UCSD p-System*

- Editor
- Filer
- Utilities

Part Two: UCSD p-System Filer

*UCSD p-System is a trademark of the Regents of the University of California.

1041570-1
[Part B]

UCSD p-System*

- Editor
- Filer
- Utilities

Part Two: UCSD p-System Filer

*UCSD p-System is a trademark of the Regents of the University of California.

This manual was developed by staff members of the Texas Instruments Education and Communications Center.

TABLE OF CONTENTS

FILER COMMANDS

RECOVERING LOST DATA

SECTION 1: GENERAL INFORMATION

The UCSD p-System* Filer provides commands that allow you to perform a variety of file and device operations. All programs and data are stored by the System in the form of files, and each file resides in a device; therefore, easy manipulation of files and devices is critical to the successful utilization of the p-System. The Filer's powerful features give you the necessary flexibility to perform the operations you want with a minimum of effort.

With the Filer, you can:

- Load, save, erase, and determine the status of the workfile.
- Create, copy, and delete files.
- Determine what peripheral devices are attached to the computer.
- Determine the names, locations, and sizes of disk files and rearrange them to maximize use of available disk memory.
- Examine diskettes that appear to be damaged and attempt to isolate and correct problems.
- Change the System date, the default prefix, and the names of files and disk volumes.

In addition to the TI Home Computer and a TI Color Monitor (or a TI Video Modulator and a television set), the Filer requires the use of the TI Memory Expansion unit, the TI P-Code peripheral, and the TI Disk Memory System (a TI Disk Drive Controller and one to three TI Disk Memory Drives).

After the Filer has been loaded, a promptline showing the Filer commands appears at the top of the screen. The Filer commands, which are accessed by a single keystroke, are the communication interface for interacting with the various Filer functions. Pressing a key causes either an action to be performed or a prompt to be displayed, requesting you to enter the information necessary for the Filer to perform its functions.

*trademark of the Regents of the University of California

1.1 USING THIS MANUAL

This manual is designed both as an introduction to the Filer and as a reference document after you are familiar with the Filer. Section 1 is an introduction to the Filer, providing set-up instructions and discussing special keys used by the System. Section 2 introduces the concept of files and explains how the various types of files are treated by the System. Section 3 details the System's use of volumes and devices, with particular attention to the use of the diskette, which is the primary medium of program and data storage. Section 4 provides general instructions for the use of the Filer, including an introduction to the concept of "wildcards," a powerful method of referencing multiple files and devices. Section 5 explains each of the Filer commands in alphabetical order, according to the letter you press to access them. Section 6 offers advice on attempting to recover data that appears to have been accidentally lost or destroyed. The remainder of the manual contains service and warranty information.

1.2 SET-UP INSTRUCTIONS

The steps involved in accessing the Filer are included in this section. Please read this material completely before proceeding.

1. Be sure that the Memory Expansion unit, the P-Code peripheral, and the Disk Memory System are attached to the computer and turned on. (Refer to the appropriate owner's manual for product details.)

2. Insert the Filer diskette into a disk drive.

3. Turn on the monitor and computer console. The p-System promptline now appears. (**Note:** If you turn on the computer before inserting a diskette in a disk drive, you must insert a diskette and then press **I** to initialize the System before you can proceed.)

4. Press **F** for F(ile to load the Filer.

5. After the Filer is loaded, the screen displays the Filer promptline, a list of the commands available with the Filer. Refer to Section 5 for an explanation of the Filer commands.

1.3 SPECIAL KEYS

A color-coded keyboard overlay for the TI-99/4 console and a two-level strip overlay for the TI-99/4A console are included with the P-Code peripheral to help you more easily identify certain keys that are used with the p-System. On the TI-99/4 console, certain keys are used in combination with the SHIFT and SPACE keys; while on the TI-99/4A console, certain keys are used in combination with the FCTN and CTRL keys. Note that, as you read the manual and use the Filer, the < and > symbols indicate function keys to be pressed and not information to be typed. For your convenience, the function keys available with the Filer are summarized here.

Name	TI-99/4	TI-99/4A	Action
<flush>	SPACE 3	FCTN 3	Acts as a toggle to suspend and restart output to the display.
<break>	SPACE 4	FCTN 4	Stops the program so that the System can be re-initialized.
<stop>	SPACE 5	FCTN 5	Acts as a toggle to suspend and continue the program.
<alpha lock>	SPACE 6	FCTN 6	Acts as a toggle to convert lower-case letter input to upper-case and back again.*
<screen left>	SPACE 7	FCTN 7	Moves the displayed text to the left 20 columns at a time.
<screen right>	SPACE 8	FCTN 8	Moves the displayed text to the right 20 columns at a time.
<line del>	SHIFT Z	FCTN 9	Erases your response to the current prompt.
{	SPACE 1	FCTN F	Types the left brace {.
}	SPACE 2	FCTN G	Types the right brace }.
[	SPACE 9	FCTN R	Types the left bracket [.
]	SPACE 0	FCTN T	Types the right bracket].
<left-arrow> or <backspace>	SHIFT S	FCTN S	Moves the cursor to the left one character and erases that character.
<return>	ENTER	ENTER	Tells the computer to accept the information you type.

*Note that, with the TI-99/4A console, you can press <alpha lock> to select lower-case characters (the default mode when the computer is turned on) and can then press the SHIFT or ALPHA LOCK key on the keyboard to select upper-case characters. With the TI-99/4 console, pressing <alpha lock> is the only toggle available for converting from lower-case letters to upper-case and back again.

SECTION 2: FILES

To the UCSD p-System, a "file" is a collection of information, usually stored on a diskette. A file may contain:

- Program statements, in Pascal or some other language, to be compiled or assembled by the computer.
- A program ready to be executed by the computer.
- Data to be created or used by a program.

A file may also contain other information serving a specific purpose. The various kinds of files are discussed in detail in this section.

Some files contain programs or data essential to the operation of the System itself or necessary to perform a major function; these files are known as "System" files. Other files, called "user" files, are created and maintained by users of the System, such as yourself, to serve a desired purpose.

2.1 FILENAMES

Every file has a unique name, called a "filename," that distinguishes it from all other files in the same volume (see Section 3). While the System allows you great flexibility in naming files, there are certain restrictions that must be carefully observed.

A valid filename consists of up to 15 characters. If you use any lower-case letters as part of a filename (as in "Myfile"), the System changes them to upper-case letters ("MYFILE"). Blanks or non-displaying characters entered as part of a filename are removed by the System.

A filename may be composed of from one to 15 of the following characters.

- The upper-case letters A-Z
- The digits 0-9
- The special characters hyphen (-), slash (/), backslash (\), underline (_), and period (.)

Filenames may not contain the dollar sign ($), colon (:), equals sign (=), question mark (?), or comma (,) characters.

The acceptable special characters, especially the period, are frequently used to indicate relationships among files or to distinguish between related files of different types. For example, the files MYFILE.TEXT and MYFILE.CODE would probably contain different representations of the same information (see Section 2.3).

2.2 SYSTEM FILES

System files, the files important to the operation of the System and its major functions, are identified by the prefix SYSTEM. Some System files contain programs that are accessed by pressing a letter from the System promptline (the System command level), as follows.

Letter	File Invoked	Function
E	SYSTEM.EDITOR	Editor
F	SYSTEM.FILER	Filer
C	SYSTEM.COMPILER	Compiler
A	SYSTEM.ASSMBLER	Assembler
L	SYSTEM.LINKER	Linker

(Note that the filename SYSTEM.ASSMBLER is missing an E so that it complies with the 15-character limit on filenames.)

When you press one of these letters from the System promptline, the System checks each disk drive for the appropriate file. If the file is found, the appropriate function is initiated; if the file is not found, an error message is displayed.

Although the file SYSTEM.COMPILER usually contains the Pascal compiler, it can actually be any available compiler (FORTRAN, BASIC, etc.). By changing filenames so that the compiler you wish to use is named SYSTEM.COMPILER (see Section 5.2), you can access that compiler by pressing **C** from the System promptline.

2.2.1 Built-In System Files

Three essential System files are actually built into your P-Code peripheral.

File	Purpose
SYSTEM.PASCAL	The Operating System
SYSTEM.MISCINFO	Miscellaneous information used by the System
SYSTEM.CHARAC	Character definitions

Since these files are part of the "firmware" (that is, they are stored on integrated circuit chips, or IC's), you cannot modify them. However, you can override the information in the files SYSTEM.MISCINFO (see the SETUP utility in the UCSD p-System Utilities manual) and SYSTEM.CHARAC by having files with those names on a diskette in Disk Drive 1. When the System is first turned on (or if you press **I** to invoke the I(nitialize command from the System promptline), the diskette is checked to see if those files are present. If they are, the System uses the files on the diskette; otherwise, it uses the files in the P-Code peripheral firmware.

Although the essential portions of the file SYSTEM.PASCAL are in firmware, a file of that name may be on the Utilities diskette in the first disk drive. If this file is present, it normally contains error messages that may be displayed when certain error conditions arise. If there is no disk file named SYSTEM.PASCAL containing error messages, an error number, rather than an error message, is displayed when an error occurs. The UCSD p-System P-Code peripheral manual contains a list of the error numbers and their meanings.

2.2.2 Optional System Files

Although they are not essential to the operation of the System, you may want to develop two System files, SYSTEM.LIBRARY and SYSTEM.STARTUP, for an added measure of convenience.

By developing a file named SYSTEM.LIBRARY, you can store routines that perform functions you want to use in many programs, to avoid retyping the routines each time you write a program. This file can contain previously compiled or assembled program routines to be linked with other routines or programs. See the UCSD p-System Utilities and Linker manuals for details on the use of SYSTEM.LIBRARY.

If you have a program that you want to run each time you start up the System (before the System promptline is displayed), name it SYSTEM.STARTUP. The System then executes the program in that file automatically each time the system is turned on or initialized.

2.3 USER FILES

Any file that is not a System file (that is, any file that does not begin with the SYSTEM. prefix) is called a "user" file. User files generally contain program or documentation text, compiled or assembled program code, or data in any kind of user-defined format.

There are no restrictions on the organization or contents of user files. However, certain types of files, indicated by reserved filename suffixes, are treated in very specific ways by the System (the Editor, for example, only edits textfiles). The reserved suffixes are as follows.

Suffix	File Contents
.TEXT	Text (often program statements) entered with the Editor
.BACK	A duplicate copy of a textfile
.CODE	A program that can be executed by the System
.DATA	Data in a user-specified format
.BAD	A file covering a damaged area of a diskette

2.3.1 .TEXT Files

.TEXT files, such as MYFILE.TEXT, are human-readable files, formatted for use by the Editor. You might, for example, use the Editor to enter statements for a Pascal program. The Editor then stores the statements on a diskette as a textfile. The UCSD p-System Editor manual provides detailed instructions for working with textfiles.

Textfiles are internally organized into "pages" of 2 "blocks" each. The length of a block is 512 bytes, making a textfile page 1024 bytes long. The first page of every textfile, called the "header" page, contains information used by the Editor. The Filer can transfer a header page from one diskette to another but not from a diskette to an output device (such as a printer). Since all files created with a .TEXT suffix have a header attached, they are always treated as textfiles, even if the suffix is later dropped from the filename.

The remainder of a textfile is also organized into pages, each containing a series of text lines, where each line of text ends with a <return>. Although one line of text may fill an entire page, the length of a text line is typically 80 characters or less. If the text lines do not exactly fill a page, the page is "padded" with NUL characters (hexadecimal 00) until the page is exactly 1024 bytes long.

If a line of text begins with spaces (such as a line that is indented), the System does not write the leading spaces into the textfile. Instead it creates a "blank-suppression pair," consisting of a DLE character (decimal 16) followed by a byte having a value that is 32 greater than the number of spaces being replaced. For example, if the first 11 characters of a line of text are spaces, the blank-suppression pair is a DLE followed by a byte with the value of 43 (32 greater than 11).

2.3.2 .BACK Files

.BACK files are simply backups, or duplicate copies, of textfiles. To make a backup copy of a .TEXT file, use the T(rans command (see Section 5.14).

2.3.3 .CODE Files

.CODE files, such as MYFILE.CODE, contain either compiled or assembled code. The term "code" refers to a program or program segment that is in a form such that it can be executed by the computer. Codefiles are typically the output of the Compiler or the Assembler; they may also be generated by the Linker from a group of previously existing codefiles.

Codefiles contain either psuedo-code (p-code) or "native" code. P-code is code generated by the Compiler that can be executed by the System. Native code, generated by the Assembler, is machine code that can be run directly by the computer's TMS9900 microprocessor.

A codefile must be created with a .CODE suffix; the suffix can later be dropped, and you can still execute the file. The Compiler and Assembler automatically append .CODE to the names of specified output files.

Codefiles begin with a single block called a segment dictionary, which contains internal information used by the System and Linker.

2.3.4 .DATA Files

.DATA files contain information organized in any format specified by the user. Datafiles typically contain data that is created or used by user programs.

2.3.5 .BAD Files

.BAD files are immobile files that cover physically damaged portions of a diskette. They differ from all other types of files in that they do not actually contain any kind of data but instead isolate only the unusable portions of a diskette. Thus, you can still record information on the remainder of the diskette.

2.4 THE WORKFILE

The workfile consists of several files designed for convenient use by the Editor and the Compiler. In addition, the p-System's R(un command executes the current workfile.

The Editor allows you to create and modify the textfile portion of the workfile (see the UCSD p-System Editor manual). Then, if you successfully compile the workfile, the Compiler creates a codefile as its output file and, if specified, also creates a listing file. If you choose the R(un command from the System promptline, the program that is run is the compiled codefile.

The Filer provides several commands that can help you manipulate the workfile.

Command	Function
G(et	Identifies an existing file as the workfile.
N(ew	Empties the workfile and removes SYSTEM.WRK.TEXT, SYSTEM.WRK.CODE, and SYSTEM.WRK.BACK.
S(ave	Saves the workfile.
W(hat	Displays the name and status of the current workfile.

The use of these commands is explained in detail in Section 5.

SECTION 3: DEVICES

The various peripherals that the p-System uses are called "devices," and the contents of devices are referred to as "volumes." For example, a disk drive (a device) may at any given time contain one of a number of diskettes (volumes). A device, like a file, may be either a source of data or a destination for data. Many of the Filer's data transfer operations apply to devices as well as to files.

The System controls devices through a group of routines collectively called the Basic Input/Output Subsystem (BIOS). A section of the BIOS that controls a particular device is called a device "driver."

3.1 DEVICE STRUCTURE

The System distinguishes between two types of devices: block-structured devices and non-block-structured devices.

Block-structured devices, such as diskettes, are organized into randomly accessible 512-byte blocks. A diskette file may be of any length but must contain a whole number of blocks.

Non-block-structured devices, such as printers, have no internal structure but deal with "serial" character streams (that is, they process a series of characters, one at a time).

3.2 DEVICE IDENTIFICATION

Devices have both names and numbers, which can generally be used interchangeably. Standard devices have fixed names; removable volumes (like diskettes) have their names recorded on them.

Device names and numbers are followed by a colon (:) to distinguish them from filenames and to allow them to be prefixed to filenames. Device numbers are also preceded by a pound sign (#). For example, MYDISK:MYFILE.TEXT refers to the textfile MYFILE.TEXT on the volume MYDISK:. If MYDISK: is in device number four, the complete filename could also be written as #4:MYFILE.TEXT.

A disk drive has no assigned device name; instead, the volume name of the diskette in the drive is considered to be the device name of the disk drive. Thus, while the device number of a disk drive never changes, the device name changes each time a new volume (diskette) is inserted.

The following table lists the device number associated with each device and the reserved volume names that refer to devices.

Number	Name	Description
#1	CONSOLE:	Keyboard and display with echo
#2	SYSTERM:	Keyboard and display without echo
#4	(diskette name):	1st disk drive
#5	(diskette name):	2nd disk drive
#6	PRINTER:	9600 baud RS232 input/output
#7	REMIN:	300 baud RS232 input
#8	REMOUT:	300 baud RS232 output
#9	(diskette name):	3rd disk drive
#14	OS:	Operating System
#31	TAPE:	Audio cassette tape
#32	TP:	Solid State Thermal Printer

3.2.1 CONSOLE: and SYSTERM:

CONSOLE: is the System's standard input and output device for entering commands and other input from the keyboard and displaying information on the monitor screen.

It is also possible to temporarily redirect the input or output of the System (see the UCSD p-System P-Code manual). If you define SYSTERM:, instead of CONSOLE:, as the System's input device, you lose the "echo" effect. Without echo, the characters you type on the keyboard are not displayed on the screen.

3.2.2 Disks

The System supports the operation of up to three disk drives. (See the TI Disk Memory System manual for information about connecting disk drives to the computer.) The first disk drive is device #4, the second drive is device #5, and the third drive is device #9. The System maintains a directory on each diskette containing the filenames and locations of up to 77 files. The prefix filename is the name of the diskette on which the file resides.

The "root" volume is the diskette that is in the first drive (device #4) when the System is "booted." (Booting, or bootstrapping, the System simply means turning it on.) You may substitute an asterisk (*) for the name or number of the root volume. For example, if MYDISK: is the name of the root volume, typing *MYFILE.TEXT is the same as typing MYDISK:MYFILE.TEXT.

The System also has a "default" prefix to save you the trouble of typing a prefix each time you refer to a file. If you type a filename with no prefix, the System automatically attaches the default prefix to the front of the filename. When the System is booted, the default prefix is the name of the root volume. After you have booted the System, you can't change the root volume without rebooting; however, you can change the default prefix with the P(refix command (see Section 5.10).

For example, assume that MYDISK:MYFILE.TEXT and DISK2:MYFILE.TEXT are files on two different diskettes (note that both files are named MYFILE.TEXT). If MYDISK: is the root volume when you boot the System, it is also the default prefix at that time. If you type MYFILE.TEXT, the System assumes that you mean MYDISK:MYFILE.TEXT. If you change the default prefix to DISK2:, typing MYFILE.TEXT is understood to mean DISK2:MYFILE.TEXT. The diskette that the default prefix refers to is often called the "default volume."

You can substitute a colon (:) for the name or number of the default volume. For example, if MYDISK: is the default prefix, typing :MYFILE.TEXT is the same as typing MYDISK:MYFILE.TEXT.

3.2.3 PRINTER:, REMIN:, and REMOUT:

PRINTER:, REMIN:, and REMOUT: all refer to devices connected to a TI RS232 Interface unit attached to the computer. PRINTER:, a 9600-baud device connected to port 2 of the RS232 Interface, can be used for both input and output. ("Baud" is the speed of data transmission in bits-per-second.)

REMIN: and REMOUT: both refer to a 300-baud device connected to port 1 of the RS232 Interface. REMIN: can only be used for input; REMOUT: can only be used for output.

See the TI RS232 Interface owner's manual for complete details on its use.

3.2.4 OS:

The device named OS: is used by the Operating System for internal purposes and is not available to the user.

3.2.5 TAPE:

The device named TAPE: is an audio cassette tape recorder connected to the computer with a TI Cassette Interface Cable (see the User's Reference Guide for set-up information).

3.2.6 TP:

The device named TP: is a TI Solid State Thermal Printer. (See the Thermal Printer owner's manual for complete details on its use.)

SECTION 4: USING THE FILER

To enter the Filer, press the letter **F** from the System promptline. The following promptline is then displayed.

```
Filer: G(et, S(ave, W(hat,? [C.10]
```

The question mark (?) indicates that there are more commands available than can fit on one promptline. Press **?** to display additional Filer commands. The other commands then appear in the order listed here.

```
Filer: N(ew, L(dir, R(em,? [C.10]
Filer: C(hng, T(rans, D(ate,? [C.10]
Filer: Q(uit, B(ad-blks,? [C.10]
Filer: E(xt-dir, K(rnch, M(ake,? [C.10]
Filer: P(refix, V(ols, X(amine,? [C.10]
Filer: Z(ero [C.10]
```

Note that pressing **?** when the last promptline is displayed returns the first promptline to the screen.

Filer commands are invoked by pressing the letter to the left of the parenthesis. For example, press **G** to select the G(et command.

Commands are available even when they are not displayed on the current promptline; if any one of the promptlines is on the display, all Filer commands are available.

(**Note:** The "C.10" displayed at the end of the promptline is the Filer version number. Future releases of the Filer, if any, may display a different version number.)

When you press a letter to select a Filer command, you are frequently prompted to enter additional information. The most common kinds of prompts are those that call for a yes or no answer and those that ask you to enter a volume ID or a file specification.

4.1 YES/NO PROMPTS

If you answer a "yes/no" question by typing any character other than a Y (upper- or lower-case), your response is assumed to be "no." You do not need to press <return> after your response to a yes/no question since your input is accepted as soon as you press a key. To return to the Filer promptline, press <esc>.

4.2 VOLUME ID'S AND FILE SPECIFICATIONS

Some prompts ask you to enter a volume ID, which may be either a volume name or a device number, followed by pressing <return>.

Many prompts ask you to specify a filename. A file specification can be a single filename, a list of filenames, or an expression using wildcards (see Section 4.2.2). A filename may be preceded by a volume ID and followed by an integer in brackets, specifying the size of the file (in blocks). For example, MYFILE.TEXT[4] indicates that the file MYFILE.TEXT occupies four blocks. Size specifications are discussed in the descriptions of the commands that are affected by them. Always follow file specifications by pressing <return>.

All Filer commands except G(et and S(ave require full filenames, including suffixes such as .TEXT and .CODE. Note that G(et and S(ave supply these suffixes automatically.

If you specify a volume or device that the Filer can't find, it displays one of the following messages, and then the Filer promptline returns to the screen.

No such vol on-line <source>
No such vol on-line <dest>

If you specify a file that the Filer can't find, it displays one of the following messages, and then the Filer promptline returns to the screen.

File not found <source>
File not found <dest>

"Source" refers to the first file specification entered; "dest" (for "destination") refers to the second specification, if any.

If more than one on-line volume (devices are correctly attached and turned on) has the same name, the Filer displays a warning to that effect. To avoid confusion, use device numbers to specify on which volume a file is located. Although you may sometimes need to have two volumes with the same name on-line at the same time, it is best to try to avoid this situation.

4.2.1 Filename Lists

When entering a file specification, you may list as many files as desired by separating file specifications with commas. Commands operating on single filenames read them from the file list and process them until there are none left. Commands operating on two filenames, such as C(hng and T(rans, read file specifications in pairs and process each pair until only one filename (or none) remains. If one filename remains, the Filer prompts you to enter the second member of the pair. If an error is detected in the list, the remainder of the list is ignored.

In a filename pair, the dollar sign character ($) indicates that the second filename is the same as the first, perhaps with a different volume name or size specification. For example, if you type #4:MYFILE.TEXT,#5:$ in response to the prompt "Transfer ?", the Filer transfers the file MYFILE.TEXT from device #4 to device #5 without changing the filename.

4.2.2 Wildcards

When used as part of a file specification, two special characters, called "wildcards," allow you to perform an operation on more than one file without entering multiple filenames. The two wildcards are the equals sign (=) and question mark (?) characters.

Just as a wildcard in a card game can represent any card, a wildcard in a file specification represents any portion of the specification. For example, the file specification MYFILE= represents every filename that starts with MYFILE, including MYFILE.TEXT and MYFILE.CODE. The specification =.TEXT refers to every file that ends with the suffix .TEXT, and the specification MY=TEXT refers to every file that starts with MY and ends with TEXT. The file specification = is also valid by itself and refers to every file.

Wildcards are said to be "subset-specifying," in that they are used to specify a group of files which is a "subset" of the "set" of all files. The characters used with wildcards are called "subset-specifying strings" because those characters determine which of the files are part of the subset. For example, the file specification MYFILE=, consisting of the subset-specifying string MYFILE and the wildcard =, specifies as a subset all files starting with MYFILE.

If you specify the = wildcard, the Filer performs the appropriate operation on all files meeting the specification as soon as you press <return>. If you use the ? wildcard, the Filer asks you for verification before performing the operation on each file.

For example, assume that the following files are on MYDISK:, the default volume.

```
FILERDOC2.TEXT
ABC.CODE
ABC
MYFILE.CODE
STATIC.TEXT
LETTER.TEXT
TESTDOC.TEXT
FILERDOC1.TEXT
STATIC.CODE
```

If you type F= in response to the prompt "Remove ?", the Filer responds:

```
MYDISK:FILERDOC2.TEXT  removed
MYDISK:FILERDOC1.TEXT  removed
```

However, if you type F? in response to the "Remove ?" prompt, the Filer first prompts:

```
Remove FILERDOC2.TEXT ?
```

After you type a response (Y or N), the Filer prompts:

```
Remove FILERDOC1.TEXT ?
```

Note that when you use the ? instead of the = wildcard, the Filer gives you a chance to make sure that the resulting operations are the ones you intended.

SECTION 5: FILER COMMANDS

The Filer contains 18 commands for creating, updating, and maintaining files. The commands are explained in this section in alphabetical order, according to the letter you press to access them. For your convenience, each command is listed here with its corresponding section number.

Command	Section
B(ad-blks	5.1
C(hng	5.2
D(ate	5.3
E(xt-dir	5.4
G(et	5.5
K(rnch	5.6
L(dir	5.7
M(ake	5.8
N(ew	5.9
P(refix	5.10
Q(uit	5.11
R(em	5.12
S(ave	5.13
T(rans	5.14
V(ols	5.15
W(hat	5.16
X(amine	5.17
Z(ero	5.18

5.1 B(ad-blks

The B(ad-blks (bad blocks) command scans a diskette to detect any "bad" blocks. A bad block is one that is unusable for some physical reason, such as scratches, warping, fingerprints, etc. Bad blocks may cause errors when you try to write to, or read from, a diskette.

To access the B(ad-blks command, press **B** from the Filer promptline. The Filer prompts:

Bad block scan of ?

Type the volume ID of the diskette you want to scan and press <return>. The Filer then prompts:

Scan for 180 blocks ? (Y/N)

Press **Y** to scan the entire diskette. To check only a portion of the diskette, press **N**. The Filer then prompts:

Scan for how many blocks ?

Type the number of blocks you want to scan and press <return>.

If a bad block is detected, the Filer displays the number of that block. When the scan is completed, the number of each bad block, as well as the total number of bad blocks, is displayed. If a bad block is detected within a file, the filename, starting block number, and ending block number of the "endangered" file are displayed. A file containing a bad block is considered to be endangered because the data in the file could be irretrievably lost.

For example, the detection of two bad blocks on a diskette might result in the display of the following messages.

```
Block 94 is bad
Block 121 is bad
2 bad blocks
File(s) endangered:
MYDISK:MYFILE.TEXT    54    132
```

The messages indicate the detection of two bad blocks. Both blocks 94 and 121 occur in the file MYDISK:MYFILE.TEXT, which extends from block 54 to block 132. Since both bad blocks occur in the same file, only one file is endangered.

If the B(ad-blks command indicates the presence of bad blocks, you can isolate those blocks with the X(amine command (see Section 5.17) so that they can cause no future problems.

5.2 C(hng

The C(hng (change) command lets you change a filename or the name of a diskette volume.

To access the C(hng command, press **C** from the Filer promptline. The Filer prompts:

Change ?

Type two file specifications separated by a comma. The first specifies the file or volume name to be changed, while the second specifies the new name.

For example, if you type MYDISK:MYFILE.TEXT,NEWNAME in response to the "Change ?" prompt, the Filer indicates the filename change as follows.

MYDISK:MYFILE.TEXT --> NEWNAME

If you type only one file specification in response to the "Change ?" prompt, the Filer prompts:

Change to what ?

This prompt allows you to enter the second specification.

The Filer ignores any volume information entered as part of the second file specification. To move files from one volume to another, use the T(rans command (see Section 5.14).

The C(hng command does not affect filetypes, even if an identifying suffix is dropped. In the previous example, NEWNAME is still a textfile because its original name (MYFILE.TEXT) included a .TEXT suffix. Note, however, that since the G(et command searches for the .TEXT suffix, NEWNAME must be renamed NEWNAME.TEXT before it can be specified as the workfile (see Section 2.4).

5.2.1 C(hng with Wildcards

If you include a wildcard character (see Section 4.2.2) in the first file specification, you must also include it in the second specification. The subset-specifying strings in the first file specification are replaced by the strings (called "replacement strings") in the second file specification.

For example, assume that the following files are on the volume MYDISK:.

```
DANIEL.TEXT
FRANCES.TEXT
FREDERICK
FRIEDA.TEXT
```

If you type MYDISK:FR=,XX= in response to the "Change ?" prompt, the Filer responds:

```
MYDISK:FRANCES.TEXT     --> XXANCES.TEXT
MYDISK:FRIEDA.TEXT      --> XXIEDA.TEXT
```

If you use the ? wildcard character instead of the =, the Filer asks you to confirm the change of each filename. For example, if you type MYDISK:FR?,XX? in response to the "Change ?" prompt, the Filer responds:

```
Change MYDISK:FRANCES.TEXT ?
```

After you type a response (Y or N), the Filer prompts:

```
Change MYDISK:FRIEDA.TEXT ?
```

This prompt gives you the opportunity to verify that you have specified the filenames you actually want to change.

The Filer does not change a filename if the change would make the filename too long (greater than 15 characters).

Both a subset-specifying string and a replacement string may be empty. The Filer interprets the file specification = (where the subset-specifying and replacement strings are both empty) to mean every file on the diskette.

For example, assume that the following files are on a diskette.

```
THIS.TEXT
THAT.TEXT
```

If you type T=T,= in response to the "Change ?" prompt, the Filer changes THIS.TEXT to HIS.TEX and THAT.TEXT to HAT.TEX.

5.2.2 Changing Volume Names

You can also change volume names with the C(hng command. For example, if you type MYDISK:,NEWNAME: in response to the "Change ?" prompt, the Filer indicates the change to the volume name as follows.

MYDISK: --> NEWNAME:

5.3 D(ate

The D(ate command lets you display or change the current System date. When you create, modify, or S(ave a file, the System date is recorded with the file. When you list a disk directory, the dates associated with each file are displayed. The System date is stored in the directory of the root volume and remains the same until you change it by selecting the D(ate command.

To access the D(ate command, press **D** from the Filer promptline. The Filer might respond as follows.

```
Date set: <1..31>-<Jan..Dec>-<00..99>
Today is 15-Aug-81
New date ?
```

The current System date is displayed in day-month-year order. To leave the date unchanged, press <return>.

To change the date, type the day, month, and year, separated by hyphens (-), and press <return>. It isn't necessary to enter the full name of the month because the Filer ignores all but the first three letters. Also, note that the year is entered as two digits, not four; for example, type 81 rather than 1981. After you press <return>, the new System date is displayed.

If you type just the day and then press <return>, the System month and year are not changed. Similarly, if you change only the day and month, the System year remains the same.

If you type a hyphen (-) instead of the month, day, or year, that item remains unchanged. For example, typing -Sep changes the System month to September but does not change the System day or year, while typing --82 changes the System year to 1982 but does not change the System day or month.

5.4 E(xt-dir

The E(xt-dir (list extended directory) command lists a disk directory with more detail than the list provided with the L(dir command (see Section 5.7). This section only describes the listing itself; see the description of the L(dir command for a complete explanation of prompts, wildcard options, and output redirection.

To access the E(xt-dir command, press **E** from the Filer promptline. A listing appears next providing the following information about all files.

- Filename
- Length (in blocks)
- Last modification date
- Starting block number
- Filetype

The size and location of all unused areas are also included in the listing.

The following is an example of an extended directory listing.

```
MYDISK:
FILERDOC2.TEXT     28   5-Oct-80     6 Text
ABC.CODE           18   4-Dec-81    34 Code
< UNUSED >         10               52
ABC                 4   1-Feb-82    62 Data
MYFILE.CODE        12   1-Sep-81    66 Code
STATIC.TEXT         8   9-Jun-81    78 Text
LETTER.TEXT        18   9-May-81    86 Text
TESTDOC.TEXT       20   1-Sep-80   104 Text
FILERDOC1.TEXT     24   1-Sep-80   124 Text
STATIC.CODE         6   9-Jun-81   148 Code
< UNUSED >         26              154
9/9 files, 36 unused, 26 in largest
```

5.5 G(et

The G(et command identifies a specified file as the workfile.

To access the G(et command, press **G** from the Filer promptline. The Filer then prompts you for additional information, depending on whether or not an unsaved workfile or a .BACK version of the workfile exists. If you have saved the workfile (see Section 5.13) but don't have a .BACK version of it, the Filer prompts:

Get ?

Type the name of the file you want to specify as the workfile. You don't need to type the suffix because the G(et command loads both the .TEXT and the .CODE versions of the file, if they exist.

If an unsaved workfile is present, the Filer first prompts:

Throw away current workfile ?

Press **Y** to clear the workfile, or press **N** to return to the Filer promptline without clearing the workfile. Remember that the contents of the workfile are lost if you don't save the current workfile before specifying a new one. If you press **Y**, the Filer then displays the "Get?" prompt.

If a .BACK version of the workfile exists, the Filer first prompts:

Remove (workfile name).BACK ?

Press **Y** to remove the .BACK file, or press **N** to leave the .BACK file undisturbed. The Filer then displays the "Get?" prompt.

For example, assume that the following files are on the default diskette.

```
FILERDOC2.TEXT
ABC.CODE
ABC
MYFILE.CODE
STATIC.TEXT
LETTER.TEXT
TESTDOC.TEXT
FILERDOC1.TEXT
STATIC.CODE
```

If you type STATIC in response to the "Get ?" prompt, the Filer responds:

```
Text & Code file loaded
```

The message indicates that both STATIC.TEXT and STATIC.CODE have been specified as the workfile.

If you type STATIC.TEXT or STATIC.CODE in response to the "Get ?" prompt, the result is the same. Both the textfile and the codefile versions are loaded, regardless of which one you specify.

If only one version of the file exists, that version is loaded whether you specify the textfile or codefile. For example, if you type ABC.TEXT in response to the "Get ?" prompt and only the codefile exists, the Filer responds:

```
Code file loaded
```

5.6 K(rnch

The K(rnch (crunch) command lets you rearrange the files on a specified diskette so that they are adjacent and combines unused blocks into one large area.

To access the K(rnch command, press **K** from the Filer promptline. The Filer then prompts:

Crunch ?

Type the volume ID of the diskette you want to crunch. The Filer next prompts:

From end of disk, block 180 ? (Y/N)

Press **Y** if you want to crunch the entire diskette, leaving all files at the front of the diskette and one large unused area at the end. If you press **N**, the Filer prompts:

Starting at block # ?

Type the number of the block from which you want the crunch to start and press <return>. Crunching from a block in the middle of the diskette leaves an unused area in the middle of the diskette, with files clustered toward either end.

The K(rnch command moves the files one at a time. As each file is moved, the Filer displays a message to that effect. For example, if MYFILE is being moved, the Filer displays:

Moving forward MYFILE

The Filer also displays a message when the crunch is completed. For example, if MYDISK: is the diskette being crunched, the Filer displays:

MYDISK: crunched

It's a good idea to scan your diskette for bad blocks (see Section 5.1) before using the K(rnch command. If the diskette contains a bad block that has not been marked .BAD, K(rnch may write part of a file in that block, and the contents of that file are then irrecoverable.

5.7 L(dir

The L(dir (list directory) command provides a complete or partial listing of a disk directory.

To access the L(dir command, press **L** from the Filer promptline. The Filer then prompts:

```
Dir listing of ?
```

Type the volume ID of the diskette. For example, if you type MYDISK: in response to the "Dir listing of ?" prompt, the Filer might respond:

```
MYDISK:
FILERDOC2.TEXT     28  5-Oct-80
ABC.CODE           18  4-Dec-81
ABC                 4  1-Feb-82
MYFILE.CODE        12  1-Sep-81
STATIC.TEXT         8  9-Jun-81
LETTER.TEXT        18  9-May-81
TESTDOC.TEXT       20  1-Sep-80
FILERDOC1.TEXT     24  1-Sep-80
STATIC.CODE         6  9-Jun-81
9/9 files, 36 unused, 26 in largest
```

The first column of the listing provides the filename of all the files, the second column gives the length in blocks, and the third indicates the last modification date.

The bottom line of the listing tells you the number of files in the listing and the number of files on the diskette, the number of unused blocks on the diskette, and the number of blocks in the largest unused area of the diskette.

Use the E(xt-dir command (see Section 5.4) to obtain information on filetypes and on the locations and sizes of files and unused areas of the diskette.

If the directory listing is too long to fit on the display, the Filer lists as many files as possible and then prompts:

Type <space> to continue

Press the <spacebar> to list the rest of the directory. When you get to the end of the directory, the Filer displays the promptline. To return to the Filer promptline before reaching the end of the directory, press <esc>.

5.7.1 L(dir with Wildcards

You may use a wildcard character (see Section 4.2.2) to list any subset of the directory. For example, if you type MYDISK:FIL=TEXT in response to the "Dir listing of ?" prompt, the Filer responds:

```
MYDISK:
FILERDOC2.TEXT      28   5-Oct-80
FILERDOC1.TEXT      24   1-Sep-80
2/9 files, 122 unused, 90 in largest
```

5.7.2 Redirecting L(dir Output

If you type a volume ID followed by a comma and a device name or filename, the directory listing is output to that device or file, instead of to the display. For example, if you type MYDISK:,TP: in response to the "Dir listing of ?" prompt, the directory listing is printed on the Solid State Thermal Printer; or if you type MYDISK:,DISK2:DIRLIST.TEXT as a response, the listing is written to the textfile DIRLIST.TEXT on the volume DISK2:. **Note:** If the output file specification is a disk volume, the volume ID must be followed by a filename or the output disk directory is destroyed. See Section 6.2 for information on the possibility of recovering a lost directory.

5.8 M(ake

The M(ake command lets you create a disk file by making an entry in the disk directory. With the M(ake command, you can create a file for future use, extend the size of a file, or recover a "lost" file (see Section 6.1).

To access the M(ake command, press **M** from the Filer promptline. The Filer then prompts:

Make ?

Type a filename, optionally followed by a file size (in blocks) enclosed in square brackets ([and]), and press <return>. If you type the name of an existing file, the Filer prompts:

Remove old (filename) ?

Press **Y** to remove the existing file, or press **N** to return to the Filer promptline.

For example, if you type MYDISK:MYFILE[28] in response to the "Make ?" prompt, the Filer responds:

MYDISK:MYFILE made

The message indicates that MYFILE was succesfully created in the first unused 28-block or larger area on MYDISK:.

If the size specification is zero or if it is omitted entirely, the Filer creates the file by completely filling the largest unused area of the diskette. If the size specification is an asterisk (*), the file is created in either the second largest unused area or half of the largest unused area, whichever is larger.

The Filer does not create a textfile that is smaller than four blocks or contains an odd number of blocks. If you try to create a textfile with an odd number of blocks, the Filer subtracts one from the size specification to make it an even number.

5.9 N(ew

The N(ew command clears the workfile so that it is empty and unnamed. This command is especially helpful if you want to create a new file with the Editor. The new workfile remains unnamed until you save it (see Section 5.13).

To access the N(ew command, press **N** from the Filer promptline. The Filer then prompts you for additional information, depending on whether or not an unsaved workfile or a .BACK version of the workfile exists. If you have saved the workfile but don't have a .BACK version of it, the Filer clears the workfile and displays the message:

Workfile cleared

If an unsaved workfile is present, the Filer first prompts:

Throw away current workfile ?

Press **Y** to clear the workfile or **N** to return to the Filer promptline without clearing the workfile. Remember that the contents of the workfile are lost if you don't save the workfile before you clear it. If you press **Y**, the Filer also displays the "Workfile cleared" message.

If a .BACK version of the workfile exists, the Filer prompts:

Remove (workfile name).BACK ?

Press **Y** to remove the .BACK file or **N** to create a new workfile while leaving the .BACK file undisturbed. If you press **Y**, the Filer also displays the "Workfile cleared" message.

5.10 P(refix

The P(refix command lets you change the default prefix to a specified volume name or display the current default prefix. The default prefix is the volume name that the Filer places before any filename entered without a volume ID (see Section 3.2.2).

To access the P(refix command, press **P** from the Filer promptline. The Filer then prompts:

Prefix titles by ?

To display the current default prefix without changing it, type a colon (:). To change the prefix, type the volume name of the diskette that you want to make the default volume and press <return>. If you type an entire file specification, the Filer ignores everything but the volume name. The specified volume does not have to be on-line when you set the default prefix.

For example, if you type MYDISK: in response to the "Prefix titles by ?" prompt, the Filer responds:

Prefix is MYDISK:

If you type a device number instead of a volume name in response to the prompt, the new default prefix is the name of the volume in that device. If no volume is in the specified device at that time, the device number becomes the new default prefix, and any volume in the default device is considered to be the default volume until you change the default prefix again.

If you type an asterisk (*) in response to the "Prefix titles by ?" prompt, the new default prefix is the name of the root volume.

5.11 Q(uit

The Q(uit command leaves the Filer and returns the System promptline to the display.

To access the Q(uit command, press **Q** from the Filer promptline.

5.12 R(em

The R(em (remove) command lets you remove (delete) file entries from a disk directory. Although removing a directory entry doesn't actually erase a file, the System considers the blocks occupied by a removed file to be unused and available for storing new files.

To access the R(emove command, press **R** from the Filer promptline. The Filer then prompts:

```
Remove ?
```

Type the name of the file that you want to remove from the disk directory and press <return>.

For example, assume that the following files are on the default volume MYDISK:.

```
FILERDOC2.TEXT
ABC.CODE
ABC
MYFILE.CODE
STATIC.TEXT
LETTER.TEXT
TESTDOC.TEXT
FILERDOC1.TEXT
STATIC.CODE
```

If you type MYFILE.CODE in response to the "Remove ?" prompt, the Filer responds:

```
MYDISK:MYFILE.CODE      --> removed
Update directory ?
```

Although the message indicates that the file has been removed, the file isn't actually removed if you respond to the "Update directory ?" prompt by pressing **N.** This safety feature helps ensure that you do not inadvertently remove the wrong files.

If you respond to the prompt by pressing **Y,** the file MYFILE.CODE is removed from the directory of MYDISK:.

Note: Do not use the R(em command to remove the workfile; instead, use the N(ew command to clear the workfile.

5.12.1 R(em with Wildcards

Wildcards (see Section 4.2.2) are helpful in diminishing the amount of typing necessary to remove multiple files. For example, if you type F= in response to the "Remove ?" prompt, the Filer responds:

```
MYDISK:FILERDOC2.TEXT  --> removed
MYDISK:FILERDOC1.TEXT  --> removed
Update directory ?
```

Press **Y** to remove both files from the MYDISK: directory, or press **N** to return to the Filer promptline.

If you use the ? wildcard character instead of the =, the Filer asks you to confirm the removal of each file. For example, if you type F? in response to the "Remove ?" prompt, the Filer responds:

```
Remove MYDISK:FILERDOC2.TEXT ?
```

After you type a response (Y or N), the Filer prompts:

```
Remove MYDISK:FILERDOC1.TEXT ?
```

This prompt allows you to verify the files you actually want to remove.

After you verify the removal of the last file that meets the specification, the Filer prompts:

```
Update directory ?
```

Press **Y** to remove all the files that you specified for removal; if you indicated that certain files should not be removed, these are left undisturbed. For example, if you press **Y** in response to the "Remove MYDISK:FILERDOC2.TEXT ?" prompt and **N** in response to the "Remove MYDISK:FILERDOC1.TEXT ?" prompt, only FILERDOC2.TEXT is removed when you press **Y** in response to the "Update directory ?" prompt.

Remember that if you type a wildcard character by itself in response to the "Remove ?" prompt, you are specifying every file on the volume. Typing only = or ? instructs the Filer to remove every file in your directory.

5.13 S(ave

The S(ave command lets you save the workfile under a specified filename. After the workfile has been saved, you can use the N(ew command to clear the workfile in preparation for creating a new file with the Editor. If you don't save the workfile before using the N(ew command, the contents of the workfile are lost.

To access the S(ave command, press **S** from the Filer promptline. If a named workfile exists, the Filer prompts:

Save as (filename) ?

Press **Y** to save the workfile under the displayed filename. If you press **N** or if a named workfile does not exist, the Filer prompts:

Save as ?

Type a filename without a suffix and press <return>. The Filer appends the appropriate suffix to the filename. For example, if you type MYFILE in response to the "Save as ?" prompt, the Filer saves the textfile version of the workfile (if it exists) as MYFILE.TEXT and the codefile version (if any) as MYFILE.CODE.

If you type the name of an existing file in response to the "Save as ?" prompt, the Filer prompts:

Remove old (filename) ?

Press **Y** to replace the existing file with the contents of the workfile, or press **N** to return to the Filer promptline without changing the existing file.

5.14 T(rans

The T(rans (transfer) command lets you copy a specified file or volume to a specified destination. The T(rans command can be used to copy a file from one diskette to another, to make a backup copy of an entire diskette, to print a copy of a file as a permanent record, to transmit a file to another computer, or to receive a file from another computer.

To access the T(rans command, press **T** from the Filer promptline. The Filer then prompts:

```
Transfer ?
```

Type two file specifications separated by a comma. The first specifies the file or volume to be copied (the "source" specification), and the second specifies the file or volume that is the destination of the copy.

For example, if you type MYDISK:MYFILE.TEXT,DISK2:MYFILE.TEXT in response to the "Transfer ?" prompt, the Filer copies MYFILE.TEXT from MYDISK: to DISK2: and displays:

```
MYDISK:MYFILE.TEXT        --> DISK2:MYFILE.TEXT
```

If you type only one file specification in response to the "Transfer ?" prompt, the Filer prompts

```
To where ?
```

This prompt allows you to enter the second specification.

To copy a file to another diskette without changing its name, you can type a dollar sign ($) instead of the destination filename. For example, typing MYDISK:MYFILE.TEXT,DISK2:$ is the equivalent of typing MYDISK:MYFILE.TEXT,DISK2:MYFILE.TEXT.

You can copy a file from one diskette to another even if your system has only one disk drive. For example, if you attempt the transfer described above on a one-drive system, the Filer displays:

```
Put in DISK2:
Type <space> to continue
```

Remove MYDISK:, insert DISK2:, and press the <spacebar>. Remember not to remove the source diskette until the Filer prompts you to insert the destination diskette. If a large file is being transferred, you may be prompted to insert the source and destination diskettes alternately until the transfer is complete.

If you include a size with the second file specification, the Filer copies the file to the first area on the diskette that contains adequate space. If you do not specify a size, the Filer copies the file to the largest unused area.

If you specify the same volume ID for both source and destination files, the file is relocated on the same diskette. If you specify the same filename for both source and destination files on a same-disk transfer, the Filer rewrites the file and removes the original file.

CAUTION

If you enter only a volume name for the destination specification and omit the filename, the destination diskette directory is destroyed. For example, if you type MYDISK:MYFILE.TEXT,DISK2: in response to the "Transfer ?" prompt, the Filer prompts:

```
Destroy DISK2: ?
```

This prompt is intended to be a warning. If you press **Y**, the directory of DISK2: is destroyed. Press **N** to return to the Filer promptline without destroying the directory. **Note:** If the file you are transferring is less than three blocks long, you DO NOT receive this warning prompt.

5.14.1 Transferring Files to Output Devices

To transfer a file to a non-block-structured device such as a printer, enter the appropriate volume ID for the destination file specification. For example, if you type MYFILE.TEXT,TP: in response to the "Transfer ?" prompt, the contents of the textfile MYFILE are printed on the TI Thermal Printer.

You can also transfer a file from a non-block-structured input device, such as another computer connected to your computer through a TI RS232 Interface unit (see Section 3.2 for the proper device name). The file being transferred must end with an EOF character (hexadecimal 0F).

5.14.2 T(rans with Wildcards

Wildcards (see Section 4.2.2) are helpful in diminishing the amount of typing necessary to transfer multiple files. For example, if you type MYDISK:F=,DISK2:$ in response to the "Transfer ?" prompt, the Filer might respond:

```
MYDISK:FILERDOC2.TEXT  --> DISK2:FILERDOC2.TEXT
MYDISK:FILERDOC1.TEXT  --> DISK2:FILERDOC1.TEXT
```

If you use the ? wildcard character instead of the =, the Filer asks you to confirm the transfer of each file. For example, if you type MYDISK:F?,DISK2:$ in response to the "Transfer ?" prompt, the Filer responds:

```
Transfer MYDISK:FILERDOC2.TEXT ?
```

After you type a response (Y or N), the Filer prompts:

```
Transfer MYDISK:FILERDOC1.TEXT ?
```

This prompt allows you to verify that you have specified the files you actually want to transfer.

5.14.3 Transferring Volumes

To use T(rans to copy one entire disk volume to another, type only the volume ID's for both the source and destination specifications. When you transfer a block-structured volume to another block-structured volume, the destination volume becomes an exact copy (including directory) of the source volume.

For example, if you want a backup copy of MYDISK: and are willing to erase the volume DISK2:, type MYDISK:,DISK2: in response to the "Transfer ?" prompt. The Filer prompts:

```
Transfer 180 blocks ?
```

Press **Y** to transfer the entire volume. If you press **N**, the Filer prompts:

```
# of blocks to transfer ?
```

Type the number of blocks you wish to transfer and press <return>. Next, the Filer prompts:

```
Destroy DISK2: ?
```

Press **Y** to make DISK2: an exact copy of MYDISK:, or press **N** to return to the Filer promptline.

Note: Although you can transfer the entire contents of one diskette to another using a single disk drive, the process can be time consuming. Because the T(rans command processes only a small amount of data at one time, a great deal of disk manipulation is often necessary to complete the transfer.

5.15 V(ols

The V(ols (volumes) command provides a list of the names and associated device numbers of all volumes currently "on-line." A volume is said to be on-line if it is accessible by the computer. Generally, "on-line" simply means that the device is attached to the computer and turned on. See Section 3 for a detailed discussion of volumes and devices.

To access the V(ols command, press **V** from the Filer promptline. The Filer might respond as follows.

```
Vols on-line:
   1    CONSOLE:
   2    SYSTERM:
   4 #  MYDISK:
   5 #  DISK2:
  14    OS:
  31    TAPE:
  32    TP:
Root vol is  - MYDISK:
Prefix is    - MYDISK:
```

The first column gives the device number and the third column gives the volume name for all on-line volumes. A number sign (#) in the second column indicates that the device is block-structured.

In addition to providing a list of volumes, the V(ols command lets you access disk drives that were empty when the System was booted. When the System is turned on, it checks to see what volumes are on-line. If a disk drive is empty at that time, the System does not access that drive even if you subsequently insert a diskette. The I(nitialize command from the System promptline rechecks each disk drive to see if it contains a disk volume; the Filer's V(ols command is the only other command that performs this function.

5.16 W(hat

The W(hat command displays the name of the workfile and indicates whether or not it has been saved (see Section 5.13).

To access the W(hat command, press **W** from the Filer promptline. If you have previously saved the workfile and have not subsequently used the N(ew command to clear it, the Filer responds:

Workfile is (filename)

If you have made changes to the workfile since the last time you saved it, the Filer also displays:

(not saved)

If a workfile is present but has never been saved, the Filer displays:

not named (not saved)

If there is no workfile, the Filer displays:

No workfile

5.17 X(amine

The X(amine (examine) command lets you try to recover suspected bad blocks on a specified diskette. Before selecting the X(amine command, use the B(ad-blks command (see Section 5.1) to determine if a diskette contains bad blocks and, if so, to determine which blocks are bad.

To access the X(amine command, press **X** from the Filer promptline. The Filer then prompts:

```
Examine blocks on ?
```

Type the volume ID of the diskette you want to examine and press <return>. The Filer next prompts:

```
Block-range ?
```

Type the numbers of the blocks identified as bad by the B(ad-blks command. Enter either a single number or two numbers separated by a hyphen to specify a range of bad blocks. Then press <return>.

If any files are endangered, the Filer lists them as follows.

```
File(s) endangered:
(filename)
      .
      .
      .
(filename)
Fix them ?
```

Press **N** to return to the Filer promptline, or press **Y** if you want the Filer to attempt to "fix" the blocks.

If you press **Y**, the Filer attempts to read the data from each bad block, write it back to the same block, and then read it once more. If the two attempts to read are successful and if exactly the same data is read both times, the Filer displays:

```
Block (block-number) may be ok
```

The message means that the block suspected of being bad has been "fixed"; that is, it is responding correctly to input and output operations, so no corrective action is necessary. A block which is fixed does not necessarily contain the expected data; the "may be ok" message just means that the block is probably physically undamaged.

If either of the attempts to read are unsuccessful or if identical data is not read each time, the Filer displays:

Block (block-number) is bad

After the Filer has examined all suspected bad blocks, it displays a list of any files containing blocks that it was unable to fix, as follows.

File(s) endangered:
(filename)
.
.
.
(filename)
Mark bad blocks ? (files will be removed !) (Y/N)

Press **Y** to mark the bad blocks, or press **N** to return to the Filer promptline. Files that contain bad blocks to be marked are first removed from the disk directory, and data in those files is lost.

When a bad block is marked, it becomes a special file with a .BAD suffix. .BAD files are unavailable for future use, effectively reducing the amount of space on the diskette. To ensure that the damaged portions of a diskette continue to be properly identified, .BAD files are not moved by the K(rnch command (see Section 5.6).

5.18 Z(ero

The Z(ero command lets you set up an empty directory on a specified disk volume. If the specified volume is an "old" diskette (one on which a directory already exists), the existing directory is destroyed.

After you format a diskette with the DFORMAT utility (see the USCD p-System Utilities manual) or with the Disk Manager Solid State Software ™ Command Module, selecting Z(ero changes the volume name to the one you specify here and also creates on the diskette a single, long file named PASCAL.

To access the Z(ero command, press **Z** from the Filer promptline. The Filer then prompts:

```
Zero dir of?
```

Type the volume ID of the diskette on which you wish to create a directory and press <return>. If you specify an old diskette, the Filer prompts:

```
Destroy (volume name)?
```

Press **Y** to destroy the old directory in preparation for the creation of a new directory, or press **N** to return to the Filer promptline.

Next, whether the specified diskette is old or "new" (a diskette without an existing directory), the Filer prompts:

```
Duplicate dir?
```

Press **Y** to create and maintain a duplicate directory, or press **N** to create a single directory only. A duplicate directory can prove to be extremely helpful if an original directory is accidentally destroyed. The COPYDUPDIR utility uses the duplicate directory to restore the original directory (see the UCSD p-System Utilities manual).

If you are creating a directory on an old diskette, the Filer prompts:

Are there 180 blks on the disk ? (Y/N)

Press **Y** to create a disk directory that allows you to use the diskette with the p-System. If you press **N** or if you are creating a directory on a new diskette, the Filer prompts:

# of blocks on the disk ?

Type the number of blocks desired and press <return>. For the diskette to be used with the p-System, your response should be 180.

Next, the Filer prompts:

New vol name ?

Type a valid volume name and press <return>. To let you verify that you have correctly typed the desired name, the Filer prompts:

(new volume name) correct ?

Press **Y** if the displayed name is correct, or press **N** to re-enter the new volume name.

When the Filer has succesfully completed writing the new directory on the diskette, it displays:

(new volume name) zeroed

SECTION 6: RECOVERING LOST DATA

If you accidentally remove a file or destroy a directory entry, you may be able to recover the information that has apparently been lost. This section describes some of the methods for recovering lost files and also explains what to do if you lose an entire directory.

6.1 LOST FILES

When you remove a file, it is still physically on the diskette but is no longer in the directory. The information that the file contained remains on the diskette until another file is written over it. Since the Filer considers the space as being usable, the file could be written over at any time. If a file is accidentally removed, be careful not to do anything that causes data to be written on the diskette, as you may overwrite the lost file. The K(rnch command, for example, is extremely likely to overwrite lost files.

The E(xt-dir command displays not only filenames but also the sizes and locations of unused blocks. By looking at the length of an unused portion and its location in the directory, you can often tell where the lost file was located. With the M(ake command (see Section 5.8) you can recreate a file in the same location, thereby recovering the lost file.

To recover a file with M(ake, the size specification should be equal to the size of the lost file. If you remember the size or if the lost file was adjacent on both sides to files that are still listed in the directory, determining the size specification presents no difficulty.

Since M(ake creates a file of the specified size in the first available location, it may be necessary to create "filler" files to fill up unused (and unwanted) space preceding the location of the lost file. These filler files may later be removed.

For example, consider the following extended directory listing.

```
MYDISK:
FILERDOC2.TEXT     28   5-Oct-80     6 Text
ABC.CODE           18   4-Dec-81    34 Code
< UNUSED >         10               52
ABC                 4   1-Feb-82    62 Data
MYFILE.CODE        12   1-Sep-81    66 Code
STATIC.TEXT         8   9-Jun-81    78 Text
LETTER.TEXT        18   9-May-81    86 Text
TESTDOC.TEXT       20   1-Sep-80   104 Text
FILERDOC1.TEXT     24   1-Sep-80   124 Text
< UNUSED >         32              148
8/8 files, 42 unused, 32 in largest
```

If you know that the file STATIC.CODE was six blocks long and was located just after the file FILERDOC1.TEXT, you can recreate it. First, use the M(ake command to create a file named FILLER, with a size specification of 10 blocks, to fill up the 10-block unused space on the diskette. Next, create STATIC.CODE with a size specification of six blocks. Finally, use the R(em command to remove FILLER from the directory. The resulting directory listing is as follows.

```
MYDISK:
FILERDOC2.TEXT     28   5-Oct-80     6 Text
ABC.CODE           18   4-Dec-81    34 Code
< UNUSED >         10               52
ABC                 4   1-Feb-82    62 Data
MYFILE.CODE        12   1-Sep-81    66 Code
STATIC.TEXT         8   9-Jun-81    78 Text
LETTER.TEXT        18   9-May-81    86 Text
TESTDOC.TEXT       20   1-Sep-80   104 Text
FILERDOC1.TEXT     24   1-Sep-80   124 Text
STATIC.CODE         6   1-Mar-82   148 Code
< UNUSED >         26              154
9/9 files, 36 unused, 26 in largest
```

Remember that, to be able to execute a codefile, it must be created with a .CODE suffix. If you lose a codefile that does not have a .CODE suffix, such as SYSTEM.EDITOR, first use the M(ake command to recreate the file with a .CODE suffix (EDITOR.CODE), and then use the C(hng command to change its name back (to SYSTEM.EDITOR).

RECOVERING LOST DATA

The UCSD p-System Utilities software package contains two programs designed to assist you in file recovery. They are RECOVER and PATCH.

The RECOVER utility can help you if you can't remember or determine where the file was located on the diskette. RECOVER scans the directory for entries that may be valid filenames. If the search doesn't find the desired file, RECOVER scans the entire diskette for areas which resemble .TEXT or .CODE files and asks you if it should attempt to recreate the files. RECOVER is described in detail in the UCSD p-System Utilities manual.

If you think that a directory entry is incorrect, use the PATCH utility to examine the exact contents of the directory. Also, use PATCH to examine a particular block on a diskette to determine whether or not it is part of a lost file. PATCH is described in detail in the UCSD p-System Utilities manual.

6.2 LOST DIRECTORIES

If you use the E(xt-dir or L(dir commands and specify an output file and your output specification is a disk volume without a filename, the disk directory is destroyed. For example, if you type MYDISK:,MYDISK: in response to the "Dir listing of ?" prompt, the Filer overwrites the first few blocks of MYDISK: with a listing of its own directory.

First, use the T(rans command to transfer MYDISK: to an output device. This provides you with a list of the MYDISK: directory. Next, use the Z(ero command to erase the directory. The Z(ero command does not alter the contents of any files on MYDISK:, only the contents of the directory itself. Finally, use the M(ake command to restore all the directory entries. **Note:** This method may not work if the directory listing was not extended and unused areas exist between files.

Recovering a lost directory is simplest when the diskette contains a duplicate directory which was not destroyed. A directory spans blocks 2-5 on a disk; if a duplicate directory exists, it occupies blocks 6-9. Each time the original directory is modified the duplicate directory is updated as well, thus providing a convenient backup.

Two methods are available for putting duplicate directories on all your diskettes. The first is to use the Z(ero command when first creating a diskette with the Filer. When the "Duplicate dir ?" prompt appears, press **Y**. The second method is to use the MARKDUPDIR utility (see below).

The Utilities software package also contains two programs to help you prevent lost directory problems. These programs are COPYDUPDIR and MARKDUPDIR.

If the directory is lost on a diskette that has a duplicate directory, use the COPYDUPDIR utility to move the duplicate directory to the location of the original disk directory. This is usually all the recovery that is necessary.

If a diskette is already in use and contains only one directory, use the MARKDUPDIR utility to create a duplicate directory. You must exercise caution when using MARKDUPDIR because blocks 6-9 of the diskette (the location of the duplicate directory) must be unused, or file information is lost.

COPYDUPDIR and MARKDUPDIR are fully described in the UCSD p-System Utilities manual.

SECTION 7: IN CASE OF DIFFICULTY

1. Be sure that the diskette you are using is the correct one. Use the L(dir (list directory) command in the Filer to check for the correct diskette or program.

2. Ensure that your Memory Expansion unit, P-Code peripheral, and Disk System are properly connected and turned on. Be certain that you have turned on all peripheral devices and have inserted the appropriate diskette before you turn on the computer.

3. If your program does not appear to be working correctly, end the session and remove the diskette from the disk drive. Reinsert the diskette, and follow the "Set-Up Instructions" carefully. If the program still does not appear to be working properly, remove the diskette from the disk drive, turn the computer and all peripherals off, wait 10 seconds, and turn them on again in the order described above. Then load the program again.

4. If you are having difficulty in operating your computer or are receiving error messages, refer to the "Maintenance and Service Information" and "Error Messages" appendices in your Users' Reference Guide or UCSD p-System P-Code manual for additional help.

5. If you continue to have difficulty with your Texas Instruments computer or the UCSD p-Sytem Filer package, please contact the dealer from whom you purchased the unit or program for service directions.

THREE-MONTH LIMITED WARRANTY
HOME COMPUTER SOFTWARE MEDIA

Texas Instruments Incorporated extends this consumer warranty only to the original consumer purchaser.

WARRANTY COVERAGE

This warranty covers the case components of the software package. The components include all cassette tapes, diskettes, plastics, containers, and all other hardware contained in this software package ("the Hardware"). This limited warranty does not extend to the programs contained in the software media and in the accompanying book materials ("the Programs").

The Hardware is warranted against malfunction due to defective materials or construction. **THIS WARRANTY IS VOID IF THE HARDWARE HAS BEEN DAMAGED BY ACCIDENT OR UNREASONABLE USE, NEGLECT, IMPROPER SERVICE OR OTHER CAUSES NOT ARISING OUT OF DEFECTS IN MATERIAL OR CONSTRUCTION.**

WARRANTY DURATION

The Hardware is warranted for a period of three months from the date of original purchase by the consumer.

WARRANTY DISCLAIMERS

ANY IMPLIED WARRANTIES ARISING OUT OF THIS SALE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE THREE-MONTH PERIOD. TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR LOSS OF USE OF THE HARDWARE OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES, OR DAMAGES INCURRED BY THE CONSUMER OR ANY OTHER USER.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you in those states.

LEGAL REMEDIES

This warranty gives you specific legal rights, and you may also have other rights that vary from state to state.

PERFORMANCE BY TI UNDER WARRANTY

During the three-month warranty period, defective Hardware will be replaced when it is returned postage prepaid to a Texas Instruments Service Facility listed below. The replacement Hardware will be warranted for a period of three months from date of replacement. TI strongly recommends that you insure the Hardware for value prior to mailing.

TEXAS INSTRUMENTS CONSUMER SERVICE FACILITIES

Texas Instruments Service Facility
P. O. Box 2500
Lubbock, Texas 79408

Geophysical Services Incorporated
41 Shelley Road
Richmond Hill, Ontario, Canada L4C5G4

Consumers in California and Oregon may contact the following Texas Instruments offices for additional assistance or information.

Texas Instruments Consumer Service
831 South Douglas Street
El Segundo, California 90245
(213) 973-1803

Texas Instruments Consumer Service
6700 Southwest 105th
Kristen Square, Suite 110
Beaverton, Oregon 97005
(503) 643-6758

IMPORTANT NOTICE OF DISCLAIMER REGARDING THE PROGRAMS

The following should be read and understood before purchasing and/or using the software media.

TI does not warrant the Programs will be free from error or will meet the specific requirements of the consumer. The consumer assumes complete responsibility for any decisions made or actions taken based on information obtained using the Programs. Any statements made concerning the utility of the Programs are not to be construed as express or implied warranties.

TEXAS INSTRUMENTS MAKES NO WARRANTY, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE PROGRAMS AND MAKES ALL PROGRAMS AVAILABLE SOLELY ON AN "AS IS" BASIS.

IN NO EVENT SHALL TEXAS INSTRUMENTS BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE PURCHASE OR USE OF THE PROGRAMS AND THE SOLE AND EXCLUSIVE LIABILITY OF TEXAS INSTRUMENTS, REGARDLESS OF THE FORM OF ACTION, SHALL NOT EXCEED THE PURCHASE PRICE OF THE SOFTWARE MEDIA. MOREOVER, TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR ANY CLAIM OF ANY KIND WHATSOEVER BY ANY OTHER PARTY AGAINST THE USER OF THE PROGRAMS.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you in those states.