83  2.2  1.9 ved edit.data

@>
```

Figure 2-4. Processes Display

Format @>pted

Description The "pted" command is the printer translation file editor. It allows a way of viewing and changing the printer translation tables. These tables are stored in the file "/etc/ptrans." Table names consist of two alphanumeric characters, a-z, A-Z, 0-9.

Upon execution, the command displays the menu shown in the following figure.

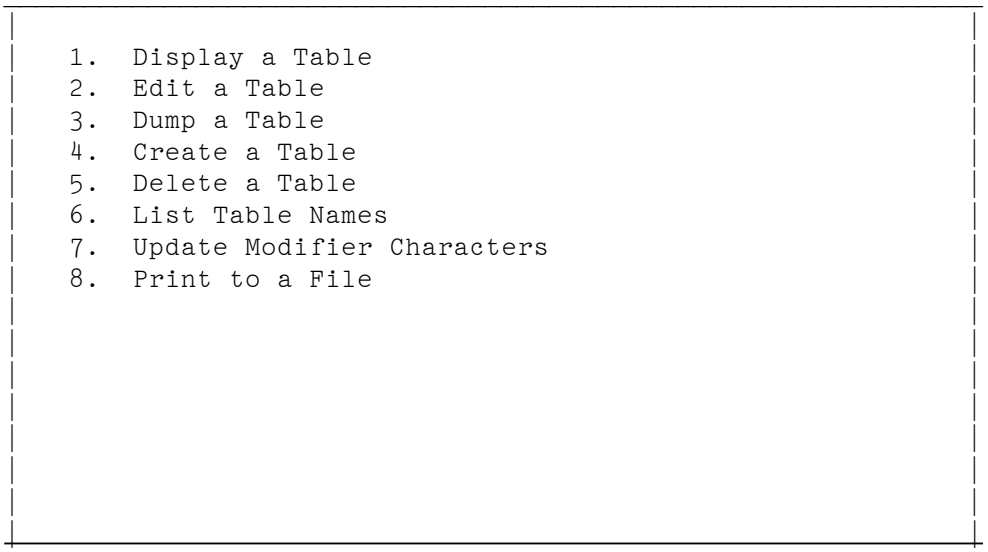

Figure 2-5. Printer Translation File Editor Menu

Display a Table This function displays the contents of a specified table. The table is arranged as a matrix with the character positions given in hexadecimal notation. The numbers running across the top are the high order bits and the numbers running down the left are the low order bits. The characters are displayed as ASCII characters. The modifier characters may also be displayed.

Edit a Table

This function displays the contents of a specified file and then allows you to modify it.

To change the character code at any position, move the cursor to that position and write over the current code. The control sequences used to move the cursor are shown in the following table.

Table 2-6. "pted" Cursor Movement Control Sequences

SEQUENCE	MOVE
CTRL+N	moves the cursor down one line
CTRL+P	moves the cursor up one line
CTRL+F	moves the cursor one character to the right
CTRL+H	moves the cursor one character to the left.

Characters can be entered in any of these four ways:

1. To enter a printable character, type the character in the desired position. Do not press RETURN.

2. To enter any character in hexadecimal notation, press CTL-II and then type the two character hexadecimal code for the character.

3. To enter a vertical displacement code and a character, press CTL-III, type the vertical displacement code, and then type the character code. When you have entered these codes, press RETURN to end the entry.

4. To enter a modifier code and a character, press CTL-IV, type the modifier code, and then the character code. When you have entered these codes, press RETURN to end the entry.

Dump a Table — This function displays a specified table in hexadecimal notation. The modifier characters may also be displayed.

Create a Table — This function makes a new table. The table name must be two characters. You may use an existing table as the template for the values in the new table, or use the default values. You also have the choice of creating a 7-bit or 8-bit table.

This function only creates the table. If you want to change any of the values, you must then use the Modify a Table function.

Delete a Table — This function deletes a specified table. You are asked to verify that you want the table deleted before the program deletes it.

List Table — This function lists the names of the printer translation tables defined on the system.

Update Modifier Characters. — This function allows you to edit the modifier characters in a particular table or in the default table. If you are editing the default table, you may only add modifier characters. If you are editing any other characters, only the first character may be changed.

When you have changed a character, you are asked if you want the change saved. When you have answered, the program either saves or abandons the change, and redisplays the menu.

Print to a File — This function writes the table to a file in either hexidecimal or ASCII form.

Format @>pwd {-all}

Description The "pwd" command prints the full directory path name of your current working directory.

Options -all Print both current and alternate search directories. The option may be abbreviated as "-a".

Examples

```
@> pwd
/usr/john/doc
```

"/usr/john/doc" is the name of the current directory.

Notes Alternate directories are added with the "ad" command.

Format @>remark

Description The "remark" command tells the command interperter not to process the arguments. This command is useful for documentation remarks in command files.

Examples @> remark "Hello world"

Format @>resume pid#

Description The "resume" command continues execution of a process which has been suspended, either intentionally, with the "suspend" command, or through an error in the program.

Notes

Resuming a process from the command level causes the command interpreter to wait for it to complete.

Resuming a process does not resume its suspended child processes.

Format @>setmask {+-}{rwx} {+-}{rwx}

Description The "setmask" command sets the file creation mode mask. This mask is used to determine access modes whenever a process creates a file.

Options

+	When used with a "r", "w" or "x", adds that mode.
-	When used with a "r", "w" or "x", subtracts that mode..
r,w,x	Without + or - assigns that mode alone.

Examples @> setmask rwx rx

Sets the mask to allow read (r), write (w) and execute (x) access to the file owner, and read and execute access only to all other users.

Notes The "setmask" command does not affect the modes of an already existing file.

The "setmask" command is internal to the command interpreter, so that it affects the mask of the command interpreter and any commands invoked by it.

The default mask is: rwx rwx.

The actual modes of the created file are the logical AND of the mask and the requested modes.

Format @>shutdown minutes {reason}

Description The "shutdown" command puts the system in single user mode. It signals the system control to terminate all extant processes. Only the system administrator can shutdown the system.

The number following the command determines when the shutdown occurs. Shutdown occurs 15 seconds after the number of minutes specified. For example, 0 brings the system down in 15 seconds, 1 brings it down in 1 minute and 15 seconds, etc.

If you give a reason, it will appear on all terminals with each warning message. The warning message is displayed every minute and then 15 seconds before the system shuts down.

When "shutdown" is executed, it kills all processes except the system control process, and prints the message:

<single user mode>

followed by the prompt "ADMIN>" indicating that the system is now in single user mode and the system administrator is the logged on user. To finish the shutdown procedure, type:

|CTRL|+|D|

at the prompt. This message is then displayed:

single, multi, or shutdown?

To shut down the system, type "shutdown" and press RETURN.

Examples 1) @> shutdown 5 for the evening

This command shuts down the system with five minutes of warning, and displays the reason.

Notes Pressing:

|CTRL|+|C|

up to the last possible moment at the terminal originating the shutdown, will abort the shutdown.

Files Used /etc/ports To find out what terminals are on the system.

/dev/tty* To write a message to each terminal.

Format @>sort {options} {parameter=value{aid}{n}} file1 ... filen

Description The "sort" command orders all the lines in the specified files according to the options specified. The result is then written to the standard output device or to a file specified with the "o=" option.

Options

-b	Ignores leading blanks when comparing records.
bs=	This parameter specifies the size of the internal buffer. The default size is 5000 bytes. A larger buffer makes for a faster sort. If the buffer size is too small for the sort, sort enlarges it.
-c	All character comparisons during the sort are case insensitive, treating upper and lower case characters as identical.
f=	This parameter specifies the fields to sort. The field separator used is specified by the "d=" option. The modifier "a" specifies an ascending sort, the modifier "d" specifies a descending sort. The modifier "n" specifies a numeric sort. Several fields can be sorted individually by separating the field specifications with commas.
d=	This parameter specifies a single character as the field separator, which may be any single character. The default field separator is a TAB.
-ns	Non-stable sort. The original order of identical items is not necessarity maintained.
-n	Sorts based on the first numeric characters in the records. Decimal points are recognized as part of a numeric expression.
o=	This parameter specifies the output file.
-r	Sorts the file in reverse ASCII order.

Examples 1) @>sort d=: f=1a /etc/passwd

This command line will display the user information file in alphabetic order by user identification.

Format @>space {-nh} dev1...devn

Description The "space" command reports the amount of space (in blocks) that is free out of the total on each device specified.

Options -nh This option suppresses the printing of headers in the display.

Examples @> space /dev/boot

Blocks					File Descriptors				
Free	Used	Total	High	Used	Free	Used	Total	Used	Device
315	3961	4096	4066	96%	318	258	576	44%	/dev/boot

This command line displays the amount of space available on the boot partition.

Notes The program executes "sync" to update the filesystem before it reads the allocation bit maps.

Using the "raw device" form is slightly faster.

Format @>suspend pid1 ... pidn

Description The "suspend" command stops the execution of the specified processes. A suspended process can then be continued, with the "resume" command, or terminated, with the "kill" command,

Examples 1)
```
@>/a/long/program &
213
.
.
.
@>suspend 213
213 suspended
```

This command line suspends execution of the program which is running in the background.

Format @>sync

Description The "sync" command causes all information in the system's disk buffers (cache) to be written out to the disk. You should use it before executing such processes such as halting the system and copying or dumping the contents of the disks, which depend on all system buffers being flushed.

Notes The shutdown and unmount commands automatically execute a "sync".

In multiuser mode, the system runs the program "/sys/update" to execute a sync every 30 seconds.

Format @>sysinfo {options}

Description The "sysinfo" command prints on standard out (the terminal screen) the system serial number, the booted OS version level, and the initial system configuration. All or selected parts of the report may be printed. The default is that all of the information is displayed.

Options All options may be abbreviated by their first letter.

-ssn Displays the system serial number.

-version Displays the booted OS version level.

-conf Displays the initial configuration record.

-all Displays all information (-ssn, -version, -conf).

-raw Displays the initial configuration record in its hexadecimal format.

Example
```
@>sysinfo
 System Serial Number: 2000-90045

 Booted O.S. Version: EOS7118C, BOSS/IX release 7.1A (Aug 22,
 1984 14:15)

 Initial System Configuration:
 SSN: egl00045
 Device    Type Unit Baud [Parity Stop Data] Flow-control  (type)
 -----     ---- ---- ----  ------ ---- ----  ------------  ------
 Boot:      WD    0    50     OFF   1    7   XON/XOFF only
 Console:   SCC   0 19200     ODD   1    7   XON/XOFF only  EVDT
 Download:  SCC   1 19200     ODD   1    7   XON/XOFF only
```

Format @>ttymntbl {table {-g}} {-default}

Description The "ttymntbl" command loads or sets the specified terminal mnemonic table according to the specified options.

Options

-default Sets the system default terminal mnemonic table. All further mnemonic processing uses this table.

Examples

1) @>ttymntbl edt

This command line loads the mnemonic file for edt terminals for local use.

2) @>ttymntbl -default

This command line resets the terminal's mnemonic control to the system default table.

Format e>ttymodes {options}

Description The "ttymodes" command displays or sets the modes on the device associated with standard input, which in most cases is your terminal.

The modes deal with input, echo and output characteristics of the terminal. Characters are divided into two major classes: alphanumeric, or printable ASCII characters, and control characters. The alpha characters are those in the range of "space" (hex 20) through ""*" (hex 7F) inclusive, and hex AO through hex DF, inclusive. The control characters are those in the ranges 000 to 037 and 0177 to 0377. Special meanings are assigned to characters below 040 and DEL, 0177, when it is the erase character. The remaining control characters are defined as NORMAL and have no individual interpretation applied to them.

To display the modes currently set on the terminal, type "ttymodes" without any arguments and press RETURN. To alter the modes, type the mode either with or without the preceding caret ("^"). The caret preceding a mode means that the mode is turned off.

Options

-default This switch sets all modes to the standard initial state. This state is shown in the example below. Note that "erase" and "kill" keys are not affected.

-nowait Normally, the system waits until all previous output has been sent to the terminal before changing modes. The "-nowait" mode forces the system to change the modes immediately.

erase=char The erase character is changed to the character specified. The character can be entered with three different notations to accommodate other meanings that some characters may have: as a number (decimal, unless it begins with Ox for hexadecimal or 0 for octal), as a caret followed by a letter (just like the character would be echoed e.g. ^H for backspace), or as the character itself.

linekill=char The line kill character is changed by this option. The three forms described above are allowed for specifying the new linekill character.

Modes — The tty modes are decsribed on the following pages.

Input Modes

irawedit — When on, permits the input-editing characters to be read as normal characters; otherwise these characters have the effect of deleting previously typed characters (the erase and line-kill functions) or of redisplaying a partial input line (CTRL + R).

wakealpha — Characters are normally kept in an intermediate buffer, which is unavailable to the user's programs, until the new line character <NL> is entered. Thus a line can be altered with the erase character or linekill before the data is committed with a new line, which is the normal "wakeup" character.

When on, wakealpha causes every printable ASCII character ('space' through octal 040 through 0176, inclusive) to be a wakeup character. When it is typed, the line so far is made immediately available to the user. Currently, wakealpha mode implies irawedit mode.

wakectl — This mode is similar to wakealpha but deals with all characters outside the printing ASCII set including ALL characters with the eighth bit on (see i8bit mode, below).

When on, wakectl causes any normal control character to be a wakeup character. Alpha characters are saved in the buffer until any control character is typed. Thus printing characters can still be deleted before they have been committed.

i8bit — (Ignored on 7.2A). This mode causes all bits including the eighth bit (parity) to be accepted on input. Characters with the eighth bit on are input. Characters with the eight bit on are treated as normal control characters for the purposes of echoing and wakeups.

Although this mode does not disable the special meaning of any control characters, the system only recognizes those charcters when the eighth bit is "off .

prctl — This mode enables recognition of process control characters (CTRL + C or CTRL + Y). When off, these characters are treated as normal control characters and thus have no system defined function.

	escape	When on, this mode enables escape processing as follows: after typing the ASCII <ESC> character, a dollar sign followed by a backspace is echoed placing the cursor on top of the dollar sign. The next character typed bypasses normal special character recognition and is echoed according to the echoing rules established with the ectl, erawctl and ealpha modes described below. The mode allows certain control characters to be entered literally. When off, <ESC> is treated as a normal control character.
	eof	When on, CTRL + D is mapped to mean end-of-file. It acts lkie a wakeup character but does not appear in the data. If there are any preceding characters (not wakeup characters) they become immediately available to the user's program. If there are no preceding characters, the program will receive a zero-length input which is considered an end-of-file. When the mode is off, it is treated as a normal control character.
	icrlf	When on, the ASCII carriage return character <CR> is mapped to the ASCII newline character (octal 012) on input. When off <CR> is entered as <CR> (octal 015).
Echoing Modes	ealpha	Echoing of alpha characters is controlled. (See wakealpha mode.) When on, this mode enables the alpha characters to be echoed. When off, no echoing of the alpha characters takes place.
	ectl	When on, this mode enables the control characters to be echoed as "^(LETTER)". When off, the erawctl mode is examined for echoing of NORMAL control characters. When on, this mode supersedes the erawctl mode, below.
	erawctl	When on, this mode enables the control characters to be echoed literally as typed. If both ectl and erawctl are off, no echoing of control characters takes place except for those that have special meaning.
	enobellerr	When off, this mode causes a <BEL> character (audible beep on most terminals) to be echoed in two cases: when the erase character is typed but there is no previous character in the buffer to erase (at the left margin) and when so many characters have been typed that the input buffer overflows.

	etab	When on, this mode enables the echoing of the <TAB> character. How it is echoed is determined by the oxtab mode.
	ecrlf	When on, the mode causes echoing of the carriage return character as a carriage return/line feed combination.
	prctlecho	When on, echos ESCAPE and to the terminal.
Output Modes	o8bit	(Ignored on 7.2A.) When on, this mode prevents masking out the eighth bit (parity) on output.
	octl	When on, this mode forces control characters to be printed as "^(LETTER)" on output, except for <TAB> and <NL>. When off, control characters are output as sent, except for <TAB> and <NL>.
	ocrlf	When on, this mode maps <NL> to carriage return/ line feed on output. When off, <NL> is output without a carriage return.
	oxtab	When on, this mode causes <TAB>, on output, to be expanded to an appropriate number of spaces, depending on column position, assuming stops every 8 columns. When off, <TAB> is output literally.
Additional Modes	mnemonics	When on, this mode enables mnemonic processing. For example, with mnemonics on, an ESCAPE CS will clear the screen. With mnemonics off, ESCAPE CS will not clear the screen, but will be printed on the terminal as an ESCAPE CS.
	mapesc	When on, maps the <ESC> key to CTRL + C.
	mapctlx	When on, after a ^X is read, the next read generates an error. Used for 13XX compatibility in BASIC.

Examples

1) @> ttymodes
erase=^H linekill=^U
^wakealpha ^wakectl ^i8bit prctl escape eof icrlf
ealpha ectl ^erawctl .etab ecrlf ^o8bit ^octl ocrlf oxtab
^irawedit ^enobellerr ^mapesc ^mapctlx prctlecho
^mnemonics

This command line displays the current tty modes set for the user's terminal.

2) @> ttymodes ^ealpha ^ectl ^erawctl ^etab ^ecrlf -default

This command line sets the modes to the defaults and turns off echoing.

3) @> ttymodes ealpha ectl ^erawctl etab ecrlf

This command line turns on the default echoing.

4) @ ttymodes -default < /dev/tty2

This comamnd changes the modes on another terminal.

Notes

The "ttymodes" command performs its changes on the device associated with standard input. It can, however, be redirected to affect other terminals (see examples).

Not all parameters recognized by ttymodes are necessarily supported by all serial devices on all machines. We have indicated which options are not yet in effect. If you attempt to use an unsupported option you will not be told of its failure.

Format @>ttyxlate {options} file

Description The "ttyxlate" command is used -to load, set, or reset I/O translation tables. An i/o translation table is one-to-one mapping of characters from one value to another. For example, one could define a translation table which takes lower case letters and translates them to upper case. Translation can take place on input, output, or both. By choosing the proper combination of options from the char below, no translation at all will be performed.

Options

-i Casues input translation to be set or reset.

-o Causes output translation to be set or reset.

-reset Resets specified table or tables.

Examples

1) @> ttyxlate -o Spanish

This command line uses the Spanish translation table for output.

2) @> ttyxlate -reset -i -o

This command line resets input and output translation.

Format @>unmount dirlidevicel {... dirnidevicen} {-oride}

Description The "unmount" command prevents any further access to the filesystem which was mounted on the specified directories or devices.

Within the command, "sync" is executed to assure that the device is in a consistent state, after which no further access is made. Unmount fails if the device is in use (which usually means that some user has a current directory on the mounted device).

If a filesystem is located on a removable medium, such as a diskette, the medium should only be removed after an "unmount" has been done.

Options The options may be abbreviated by their first letter.

-oride This option overrides mount errors, allowing for unconditional unmounting of a filesystem or directory.

Examples 1) @>unmount /mnt

This command line unmounts the filesystem mounted to directory /mnt.

Notes Only mounted filesystems that are not busy can be unmounted. That is, if any user's current or alternate working directory is located in a mounted filesystem, or if any user or process is accessing a file in a mounted filesystem, that filesystem cannot be unmounted.

Files Used /etc/mounttab To update list of currently mounted devices.

Format @>usb device {type=*}{desc=*}{-save}{-set}{-get}{-strip}

Description The "usb" command updates or creates the superblock for the specified device. A superblock contains certain information about that device that is needed by the device driver. Only the system administrator can update a superblock.

Options and parameters

type= This option specifies the name of device characteristics file in the directory "/etc/diskdesc" which contains device characteristics. This parameter is used when writing the information to the device or memory.

desc= This option specifies the device characteristics file in the directory "/etc/diskdesc" created by reading the superblock from either the memory or the device.

-set This option writes superblock to system memory.

-get This option reads superblock from system memory.

-save This option writes the superblock to device.

-strip This option disregards any existing partitions read from memory, device, or device characteristic file.

Examples 1) @>usb /dev/rwd0 type=wd0 -save -set

This command line reads device file "wd0" and writes the superblock to both system memeory and rwd0.

2) @>usb /dev/rwdo -get -save

This command line reads superblock from system memory and writes it to rwd0.

Format @>vconf file {options} {parameter=value}

Description The "vconf" command reads and writes configuration files. It allows changes to configuration files to be done quickly and easily. If a configuration file is saved, the old file name is moved to file name.bak.

Options

root=MMmm	Root device; MM is device major number, mm is device minor number. These numbers should be in hexadecimal.
swap=MMmm	Swap device number.
size=	Size of swapper in blocks.
ram=	RAM disk size in blocks.
system=	A string specifying the system name.
-save	Save configuration information to "file."
-silent	Display no messages on boot.
-verbose	Display normal messages on boot.
-os	Read in the configuration file of the operating system that is currently being used.
-oride	Override error. s "directory system left mounted"
-noride	Does not override "directory system left mounted" error.
buffers=	Number of system buffers.
linekill=char	Set linekill character (key).
erase=string	Set erase character (key).
locks=	Sets maximum number of file locks.
mfsys=	Sets the maximum number of mounted directory systems.
-single	After completing the boot procedure, the system will be in single-user mode.
-multi	After completing the boot procedure, the system will be in multi-user mode.

-secure	The system administrator password is asked for before the boot process can be completed.
-nonsecure	The system administrator password is not asked for in order to complete the boot process.
printers=	Sets the maximum number of printers allowed.
sockets=	Sets the number of "well known" LAN sockets.
dsockets=	Sets the number of "dynamic" LAN sockets.
lanbuffers=	Sets the number of LAN buffers.
lus=	Sets the maximum number of logical units on the system.
opens=	Sets the maximum number of open files allowed on the system.
eventcalls=	Sets the maximum number of eventcalls allowed on the system.
tsegs=	Sets the maximum number of shared text segments.
procs=	Sets the maximum number of processes that can simultaneously exist on the system.
-dump	Enables the crash dump.
-nodump	Disables the crash dump.
-debug	System will go to the O.S. debugger on a system crash or when the reset button is pressed.
-nodebug	System will not go to the O.S. debugger on a system crash.
ibsize=	TTY input buffer size.
tbsize=	TTY type ahead buffer size.

Examples

1) @>vconf /etc/conf root=1403 swap=/dev/swap size=2000 -save

This command line alters the root, swap and swapsize in "/etc/conf." Note here that "/etc/conf" must already exist.

2) @>vconf -os /etc/current -save

This command line reads the current operating system's configuration file and saves it under the name "/etc/current."

Note

An existing configuration file is needed if the "-os" option is not used.

If the "-save" option is not used, the changes in the configuration file will not be saved.

Format @>ved {-readonly} {-vedhelp} {file/name}

Description The "ved" command calls the BOSS/IX screen-oriented text editor. If you do not name a file, "ved" will assume you want to edit the last file you edited in your current directory. When you edit a file for the first time in an editing session, specify the file name.

Options The options may be abbreviated by their first letter.

-readonly During the editing session, you can only read the files. This is the "look-only" mode and since the file is not altered, the file's modification date remains unchanged.

-vedhelp The editor is called in read only mode with the on-line help file.

Editor Commands All basic "ved" commands are control characters (the control key plus a character) or escape characters (the ESCAPE key and then another key). In the following description,

|CTRL|

represents the control key, and

|ESC|

represents the escape key. The combination

|CTRL| + |X|

is a single keystroke, while

|ESC| |X|

are two keystrokes. If the command takes an argument, the argument is terminated with a carriage return.

A command may be modified in three ways:

1. Pressing the ESC key before the command will change it to an alternate form of the command, usually a related function.

2. Numbers are entered by typing:

   |ESC| ## |RETURN|

   When they precede a command, either a control character or an escape control character, the command uses the number.

3. The ALT mode command,

   |CTRL| + |l|

   inserts a mark in the text usually at the current cursor position. You can precede any command that moves the cursor with the ALT command; "ved" drops the mark before moving the cursor.

Line One

When you execute "ved", the named file is displayed on the screen and the top line on the screen contains information about the editing session. This first line is divided into the following five fields:

Field 1 Reserved for system messages, and questions. All confirmation is done here.

Field 2 Contains the current argument to the command. It is reset to 1 after every command and can be set by the user.

Field 3 The current take buffer. Default is "default", however, it too can be set by the user.

Field 4 The name of the file being edited.

Field 5 Error field. All warning messages appear here.

Command Summary

Table 2-7 summarizes the "ved" commands. The character key in the left column must be pressed while holding down the key or keys shown in the top row. A dot indicates that the combination has no command value.

A more detailed explanation of editing commands, grouped according to function, is contained in Section 6 of the MAI 2000 User Guide, "Text Editor."

Table 2-7. "ved" Command Summary

	|CTRL|+CHAR	|ESC| then |CTRL|+CHAR	ALT then |CTRL|+CHAR	ALT then |ESC| then |CTRL|+CHAR
A	start of line	start of paragr	start of line	start of paragr
B	beginning	bottom	beginning	bottom
C	exit	backup file		
D	delete char	delete words		
E	end of line	end of paragraph	end of line	end of paragraph
F	forward char	forward word	forward char	forward word
G	get file	change file	edit macro file	.
H	back char	back word	back char	back word
I	tab	.	tab	.
J	save char	save word	jump to mark	switch mark and cursor
K	kill line	kill paragraph	kill selection	
L	adjust window	re-display	.	.
M	carriage return	change modes	sets mark	.
N	down line	down paragraph	down line	down paragraph
O	open line	.	open line	.
P	up line	up paragraph	up line	up paragraph
Q	XON	.	.	.
R	forward search	reverse search	forward search	reverse search
S	XOFF	.	.	.
T	save line	save paragraph	save selection	save selection
U	#=mult	mult=1	.	.
V	paste	paste and clear buffer	replace select with buffer	replace select clear buffer
W	write file	write take buffer	write selection	
X	execute command	execute buffer	exec selection	
Y	suspend	help	.	.
Z	re-search	reverse search direction	re-search	reverse search direction
@	position from beginning top	position from end bottom	position of mark from beginning from top	position of mark from end from bottom
[	ESC	.	.	.
]	ALT	ALT ESC		
\	change buffer	clear pres buffer		
^	quote char			
_	undo delete		undo mark	
DEL	erase char	erase word		

Files Used

/tmp/vedB.pid#	Backwards delete take buffer.
/tmp/vedD.pid#	Delete buffer.
/tmp/vedT.pid#	Take buffer.
/tmp/vedX.pid#	Command buffer.
/tmp/ved.default.pid#	Default buffer.
/doc/comds/ved.help	On line help file.
.EDTMP.#	Stores the name of the last file edited in this directory by user number #

Format wait {pid1...pidn}

Description The "wait" command causes the command interpreter to wait for child processes pidl ... pidn to exit before accepting another command. If no pid is specified, the command interpreter waits for the first child that exits.

Examples 1) @>wait 115

This command causes the command interpreter to wait for process 115 to exit.

2) @> wait

This command causes the command interpreter to Wait for any process to exit.

Notes An attempt to wait for a process that is not a descendant of the command interpreter executing the "wait" causes the command interpreter to wait for all child processes (if any) and to print the message "no children".

Format @>who

Description The "who" command lists logged in users by terminal id and provides time of last login.

Examples
```
1)   @> who
 john     ttyl Jun 24 08:58:29
 sue      tty8 Jun 23 08:43:17
```

Files Used /tmp/logins To find out who is on the system.

Format @>write user(s) {-all} {-silent} {-noheader} {-oneheader} {message}

Description The "write" command displays a message on the specified user's terminal. If you specify several users, separate their names with commas.

If you do not specify a message, write prompts you for lines of text. It sends each line until you type:

|CTRL|+|D|

in response to the prompt. This allows you to send multi-line messages or have conversations.

Unless otherwise requested, write prefixes the sender's name and the time to all messages, and informs you if there are no recipients.

Options All options can be abbreviated to their first letter.

-noheader No header on the message. A message is sent without the sender and time header.

-oneheader One header on the message. This option is used when engaged in a conversation. The first line of the message has a header, all succeeding lines do not.

-silent Silent. Do not inform the sender if recipients were not logged on.

Examples 1) @>write jeff,toni What about lunch?

On jeff's and toni's screens, the following is displayed:

```
From dianne @ 12:45:09 What about lunch?
```

2) @> write jeff -noheader "What about lunch?"

On jeff's screen, the following is displayed:

```
What about lunch?
```

3) @> write jeff -oneheader
```
write> How about lunch?
write> I think it is about time.
```

On jeff's screen, the following is displayed:

```
From dianne@ 12:45:09 What about lunch?
I think it is about time.
```

MAI(R) 2000 TECHNICAL REFERENCE MANUAL

SECTION 4

SYSTEM INTERACTION TEST (SIT)

AUGUST, 1985

LIST OF FIGURES

LIST OF TABLES

THE SYSTEM INTERACTION TEST

This section describes the System Interaction Test (SIT) diagnostic and provides instructions for its use. It is contained on the user diagnostics MCS cartridge or floppy diskette, DIA, that was included with your system software.

SIT is designed to detect interaction problems occurring between competing devices (e.g. Magnetic Cartridge Streamer), controller boards (e.g. LAN and 4-way controllers), and memory. If problems are found, SIT also provides a method for isolating the source of the problem.

When you load SIT, the program performs a process of "autosizing." During this procedure, the program checks to see which devices and controllers are installed on the system. It then defines a system configuration for those devices and controllers.

The test proceeds by exercising the devices and controllers in a semi-random fashion, in an attempt to create as many conflicts and "worst-case" situations as possible. A part of the SIT program called the "sequencer" controls the exercising of the devices and controllers so that processes are started on them sometimes in random order, and sometimes in a determined order.

While the devices and controllers are being exercised, this test keeps track of any problems that occur, such as multiple interrupts, surges on the power supply and overlapping of memory accesses. Periodic displays provide a running report of the test results.

If a problem is determined to exist, a knowledgable user can employ the "SELECT" command to specify individual boards or groups of boards for further testing. Running tests on specific boards makes it possible to "narrow down" the possible sources of the problem.

A number of further commands allow a sophisticated user isolate the problem even further, to a portion of the fixed disk for example. The isolation procedure is illustrated in Figure 1-1.

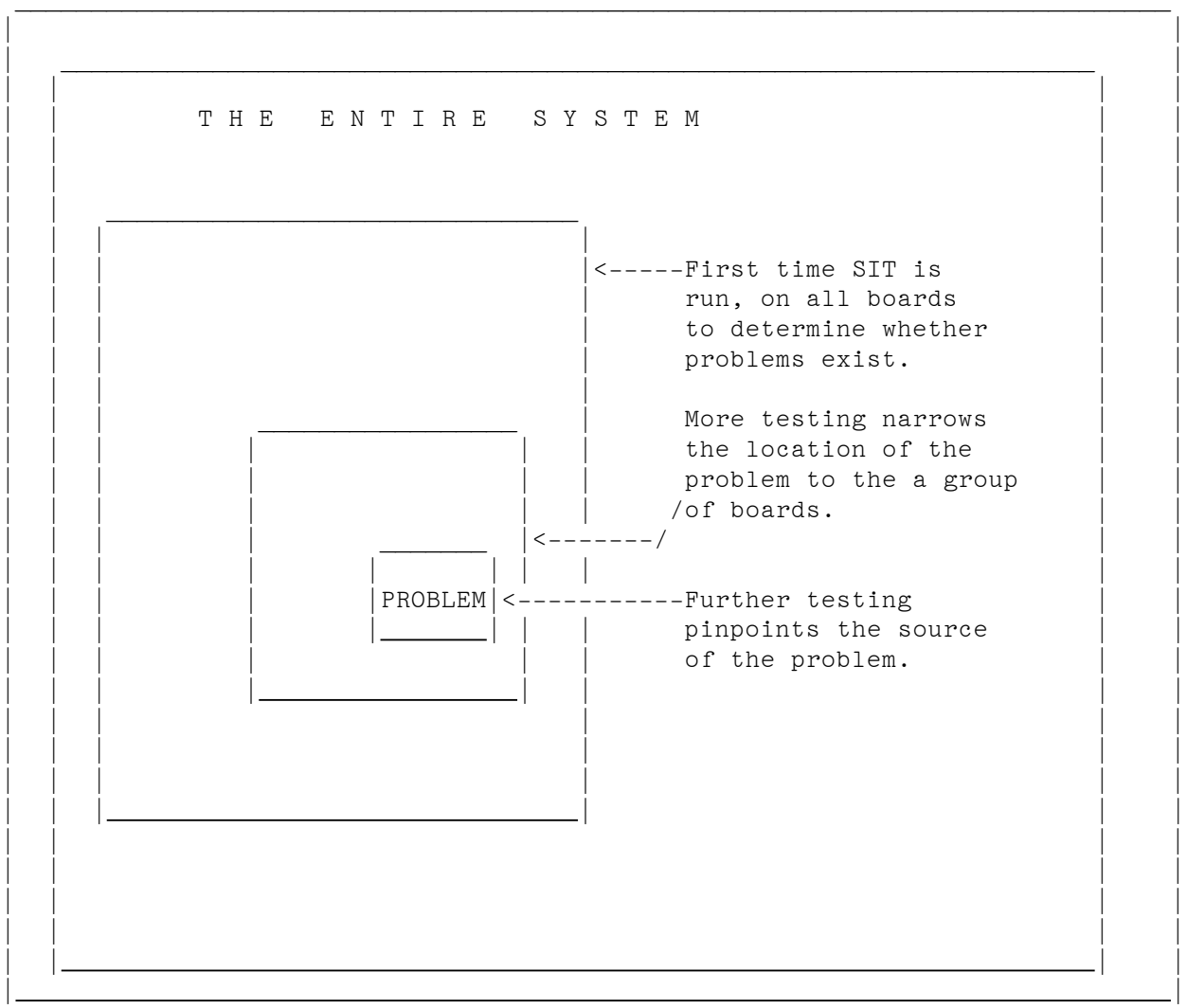

Figure 1-1. The Testing Procedure

WHEN TO USE SIT

There are several situations in which it is useful to run the SIT diagnostic program.

1. You may wish to run SIT periodically to check the state of Your system. Developing problems in the hardware can then be caught before they have a chance to cause serious damage and loss of data. This can be done periodically as part of a regular maintenance program.

2. If you are upgrading your operating system software, or re-installing the operating system, it is a good idea to run SIT before you do the installation. This will verify that the system hardware is in good working order prior to the installation.

3. If your system is having problems, the SIT is useful as a "trouble-shooting" (diagnostic) tool. It can help you to eliminate possible causes and to isolate the source of the problem. If you are using the SIT as a diagnostic, you should leave the interpretation of the results to a trained service representative.

HARDWARE REQUIREMENTS

In order to run the SIT, your 2000 must meet the following minimum configuration requirements:

1. A video display terminal configured on port 0. The terminal may be any of these types:

   - Ergonomic Video Display Terminal (EVDT)

   - Ergonomic Display Terminal (EDT)

   - 7270 VDT

2. A minimum of 512 KB of RAM (random access memory)

3. 1 floppy diskette drive or magnetic cartridge streamer drive

SOFTWARE REQUIREMENTS

The only software requirement to use the System Interaction Test is a floppy diskette or a MCS cartridge containing the bootable Diagnostic Executive and the System Interaction Test.

INTRODUCTION

In this chapter we describe the procedures for using the System Interaction Test. These procedures include booting the diagnostics tape or diskette, loading the SIT program, and executing the commands.

The SIT, together with the bootable Diagnostics Executive program and Logic Tests, is contained on the user diagnostics floppy diskette or MCS cartridge supplied with your system. It is labeled:

DIAXXXX

where "XXXX" is some four-digit number indicating its release level.

The flow of the procedures covered in this chapter is summarized in Figure 2-1.

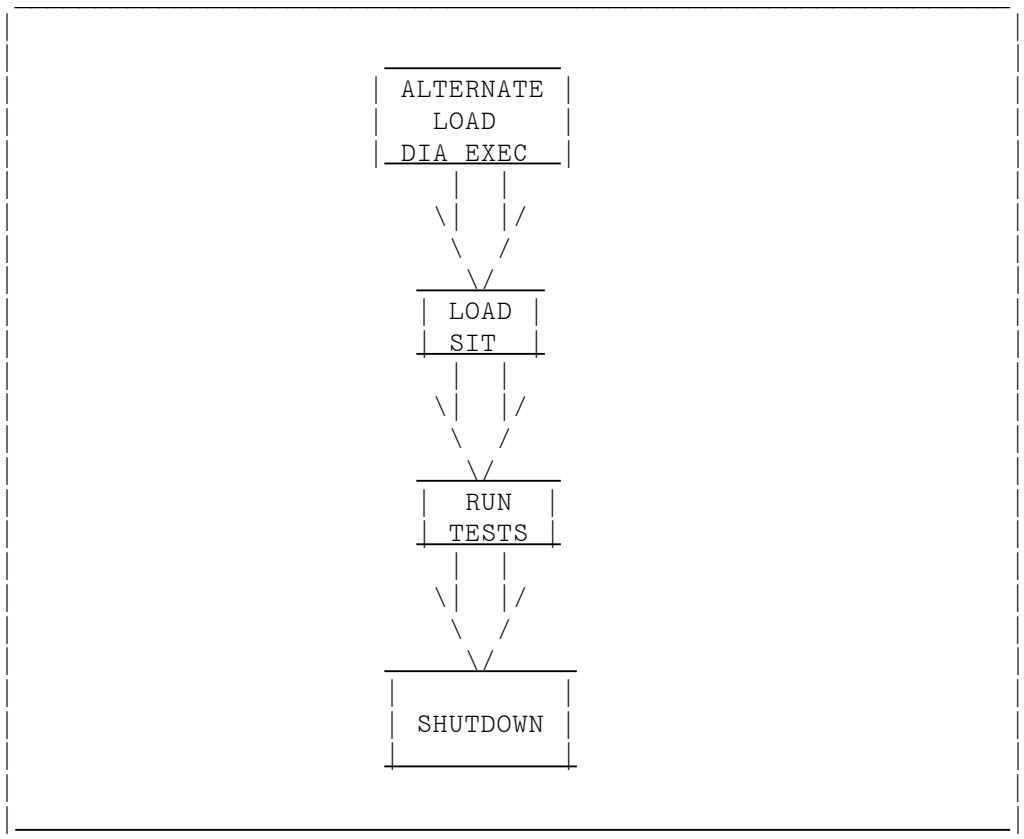

Figure 2-1. Flow of Procedures

ALTERNATE LOAD

To run the System Interaction Test, first you need to load the Diagnostics Executive. To do this, you need to perform an alternate system load. It is assumed that your system is either running or powered off.

If your system is currently running, perform a shutdown so the re-boot prompt is displayed:

```
Press 'RETURN' key to reboot ('^C'=alt-load,
  '^S'=self-test):
```

Press at the same time:

|CTRL| + |C|

This is what is meant by "^C in the prompt.

If the system is currently powered off, turn it on. When the self-test screen is displayed, press at the same time:

|CTRL| + |C|

Either of the above procedures specify that you are performing an alternate system load. The boot device prompt is now displayed:

```
Boot device:
```

Your user diagnostics came on either a floppy diskette or on an MCS cartridge.

If your diagnostics are on floppy, insert the diskette into the drive and press in the lock button. Then type "fd" and press RETURN.

If your diagnostics are on MCS cartridge, insert the cartridge into the MCS drive, slide in the tray, and turn the locking lever up. Then type "cs" and press RETURN.

The system file prompt is then displayed:

```
System file:
```

Press RETURN alone.

The system then performs the boot procedure.

When the boot is completed, the diagnostic executive prompt is displayed:

```
<exec>
```

You are now ready to load SIT.

LOADING SIT

Once the diagnostics executive has been loaded, you are ready to load the SIT program. At the diagnostics executive prompt:

```
<exec>
```

type:

```
LOAD SIT
```

in either upper or lower case, and then press:

|RETURN|

The diagnostics executive then begins loading the SIT program into memory. While it is loading the program, it displays several rows of dots on the screen. When it is finished loading, it displays:

```
SYSTEM INTERACTION TEST (SIT) Rev. XX, wait <= 30
seconds for auto sizing
```

The SIT program is now operating. It begins by searching for the devices that are ready to be used during the test. It then builds a configuration file for those devices for use during the test. This is what is meant by "auto sizing"

Note that SIT will configure a floppy drive for testing only if there is a diskette in the drive and the lock button is pressed in. Similarly, it will configure the MCS drive for testing only if there is a cartridge locked in the drive and the cartridge is not write-protected.

When SIT is finished auto sizing, it displays a report of the controllers and devices present and the status of each, as in Figure 2-2.

```
SYSTEM INTERACTION TEST (SIT) Rev. xx, wait <= 30 seconds for auto sizing

       SYSTEM CONFIGURATION TABLE

 cont  status   device  status    properties
 ----  ------   ------  ------    ----------
 ln1x  present  ln10    runnable  1 megahertz CSMA Local Area Network
 fwOx  present  fwOO    runnable  serial comm port-chan a
 fwOx  present  fw01    runnable  serial comm port-chan b
 fwOx  present  fw02    runnable  serial comm port-chan c
 fwOx  present  fw03    runnable  serial comm port-chan d
 fwlx  present  fw10    runnable  serial comm port-chan a
 fwlx  present  fw11    runnable  serial comm port-chan b
 fwlx  present  fw12    runnable  serial comm port-chan c
 fwlx  present  fw13    runnable  serial comm port-chan d
 mcOx  present  mcOO    cant run  streamer cartridge tape
 wdOx  present  wdOO    runnable  Hard Disk, #cyl=830, #heads=6
 fdOx  present  fdOO    runnable  double sided, double density, 5 1/4
 fdOx  present  fd01    cant run  double sided, double density, 5 1/4
 sccx  present  sccO    runnable  serial communications port
 sccx  present  scd     runnable  serial communications port

MEMORY MAP: SIT area=000000-034800,freespace=034800-0BFFF

<sit>
```

Figure 2-2. System Configuration Table

The column labeled "cont" lists the controller boards, the column labeled "device" lists the devices. The kind of board is indicated by a two or three letter designation. The number of the board is indicated by the first digit, and the number of the device on that board is indicated by the second digit. E.g., "In10" indicates the Local Area Networking controller board number 1 as the controller, and port 0 as the device, "fw12" is the four-way controller board number 1, and port number 2. The exceptions to this are the sccx ports, which reside on the Central Microprocessor Board (CMB).

Note that controller boards and devices are numbered 0,1,2,..., so that "0" is the first, "1" is the second, etc.

If the status of a device is "cant run", the device was found not to be ready to use, as in the case of a floppy drive that did not have a diskette properly in place.

Below the configuration table, the memory addresses are displayed where the SIT program is contained and the memory addresses that are free. Below that is the SIT prompt:

```
<sit>
```

USING SIT

Using SIT as a basic diagnostic is a simple matter of selecting tasks and running them. When you first start SIT, all tasks are selected, by default. You only need to enter the "RUN" command, and let it go. Later, you can stop tests to narrow down the possible source of the system's problems.

Some commands and options allow a very knowledgable user to place special conditions on certain tasks, but these should only be used by a trained service representative.

To display a list of the available commands, type "list" and press RETURN at the SIT prompt. "List" is actually a diagnostic executive command, but it works while SIT is running.

To execute a command, type the name of the command at the SIT prompt, together with a task name or option, as required, and then press the RETURN key. The commands may be entered in either upper or lower case.

RUN

The command that you are most likely to use is the "run" command. On the screen display, its format is described as:

```
run [[-]kill]
```

The expressions enclosed in brackets are options. To execute the command, type "run" and press RETURN at the SIT prompt. This starts running all of the tests currently selected.

CAUTION

THE "KILL" OPTION DESTROYS DATA ON FIXED DISK. This begins the test on the fixed disk with both read and write tests. This action destroys any data currently on the disk. The only recovery is by a complete installation.

SELECT

When you first start SIT, all tests are selected by default. These can be changed by using the "select" command.

While the tests are running, SIT reports any errors encountered and periodically displays a "news" summary of the test results.

During this time, the SIT prompt is not displayed. Consequently, you cannot process any other commands, and so cannot use "select". You can get the prompt back, without stopping the tests, by pressing the ESCAPE key. Now you are free to execute another command.

Type "select" at the SIT prompt and press RETURN. SIT then prompts you for each task that was found to be runnable during autosizing. For example:

```
fw03 (n=no,<cr>=yes)?
```

If you want to run the indicated task, "fw03" in the example, press RETURN. If you do not want to run this task, type "n" and press RETURN.

When you have responded to all of the prompts, the SIT prompt is displayed again. Type "run" and press RETURN to run the tasks you have selected.

STOP

While the "run" command is executing, it may be useful to stop a task while leaving the other tasks running. Stopping a task can be used to see if the given task is the source of a recurrent error.

To stop the task, type "stop", followed by the name of the task, and then press RETURN.

SLEEP

When you have stopped a task, you can resume the display of news reports with the "sleep" command. The SIT prompt is then not displayed, but the running report or errors and the news summaries are resumed.

To interrupt "sleep", press the ESCAPE key.

SHUTDOWN

When you are finished running SIT, use the "shutdown" command. This ends the execution of all tests and shuts down the system. The re-boot prompt is then displayed:

```
Press 'RETURN' key to reboot ('^C'=alt-load,
  '^S'=self-test):
```

You may return the system to normal operation or proceed with whatever other procedures you need to perform.

SUMMARY

The commands briefly described above are all you need to use SIT as a basic diagnostic. Again, these commands are:

RUN
SELECT
STOP
SLEEP
SHUTDOWN

These commands are fully explained in the next chapter, under "Basic Commands." The tasks that the commands control are also described there. Finally, the advanced commands and options are also described, under the heading "Advanced Commands." These you should only use under the direction of an authorized service representative.

INTRODUCTION

In this chapter, we describe the tasks available for execution by SIT and the commands that allow you to control their execution. In the next chapter, we discuss some strategies for running SIT and some help in interpreting the results of the tests.

TASKS

The System Interaction Test uses "tasks" to check the integrity of various parts of the 2000 system. There is a task associated with each controller/device unit of the system. A task is essentially an exercise of the unit, consisting of writing and/or reading data to the device. For instance, the MCS task writes some data to the tape, reads the data back, then compares what was written and was read back. Some tasks only write, others only read, though most both read and write. The procedure is illustrated in Figure 3-1.

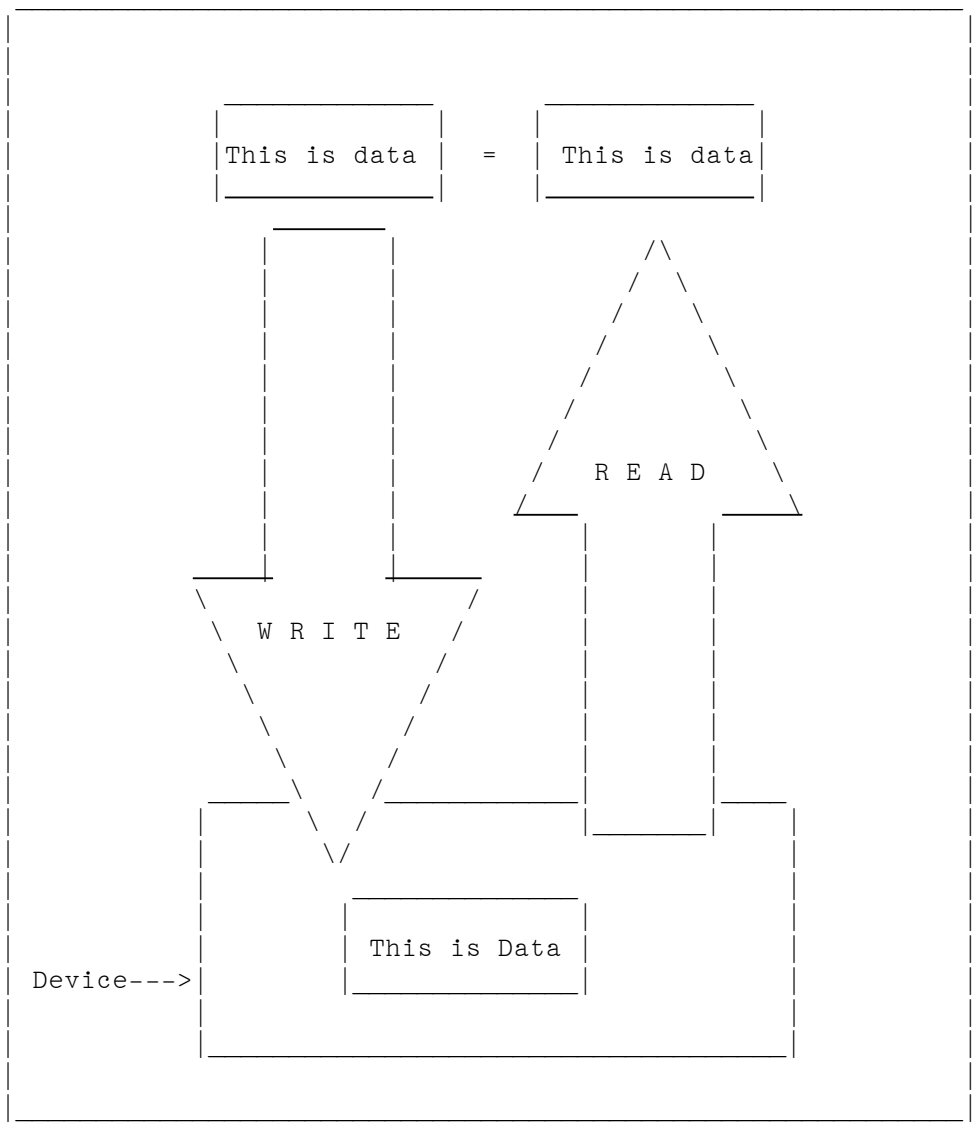

Figure 3-1. One Pass of a Task

When more than one task is executed at the same time, conflicts between the various devices, which can give rise to system errors, can be monitored. The SIT sequencer controls the order in which the cycles of each task are started in order to generate both a wide variety of possible conflicts as well as many known "worst case" situations.

You can exercise control of which tasks are running at any given time by use of the SIT commands. Most commonly, you would use the commands to stop or start specific tasks in an attempt to narrow down the source of conflicts.

The tasks available for SIT are summarized in the following table. When a letter is used for the contoller and device, the numbers must be specified. Full descriptions of the tasks are given in following the table.

Table 3-1. Tasks

TASK NAME	DESCRIPTION
lnxO	Exercises the Local Area Networking controller and port, "x" is the number of the LAN controller board.
fwxy	Exercises the four-way controllers and their ports, "x" is the number of the controller board, "y" is the number of the port.
mcxy	Exercises the magnetic cartridge streamer controller and the attached drive, "x" and "y" are the controller and driver numbers respectively.
fdxy	Exercises the floppy diskette controller and the drive, "x" and "y" are the controller and drive numbers, respectively.
wdxy	Exercises the fixed disk controller and it drive, "x" and "y" are the controller and drive numbers, respectively.
memm	Exercises the RAM not occupied by the SIT program.

The task "lnxO" exercises the Local Area Networking controller board and port, "x" is the number (0 or 1) of the controller board. There is only one port per board.

The task broadcasts a message between 1 byte and 1 KB in length for each pass. This is a "write only" test, since no return message can be expected. The LAN cable does not need to be connected for this test.

The task "fwxy" exercises the four-way serial communications controllers and their ports, "x" is the number (0 or 1) of the controller board any "y" is the number (0-3) of the port on that board.

The task causes four-way controller x to transmit a message to port y. The message is a multiple of 80 characters long, between 80 and 1920.

If a terminal is connected to the port, it displays a diagonal scrolling of the terminals printable ASCII characters. This is a "write only" test, and can be run without a terminal connected to the port.

The task "mcxy" exercises the MCS controller board and its port. Since only one MCS is supported, and it has only one port, both x and y must be 0.

This is a read/write test. One pass writes between 1 and 17 blocks (512 Bytes), reads back what was written, and then compares the messages.

The sequencer may set a flag for a pass that causes the drive motor to be turned on and off for each access. This causes a "worst case" drain on the power supply.

FLOPPY DRIVE

The task "fdxy" exercises the floppy controller and drive. Since there is only one controller, x must be 0. "y" is the number of the drive (0 or 1).

This is a read/write test, and can be run only if a diskette is in the drive. Errors will occur if the diskette is write protected or if the diskette is damaged, as well as if there is a problem with the controller or drive.

One pass causes one block (512 Bytes) to be written, read back, and the message compared. The drive motor is usually turned off and back on approximately one out of ten accesses. The sequencer may set a flag that turns the power off and on for every access, causing a worst case drain on the power supply.

FIXED DISK

The task "wdxy" exercises the fixed disk controller and drive. Since there is only one controller, x must be 0. "y" is the number of the drive (0 or 1).

If the "kill" option is not used with the "run" command, this task only accesses the diagnostic cylinder, so the test does not destroy disk data. Using the "kill" option causes it to perform a read/write test on the whole disk, which is destructive to data on the disk.

As a read/write test, between 1 and 17 blocks of data are written to the disk, then read back and compared. As a read only test, between 1 and 17 blocks are read and examined for errors.

MEMORY

The task "memm" exercises the system RAM (random access memory). The area of memory from the end of SIT's program and data areas to the top of RAM are tested.

This is a read/write test. Unlike the other tests, one pass involves two write/read cycles. The first cycle writes the word addresses for each word of memory to the address, reads it back, and compares the two. The second cycle writes, reads, and compares the ones complement of the word address to the address.

COMMANDS

The SIT commands are used to handle the tasks, e.g. select tasks, run selected tasks, start and stop specific tasks, and so on. By carefully controlling the tasks, problems with the system's controllers and devices can be isolated.

The user commands are summarized in Table 3-2. Command formats are summarized on the screen in the command listing and are fully explained in the following discussion. The tasknames used in the commands are those listed in Table 3-1?

Note that SIT is not case sensitive, so the commands may be entered in either upper-case or lower-case letters.

RUN

The "RUN" command starts all tasks that were found to be runnable during the autosize and have not been deselected by the "SELECT" command. The user control task is then put to sleep, and error and news reporting begins.

If the "kill" option is specified, the Winchester task performs both reads and writes.

CAUTION

Using the "RUN" command with the "KILL" option destroys any data on the disk prior to the test.

If the "-kill" option is not specified, write access to the Winchester drive is stopped. Only non-destructive read tests are run.

Format

<sit>run {{-}kill}

Example

<sit>run kill

This command begins all tasks (if none have been deselected), with both read and write tests for the fixed disk driyes. This test is destructive of data on the fixed disks.

Table 3-2. Summary of User Commands

COMMAND	SUMMARY
bferr	Causes a break during comparison of read and write buffers.
cylinder	Causes a particular task to access a particular cylinder.
debug	Enter the ROM debugger.
delay	Specifies a delay between passes.
dlth	Specifies that <taskname> uses data length <hexadec> for all reads and writes.
dpat	Specifies that all data written by a task name are to be a repetition of pattern <hexadec>.
err	Not used
head	Specify that accesses to fixed or floppy disk drive use head number <decimal>.
init	Reinitialize System Interaction Test.
news	Sets time between news updates
option	Not used
prior	Specify order in which tasks are executed.
random	Randomize parameters of tasks.
run	Starts execution of selected tasks.
sage	Enter Sage debugger.
sector	Directs all disk accesses to sector <decimal>.
select	Sends prompt to user for various tasks.
shutdown	Shuts down the system.
sleep	Release console for other tasks.
start	Executes a specified task.
stop	Stops execution of a specific task.
tasks	Displays the tasks available based on autosize,

SELECT

The "SELECT" command specifies the tasks to be started by the "RUN" command.

If "select" is entered without options, you are prompted for eash task found to be runnable during autosizing. For instance:

```
fw03 (n=no,<cr=yes)?
```

To include a task, press RETURN alone. To exclude a task, type "n" and press RETURN.

The "all" option selects all tasks, and the "none" option deselects all tasks.

The "taskname" option adds the specified task to the selected tasks, and the "-taskname" deselects the task. Only one task may be selected or deselected per command line.

The "show" option displays the currently selected tasks.

Format

<sit>select {alllnone!{-}taskname!show}

Example

<sit>select fw02

This command adds task "fw02" to the list of tasks already selected.

<sit>select -wd00

This command deletes the task "wd00" from the list of selected tasks.

SLEEP

Description

The "SLEEP" command releases the console for use by other tasks, while errors and news updates are reported.

"SLEEP" is invoked by the "RUN" command.

To execute further commands, "sleep" must be interrupted. To interrupt "SLEEP", press the ESCAPE key.

Format

<sit>sleep

The "STOP" command ends execution of the specified task. It is used to stop one task while continuing execution of others.

```
<sit>stop taskname
```

```
<sit>stop wd00
```

This command ends execution of the task "wd00".

The "TASKS" command displays the tasks that were found to be "runnable" during autosizing. These are the tasks that are available for selection with the "SELECT" command.

```
<sit>tasks
```

The "SHUTDOWN" command shuts down the system from SIT. This is the only method available for exiting SIT. When the system is shutdown, the re-boot prompt is displayed.

```
     Press 'RETURN' key to reboot ('^C'=alt-load,
       '^S'=self-test):
```

In order to return from SIT to the Diagnostic Executive, it is necessary to shutdown and reboot the executive.

```
<sit>shutdown
```

BFERR

The "BFERR" command is used to cause a break if an error is detected during a comparison of read and write buffers.

If the optional taskname is not specified, any data miscompare between read and write buffers causes a break. If a taskname is specificed, the break only occurs on a miscompare of the read and write buffers for that task.

When a break occurs, the screen displays the prompt:

<bferr>

At this prompt, you can type:

o c - to continue testing

o d - to dump the read/write buffers to the screen.

If the "OFF" option is used (see format), no break occurs. This is the default.

The "ALL" option restores the break occuring for all miscompares.

Format <sit>bferr {tasknameloff!all}

Example <sit>bferr fd00

This command causes a break if there is any miscomparison between the read and write buffers for task "fd00" only.

CYLINDER

The "CYLINDER" command is used to cause the specified task to access the specified cylinder or cylinders, overriding the sequencer.

Lowerbound and upperbound may be any decimal number, and specify cylinders.

If only the lowerbound is specified, only that cylinder is exercised by the task. If both the lowerbound and the upperbound are specified without the "A" option, then the inclusive range of cylinders are tested. If the "A" option is specified, then the task alternates between the two cylinders.

The cylinder numbers are not checked for being in a legal range.

Format <sit>cylinder taskname lowerbound {upperbound} {A}

Example <sit>cylinder wd00 5

This command causes task "wd00" to test cylinder number 5 only.

DEBUG

The "DEBUG" command is used to enter the ROM debugger.

Format

<sit>debug

DELAY

By default, a task is made available to the sequencer for a new pass as soon as it has completed the previous pass. The "DELAY" command causes a delay of the value times 100 milliseconds before the specified task is made available for another pass. If no task is specified, the delay applies to all tasks.

Format

<sit>delay {taskname} value

where "value" is any integer greater than or equal to zero.

DLTH

The "DLTH" command specifies the data length to be used by the specified task for all reads and writes. Normally the sequencer varies the data length.

The data length, dlength, is given in hexadecimal notation specifying the number of bytes.

Format

<sit>dlth taskname dlength

Examples

<sit>dlth wd00 A1

This command sets the data length for task "wd00" to be A1

Notes

Tasks that take data lengths as multiples of a constant, c, will calculate the data length based on the specified hexadecimal value according to this formula:

data length=(int(dlength/c)+1)c

("int" is the integer function).

The "DPAT" command specifies the data pattern used by the specified task.

The data pattern is a four byte pattern in hexadecimal notation. If the pattern entered is less than four bytes, the pattern is filled with hex zeros.

All data written by the task then consists of repetitions of this pattern filled out with zeros to a 32-bits.

<sit>dpat taskname dpattern

<sit>dpat fwOO B1B2B3

This command causes all data written by the floppy diskette drive task to be a repetition of "123".

The "HEAD" command specifies the read/write head or heads to be used in the Winchester or floppy drive task specified. The head numbers are given in decimal notation.

If only the lowerbound is specified, only that head is used. If both the upper and lower bounds are specified but without the "A", then the inclusive range of head is used. If the "A" option is used, then the task alternates between the two heads.

The lower and upper bounds are not checked for being in the legal range.

<sit>head taskname lowerbound {upperbound} {A}

<sit>head wdOO 3 6

This command specifies that task "wdOO" will use only heads 3 through 6, inclusive.

The "INIT" command reinitializes the SIT news update counts.

<sit>init

The "NEWS" command sets the amount of time between news update status messages in seconds. The time is specified in hexadecimal notation.

```
<sit>news seconds
```

The "PRIOR" command specifies the order in which tasks are executed. The larger the number assigned as a task's level, the higher the probability that the task will be started before any other.

```
<sit>prior taskname level
```

```
<sit> fw02 4
```

This command specifies a priority level of 4 to the task "fw02".

The "RANDOM" command instructs the sequencer to make all parameters sent to the tasks to be random. The non-random "worst-case" parameters are not used.

The seed is the value used to initialize the random number generator.

```
<sit>random seed
```

```
<sit>random 2.4
```

This command sets the sequencer to submit random parameters only, and initializes the random number generator with 2.4.

The "SAGE" command invokes the sage debugger. Documentation is not provided at this time. Enter "GO" to continue execution, "DEBUG" to enter the ROM debugger.

```
<sit>sage
```

SECTOR

The "SECTOR" command specifies the sector or sectors to be tested by the specified floppy or fixed disk task. The upper and lower bounds are specified in decimal notation.

If only the upper bound is specified, only that sector is tested. If the upper and lower bounds are specified, but without the "A" option, then the range of sectors from the lower to the upper bound inclusive are tested. If the "A" option is specified, the tasks will alternate between the two sectors.

Format

<sit>sector taskname lowerbound {upperbound} {A}

Example

<sit>sector wd00 45 89

This command specifies that task "wd00" will test sectors 45 through 89 inclusively.

START

The "start" command begins execution of the specified task. Only one task may be started per command line.

"Start" is used to begin execution of tasks in addition to those started by the "RUN" command.

To begin new updates and error message displays for the task, use "sleep".

Format

<sit>start taskname

Example

<sit>start fw03

This command begins execution of the task "fw03".

INDEX