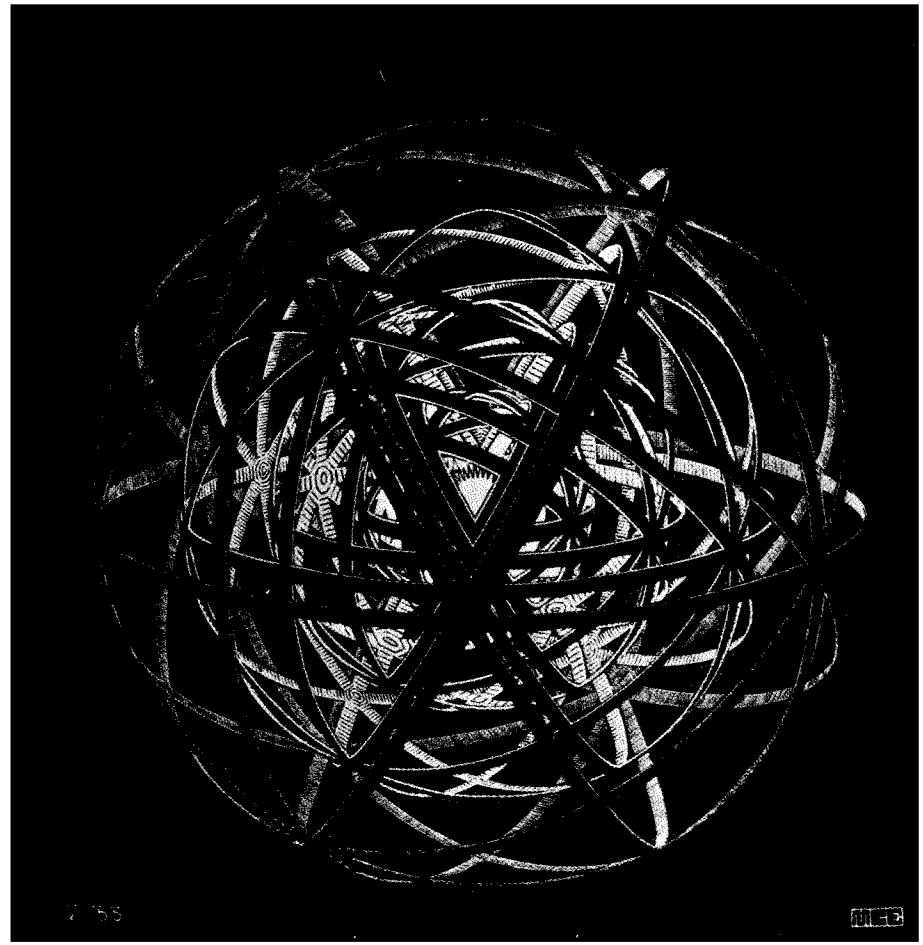
Apple III



Notice

Apple Computer reserves the right to make improvements in the product described in this manual at any time and without notice.

Disclaimer of All Warranties And Liabilities

Apple Computer makes no warranties, either express or implied, with respect to this manual or with respect to the software described in this manual, its quality, performance, merchantability, or fitness for any particular purpose. Apple Computer software is sold or licensed "as is." The entire risk as to its quality and performance is with the buyer. Should the programs prove defective following their purchase, the buyer (and not Apple Computer, its distributor, or its retailer) assumes the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. In no event will Apple Computer be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software, even if Apple Computer has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010

Reorder Apple Product #A3L0004

Apple /// Pascal

Introduction, Filer, and Editor

Acknowledgements

The Apple III Filer and Editor are based on the Filer and Editor developed as part of UCSD Pascal. "UCSD PASCAL" is a trademark of The Regents of The University of California. Use thereof in conjunction with any goods or services is authorized by specific license only and is an indication that the associated product or service has met quality assurance standards prescribed by the University. Any unauthorized use thereof is contrary to the laws of the State of California.

Contents

3 The Filer 28

4 The Editor 90

Appendices

Preface

The Apple III Pascal system is described in three manuals:

Apple III Pascal: Introduction, Filer, and Editor
Apple III Pascal Program Preparation Tools
Apple III Pascal Programmer's Manual (Volumes 1 and 2)

Before using the Apple III Pascal system, or reading its manuals, you should be familiar with starting up the Apple III as described in the Apple III Owner's Guide.

When you are familiar with the contents of that manual, begin reading the Apple III Pascal Introduction, Filer, and Editor manual. The Filer and the Editor described in this manual are needed by everyone who uses the Pascal system. If you are familiar with the Apple II Pascal system, this manual will also show you the differences in operation between the two systems.

Apple III Program Preparation Tools is the next manual that you should read before you start to develop Pascal and assembly-language programs to run on the Apple III. The components of the Apple III Pascal system covered in this manual include:

The Linker, used to combine separately-developed program segments stored in libraries with your application program.

The Apple III Pascal 6502 Assembler, used to translate assembly-language source files produced by the Pascal Editor into machine-language code files.

The Librarian, used to put commonly-used routines into libraries for use with application programs.

Your main source of information while developing Pascal programs will be the two volumes of the Apple III Pascal Programmer's Manual, which contain a complete description of the Pascal language on the Apple III and the use of the Apple III Pascal Compiler.

The Content of this Manual

Chapter 1 presents an overview of Apple III Pascal including how to start the system and how to read the system's prompt messages.

Chapter 2 describes the Pascal system's Command level options and the commands that can be used at all system levels.

Chapter 3 describes the Filer, the part of the system used for manipulating files.

Chapter 4 tells you how to use the Editor and includes a brief tutorial for users having little or no experience using text editors.

Appendix A describes the internal structure of Apple III files.

Appendix B explains how to transport files and programs between the Apple II and Apple III Pascal systems.

Appendix C is a summary of the Command level options, the Filer and Editor commands, and the commands usable at all levels of the system.

Appendix D explains how Apple III Pascal assigns numbers to device drivers that have been configured into the system.

Appendix E is a table of the ASCII character codes.

Symbols Used in this Manual

The following symbols are used throughout this manual:

The pointing finger indicates an especially useful or noteworthy piece of information.

The eye means "watch out." It indicates a warning of a potential hazard to which you should be alert.

1

A Brief Overview

1
A Brief Overview

Apple III Pascal is a complete set of tools used in the development of Pascal and Apple III Pascal Assembly Language subroutines for the Apple III computer. These tools are used together with SOS (the operating system used by the Apple III) and are provided as files on the Apple III Pascal system disks, PASCAL1, PASCAL2, and PASCAL3. The major components of Apple III Pascal include:

The Editor—for creating and modifying program files and other text files

The Filer—for moving files from place to place, copying disks, removing files, renaming files, and other similar chores

The Pascal Compiler—for converting Pascal programs into executable form

The Apple III Pascal Assembler—for converting Apple III Pascal Assembly Language subroutines into machine language

The Linker—for combining separate sections of Pascal and assembly-language programs

The Librarian—a utility program used for putting frequently-used routines into system libraries.

It may be helpful to think of Apple III Pascal as a high-level operating system using the features of SOS. Note that Apple III Pascal is not an independent entity since its function requires the presence of SOS. Each time you boot Apple III Pascal, SOS is automatically loaded into the Apple III's memory.

The most basic information needed to use Pascal on the Apple III is found in this chapter. This information includes starting

the system, interpreting system prompt lines, manipulating directories and files on system diskettes, and creating and editing files.

To use Apple III Pascal, you need a 128K Apple III with a video monitor and at least one external disk drive in addition to the Apple III's built-in drive.

You should set up your Apple, read the Apple III Owner's Guide, and run the programs on the Apple III Demonstration diskette before going any farther in this chapter. The Apple III Owner's guide contains much information related to using Apple III Pascal.

Many programs written for the Apple II Pascal system can be run on an Apple III Pascal system after being recompiled. In addition, some files written on an Apple III Pascal system can be used on an Apple II Pascal system. Appendix B explains how to do this.

You can also use the Apple III Pascal system to edit BASIC text files. For a complete explanation, see this manual's chapter on the Editor.

Starting the System

The three system diskettes that come with your system, PASCAL1, PASCAL2, and PASCAL3, are write-protected. This means that you won't be able to store any new information on them. Before you try to boot Apple III Pascal for the first time, we strongly recommend that you use the Apple III Utilities diskette to make copies of each system diskette. If you are uncertain about how to copy diskettes, see the Apple III Owner's Guide.

After being copied, all three system diskettes should be stored as backups in a safe place. If something happens to the copies, the originals can always be used to produce another set of working diskettes. Don't take chances with your originals!

To start Apple III Pascal (also known as "booting" the system), insert your newly-made copy of PASCAL1 in the built-in drive, close the door to the drive, turn on your monitor, and turn on the Apple III. The built-in disk drive's IN USE light comes on, and the disk drive emits a whirring sound that lets you

know that everything is working. After a while, the disk drive will stop whirring and the monitor screen display will look like Figure 1-1.

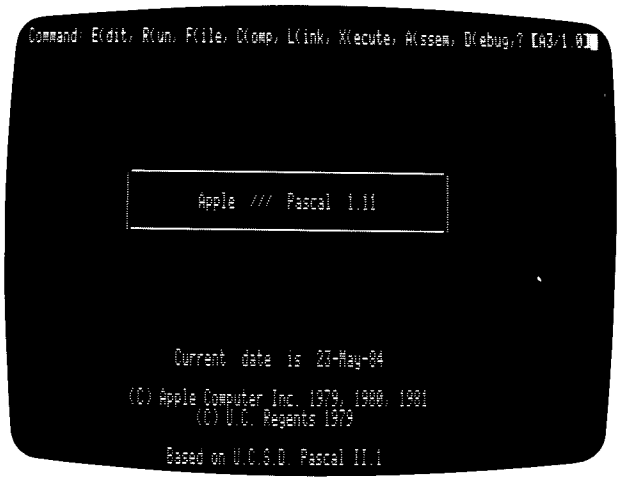


Figure 1-1. System Startup Screen

If you want to begin using Apple III Pascal when your Apple III is already turned on, just insert diskette PASCAL1 in the built-in drive, close the drive door, and then hold down the CONTROL key while pressing the RESET button behind the keyboard.

Prompt Lines

The line of text appearing at the top of the screen when you start up Apple III Pascal is the Command prompt line. Typing the first letter (either upper or lower case) of a Command prompt option on the Command prompt line immediately invokes that option. To use the Editor, just enter an E, and so on. Here is what the Command prompt line looks like:

```
Command: E(dit, R(un, F(ile, C(omp, L(ink, X(ecute, A(ssm,? [A3/1.Ø]
```

Actually, this is only the first half of the Command prompt line. To see the second half, enter a ? . The top line of the screen now looks like this:

```
Command: U(ser restart, I(nitialize, O(ptions, H(alt, M(ake exec
```


Type another ? to return the first half of the Command prompt line to the screen.

Whenever the Command prompt line is displayed, you are at the Command level of the system. Each of the words following the word "Command" indicates the name of an option available to you from the Command level. Figure 1-1 shows the options displayed at the Command level.

To use any given option, the file or files containing the programs needed by that option must be in a disk drive connected to the Apple III. To use the Filer, for example, the system must have access to the program file SYSTEM.FILER ; to use the Editor, the system must have access to the program file SYSTEM.EDITOR . The requirements for the use of all the Apple III Pascal system options are given in Appendix D.

Some of the command options have prompt lines of their own similar in format to the Command prompt line. For example, the Edit prompt line looks like this:

```
>Edit: A(djst C(py D(lete F(nd I(nsrt J(mp M(rgin Q(uit R(plce S(et X(chng Z(ap
```

Each of the words following the word Edit indicates a different command available to you in the Editor. Thus, in the Editor you can insert text, delete text, copy text, and so on.

Note that you always have to go back to the Command level before going to any different Command option as found on the Command option line. It is not possible, for example, to go directly from the Editor to the Filer or from the Filer to the Linker without first passing through the Command level.

Each of the options available from the Command level is explained further in Chapter 2 of this manual.

Rearranging System Diskette Files

The arrangement of files on your system diskettes is by no means fixed. In fact, you are encouraged to move files from one system diskette to another until you have the file arrangement that best suits your needs.

Table 1-1 lists the names and approximate block lengths of the diskette files included on each system diskette. The Filer's List-directory command will display the exact length of each file for you.

PASCAL1 (boot diskette)		PASCAL2 (editing and language processing)	
SOS.KERNEL	44	SYSTEM.EDITOR	56
SOS.DRIVER	39	SYSTEM.SYNTAX	14
SOS.INTERP	34	SYSTEM.COMPILER	8Ø
SYSTEM.PASCAL	63	SYSTEM.ASSMBLER	49
SYSTEM.MISCINFO	1	OPCODES.65Ø2	2
SYSTEM.LIBRARY	39	ERRORS.65Ø2	7
SYSTEM.FILER	5Ø	SYSTEM.LINKER	25
TOTAL	27Ø	TOTAL	233
UNUSED BLOCKS	3	UNUSED BLOCKS	4Ø

PASCAL3 (utilities diskette)	
LIBMAP.CODE	1Ø
LIBRARY.CODE	8
SETUP.CODE	1Ø
AIIFORMAT.CODE	4
TOTAL	32
UNUSED BLOCKS	241

Table 1-1. Contents of System Diskettes

The following sections describe various ways of arranging diskette files on your system diskettes to suit particular needs. When building these diskettes, you will need to use the Filer's Transfer command to move files from one diskette to another, the Filer's Remove command to remove unnecessary files, and the Filer's Change command to change the names of files and diskettes. Filer commands are explained in Chapter 3 of this manual.

Note that SYSTEM.FILER , the file that contains the Filer program, resides on diskette PASCAL1. SYSTEM.FILER does not, however, need to be available to the system once the Filer prompt line is displayed on the screen. Thus, when rearranging diskette files, you can boot the system with diskette PASCAL1,

press F to enter the Filer, and then, if necessary, remove diskette PASCAL1 from the built-in drive to make room for other diskettes. When you want to return to the Command level, place PASCAL1 back in the built-in drive and press Q for Quit.

Text Editing Systems

If you plan to use your system exclusively for text editing, you can create a single boot diskette containing all of the files needed for this purpose. This enables you to leave your other drive(s) free for diskettes containing the files you are currently editing. Here is all you need to do to create a single text editing diskette:

1. Format a new disk with the SOS System Utilities disk, naming the new disk EDIT1.

2. Boot the Pascal system disk, /PASCAL1, and type F to invoke the Filer. Place /EDIT1 in the first external drive.

3. Transfer all files from /PASCAL1 to /EDIT1 by typing

    T=,/EDIT1/$

and pressing RETURN.

4. Remove the /EDIT1/SYSTEM.LIBRARY file by typing

    R/EDIT1/SYSTEM.LIBRARY

and pressing RETURN, then pressing Y when the message

    Update directory?

appears on the screen.

5. Transfer the Editor program file from the disk /PASCAL2 to /EDIT1 by replacing diskette /PASCAL1 in the built-in drive with diskette /PASCAL2 and typing

    T/PASCAL2/SYSTEM.EDITOR,/EDIT1/$

and pressing RETURN.

Your new boot diskette, EDIT1, now contains all of the files needed to create and edit files.

Edit/Compile Systems

If you want to create a file arrangement allowing you to place all of the diskette files needed to edit and compile Pascal programs on a single diskette, you will have to use a two-stage boot. This means that two different diskettes are needed each time you boot the system.

If you look at Table 1-2, you will see that five files are needed to boot Apple III Pascal. These files are:

Diskette 1:

SOS.KERNEL
SOS.INTERP
SOS.DRIVER

Diskette 2:

SYSTEM.PASCAL
SYSTEM.MISCINFO

Table 1-2. Files Used with Two-Stage Boot.

Apple III Pascal refers to the diskette containing the file SYSTEM.PASCAL as the Pascal system disk. Thus, whenever you are prompted to insert the Pascal system disk, you should insert the diskette containing the file SYSTEM.PASCAL . This file is needed each time the system returns to the Command level.

You can make a set of diskettes for a two-stage boot system by using the Filer to make two copies of your new text editing diskette named EDIT1 on SOS-format diskettes.

Change the name of the first copy of EDIT1 to NEWPASCAL1 and remove all files on it except for SOS.KERNEL, SOS.INTERP, and SOS.DRIVER. This diskette will be your first-stage boot diskette.

Now change the name of the second copy of EDIT1 to NEWPASCAL2 and remove only the files SOS.KERNEL, SOS.INTERP, and SOS.DRIVER. This will be your second-stage boot diskette and will remain in the built-in drive most of the time that you are using Apple III Pascal.

If your Apple III is already turned on, you can do a two-stage boot by inserting diskette NEWPASCAL1 in the built-in drive, closing the drive door, and then holding down the CONTROL key while pressing the RESET button behind the keyboard. Then, when prompted, insert diskette NEWPASCAL2 .

More room can be provided on your system disks by removing unnecessary drivers from the SOS.DRIVER file and by removing seldom-used routines from the SYSTEM.LIBRARY file. See the Apple III Standard Device Drivers Manual and the Apple III Pascal Program Preparation Tools Manual's chapter on the Librarian.

Turnkey Systems

Apple III Pascal allows you to set up a turnkey system that automatically begins running a particular program each time the system is booted. Information about how to create a turnkey program, as well as a list of the files that should be included on the turnkey diskette, is included in Apple III Pascal Program Preparation Tools Manual.

Files

A file is defined as a stream of bytes. Thus, information sent to a printer, as well as computer programs, letters, and lists stored on diskettes, are all examples of files.

Most files used by Apple III Pascal are of either of two types: text files that store information such as computer programs, letters, and reports; and code files containing P-code, the translated version of a program.

Much of this manual is devoted to a discussion of diskette files--files stored on disks or diskettes. When a file is created by the Editor, it is stored in the Apple's memory. Then, when you are ready to save your file, you use one of Apple III Pascal Editor's commands to save a copy of the file on to a diskette. To change the content of a file already on disk by using the Editor, you must first copy the file into memory; then you can change the contents of the file with the Editor, and again save the file on to diskette.

Each time a diskette file is created or modified, information about that file, including the file's name, length, type, and last modification date, is placed in a special diskette file

called the diskette directory. Filer commands enable you to display and manipulate the information contained in the diskette directory.

The Workfile

The workfile is a special file that may aid in the development or revision of a program. It is most useful in developing smaller or "beginner" programs whose source is contained in a single text file. If you are working with something like the Great American Novel (or Program), you will find it easier to not use the system workfile.

There are two parts to a workfile: the text portion, containing human-readable text, and the code portion, containing compiled P-code. The text portion of the workfile is always listed on the Pascal system diskette's directory as SYSTEM.WRK.TEXT while the code portion is listed on the same diskette's directory as SYSTEM.WRK.CODE .

A workfile is usually created in the Editor. A new text file that you create with the Editor has no name until it is saved to disk. Saving your new file to disk with the Editor's Quit Update command directs the system to store the workfile on the Pascal system diskette, under the name SYSTEM.WRK.TEXT .

If, after the workfile has been saved onto diskette, you type R for Run, the workfile is compiled and executed (assuming the workfile is a Pascal program). Following a successful compilation, the compiled version of the workfile is automatically saved on the Pascal system diskette, under the name SYSTEM.WRK.CODE . Another way to make a workfile is to change the name of a textfile to SYSTEM.WRK.TEXT and store it on the Pascal system diskette.

You can edit, compile, assemble, link or run the workfile as often as you wish without having to tell the system that the file you want it to act on is the workfile. Each of these operations assumes that you are referring to the workfile on the Pascal system diskette.

Suppose, for example, that you have just started up the system and that you have both a text portion (SYSTEM.WRK.TEXT) and a code portion (SYSTEM.WRK.CODE) of the workfile stored on the Pascal system diskette. Then, with the Command prompt line showing on the screen, you type E for Edit. Rather than asking you to specify the name of the file you want to edit, the

system automatically gets the text portion of the workfile from the Pascal system diskette, reads it into the Editor's workspace and displays it on the screen.

Only one workfile may exist in the system at any one time. If a workfile already exists and you want to create a new workfile, you can use the Filer's Save command. The Filer's Save command allows you to give the workfile its own unique local filename and then saves your file, using its new name, onto whatever diskette you specify.

Next you can use the Filer's New command to destroy the old workfile, thus making room for a new workfile. If you want to designate a file that has been stored on diskette as the next workfile, use the Filer's Get command.

The two parts of the workfile, SYSTEM.WRK.TEXT and SYSTEM.WRK.CODE , are saved and retrieved together. Individual commands automatically act on the appropriate part of the workfile. The Edit command, for example, only acts on the text portion of the workfile (SYSTEM.WRK.TEXT) while the Execute or Run commands (commands that execute compiled P-code) act on the code portion of the workfile (SYSTEM.WRK.CODE).

A description of the Filer commands used to manipulate workfiles can be found in the workfile commands section of this manual's chapter on the Filer. The Editor chapter includes a step-by-step demonstration illustrating the use of a workfile.

Creating and Editing Files

In general, users will follow one of two procedures when creating and editing text files.

The first method, which uses the system's workfile, is most useful when you are using the Editor to write very small computer programs.

The second method, which uses the Editor's Write and Save commands, is used both for program preparation and for editing non-program text such as letters, reports, and manuals. These two methods are illustrated in Figure 1-2.

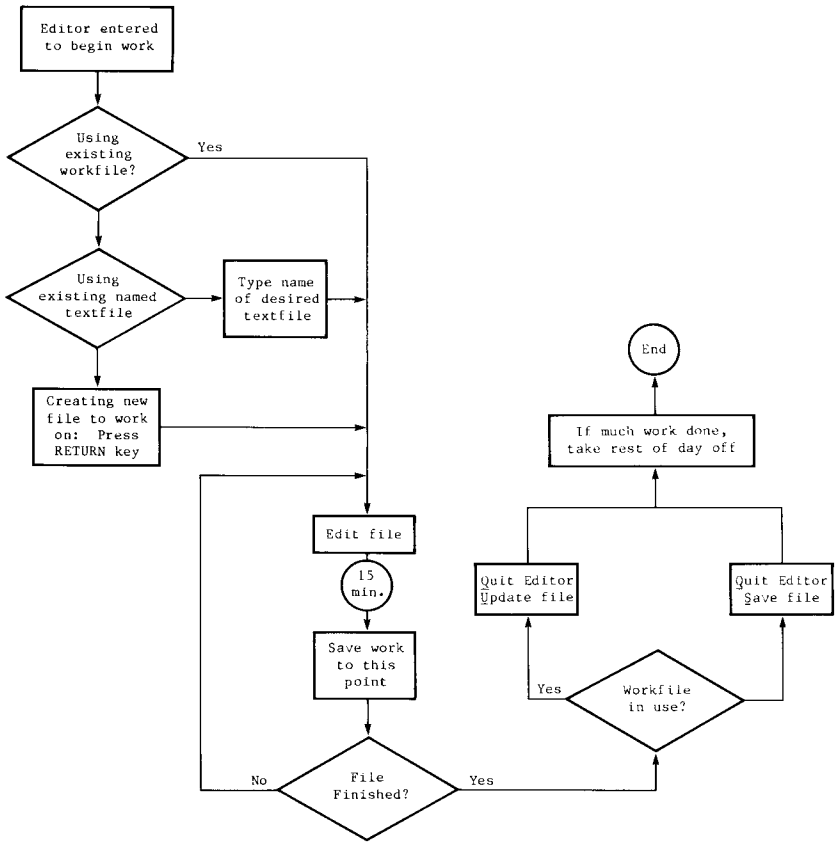


Figure 1-2. Creating and Editing Text Files.

A more detailed description of creating and editing files is included in this manual's chapter on the Editor. Information about program execution is included in the Apple III Pascal Program Preparation Tools manual.

2

The Command Level

2
The Command Level

Using the Command Level

You reach the Command level of the system whenever you boot the system or press CONTROL-RESET, when the system re-initializes itself after a run-time error, when you quit the Editor or the Filer, and when you finish compiling, assembling, linking, executing, or running any program. You have already seen the Command prompt line:

```
Command: E(dit, R(un, F(ile, C(omp, L(ink, X(ecute, A(ssem, ? [A3/1.Ø]
```

When you type a ?, the remaining Command-level options are shown:

```
Command: U(ser restart, I(nitialize, H(alt, O(ptions, M(ake exec
```

Before you specify a particular Command-level option, you should make sure that the diskette file(s) needed by that option are available. In most cases, the required diskette file may be on any of your system's disk drives. The system just goes through the diskettes in every drive until it finds a file with the necessary local filename.

The default workfiles (SYSTEM.WRK.TEXT and SYSTEM.WRK.CODE) and the files SYSTEM.LIBRARY , SYSTEM.PASCAL , and SYSTEM.MISCINFO will be found by the system only if they are on the Pascal system diskette.

Each time the system returns to the Command level following the termination of any option or program, the diskette file SYSTEM.PASCAL should be in the built-in drive. This file contains the Command-level portion of the Apple III Pascal system.

The system tries to re-enter the Command level when you exit from any one of the command-level options, when the system

reaches the end of any program that you have executed, or when the system encounters any non-fatal execution error. If the system diskette is not in the built-in drive at any of these times, the system will ask you to place it in the built-in drive.

Most system files must be available in one of the disk drives constantly, from the moment you select the Command option using that file until you quit that option or until it terminates. This is true of the Edit, Compile, and Assemble options. These programs have been written so that different portions of the program are called in from files as they are needed, thus taking up a minimum of the computer's memory. If the system needs a particular file, it will ask you to insert the appropriate diskette.

The Filer and Linker options operate differently than the other options. The only time SYSTEM.FILER is needed is at the moment you select the File option. When the Filer prompt line appears, SYSTEM.FILER is no longer necessary and the diskette containing SYSTEM.FILER may be removed from the system to make room for other diskettes. The only time SYSTEM.LINKER must be available is when you type L to invoke the Linker. When the Linker prompt line appears, SYSTEM.LINKER is no longer necessary and the diskette containing SYSTEM.LINKER may be removed from the system to make room for other diskettes.

The Command Level Options

The following section describes each of the Command-level options. Many of these options are explained only briefly below. You will find complete descriptions of the Filer and Editor later in this manual. Apple III Pascal Program Preparation Tools describes the Assembler and Linker in detail. Information about the Compiler can be found in the Apple III Pascal Programmer's Manual.

Figure 2-1 shown below illustrates the over-all relationship between the Command level and the functions available through it. All major system functions are reached from the Command level, and you cannot go from any function to any other function without passing through the Command level.

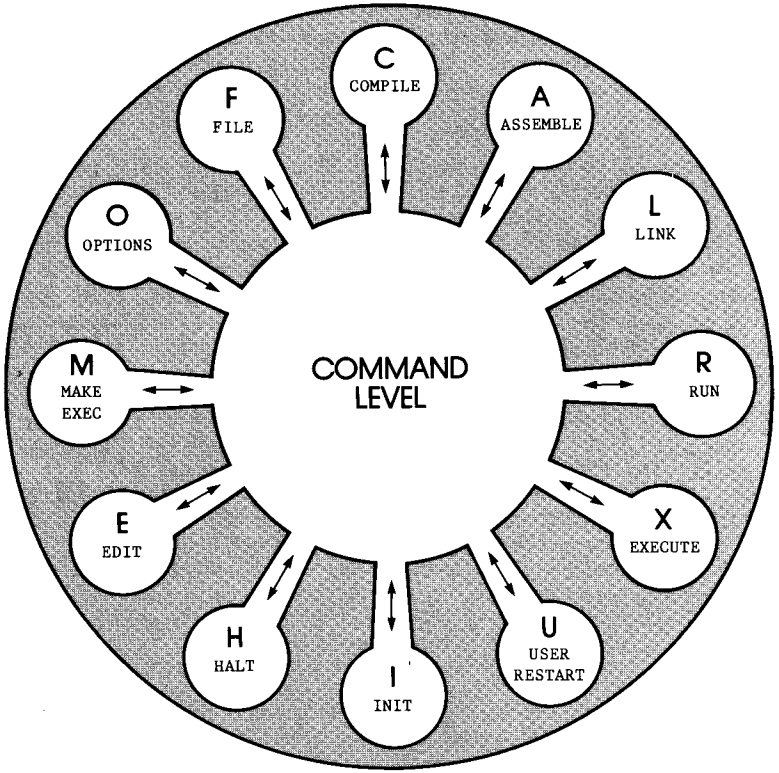


Figure 2-1. The Command Level.

File

To invoke the Filer, type F while at the Command level. The Filer contains commands for moving and deleting files. Other Filer commands tell you what peripheral devices and diskettes are currently available to the system, and what files are stored on each diskette.

Still other Filer commands let you check diskettes for damage or recording errors, and let you set the system's default directory or subdirectory name and the date. For a complete description of these commands, see the chapter The Filer in this manual.

Edit

Typing E while at the Command level invokes the Editor program. If SYSTEM.WRK.TEXT is on the boot volume when the Editor is invoked, the system will load the file into memory for editing. Otherwise, the Editor gives you the option of either editing a textfile already stored on disk or creating a new textfile.

Editor commands allow you to insert and delete information, find and replace specified character strings, change the text format, combine files, etc. After exiting from the Editor, you may save your edited text back to your original file or in another specified disk file. For details, see the chapter The Editor in this manual.

Compile

Typing C while at the Command level invokes the Pascal Compiler. The Compiler reads a text file containing Pascal language statements and translates this into machine code for a so-called "P-machine". The P-machine is not hardware, but an interpreting program that reads the P-code file and executes the instructions given to it there.

If SYSTEM.WRK.TEXT is available on the system disk, it is automatically read into memory and compiled. Otherwise, the Compiler asks you to specify the name of the file to be compiled and the name of the codefile that will contain the compiled program.

When the Compiler detects a syntax error during compilation, the system asks you whether you want to enter the Editor to correct the error, continue compiling the program, or exit from the program. If you continue compilation after finding errors, you won't have a usable codefile to execute, but you will be able to see what errors have been found in your program.

After a successful compilation, the file that has been converted into P-code is saved in the code workfile unless you have previously specified another codefile. For more details about the Compiler, see the chapter on the Pascal Compiler in the Apple III Pascal Programmer's Manual.

Assemble

The Apple III Pascal Assembler is invoked by typing A from the Command level. The Assembler translates a text file containing assembly-language statements and converts them into 65Ø2

machine language to be run as a subroutine for a Pascal program.

If SYSTEM.WRK.TEXT is available, it is automatically read into the computer's memory for assembling. Otherwise, the Assembler asks you to specify the name of the file to be assembled and the name of the codefile that will contain the assembled program.

If the Assembler detects a syntax error during assembly, it gives you the option of calling the Editor, which points out the error and lets you correct it. After a successful assembly, the resulting machine code is saved in the code workfile unless you have previously specified another codefile. For more information, see the chapter on the Assembler in Apple III Pascal Program Preparation Tools.

Link

Typing L from the Command level invokes the system's Linker program. The Linker is used to combine code files containing P-code or assembled machine code into a single codefile.

Unlike the automatic linking initiated by the Run option (see description below), the Link option allows you to link previously compiled or assembled routines into your program, taking those routines from SYSTEM.LIBRARY or any other specified library file. For more information, see the chapter on the Linker in Apple III Pascal Program Preparation Tools.

Execute

To use the Execute option, type X from the Command level. The function of the Execute option is to execute a previously compiled codefile. After you invoke this option, the system asks you to specify the compiled codefile that you want to execute. You should respond by typing the pathname of the compiled P-code program. Pathnames are explained in the Apple III Owner's guide.

In most instances you will use the Execute option rather than the Run option when you want to execute a program that has already been compiled but is not currently in the workfile.

The Execute option is also used to invoke system utilities such as the system librarian, a program that allows you to combine separately compiled or assembled codefiles to make a library file. Information about the Execute option as well as a

complete explanation of the system librarian is included in Apple III Pascal Program Preparation Tools.

Run

Typing R from the Command level initiates the Run sequence, which combines the Command options Compile, Link, and Execute, as needed. Run is one of the options of the normal program-development process.

To successfully use the Run option, you must have a workfile on the Pascal system diskette. If the code version of the workfile (SYSTEM.WRK.CODE) is present, the Run option simply executes your program. If only the text portion of the workfile is present, the Run option compiles the workfile, storing the result as SYSTEM.WRK.CODE and then executes the code portion of the workfile. If the codefile requires linkage to other routines, the Linker is automatically invoked and looks for the specified routines in the file SYSTEM.LIBRARY on the Pascal system diskette.

User Restart

Typing U from the Command level tells the system to begin executing the program or option that was last used. User restart is quicker and uses fewer keystrokes than using Execute to rerun a program. For example, if you have just left the Editor, the User restart option will re-invoke it; if you have just finished executing a program, that program will be executed again.

Initialize

Typing I from the Command level causes the system to restart the Pascal command processor. This has the effect of restarting the Pascal system. The Prefix directory or subdirectory name that has been assigned by the Filer's Prefix command does not change. See the chapter in this manual on the Filer for more information on the Prefix command. A diskette containing SYSTEM.PASCAL should be in the built-in drive when this option is chosen.

Halt

Typing H from the Command level causes the P-code interpreter to be restarted and the file containing the Pascal command

processor (SYSTEM.PASCAL) and SYSTEM.MISCINFO to be reloaded into the computer's memory.

A more detailed description of the operation of the Halt command is given below.

First, the Pascal operating system (contained in the file SYSTEM.PASCAL) terminates its operation and control is passed to the Pascal interpreter (SOS.INTERP).

The interpreter then goes through its BIOS (Basic I/O System) initialization sequence. As this sequence progresses, SOS drivers are allocated to Pascal device numbers, CONSOLE is opened as an I/O device, and a number of CONSOLE parameters are set.

After this is completed, SYSTEM.PASCAL is read back into memory from disk and begins execution. It then performs system initialization tasks including reading SYSTEM.MISCINFO into memory from disk. SYSTEM.MISCINFO is used by the Pascal system to store information about the system configuration, the last-set date, and so on.

After all this has been completed, the system startup display appears and the command promptline is displayed.

Options

Typing an O from the Command level causes the following Options menu to appear on the screen.

```
Options: A, B, C, Q(uit

   A)  Change Graphics space allocation
       Currently ØK bytes reserved for Graphics (No Graphics)

   B)  Change status of Apple II disk routines
       Currently Apple II disk routines are resident

   C)  Change file name display format
       Currently file names are displayed in Apple II Pascal
       format
```

The first option pertains to the amount of memory the system sets aside for the display of graphics. If, when executing a program that uses graphics, you fail to reserve sufficient space in memory, the program will abort.

If you select a graphics mode requiring re-configuration of the Apple's memory, the system will reboot immediately after you type Q to exit from the Options menu. A full discussion of using graphics with Apple III Pascal is included in the Apple III Pascal Programmer's Manual.

If you press A to change the system's graphics space allocation, you are shown the graphics options menu:

```
Graphics Options: A, B, C, D, Q
```

Option	Bytes reserved for Graphics
A)	ØK bytes (No Graphics)
B)	8K bytes
C)	16K bytes
D)	32K bytes

```
Currently you have option A
```

You can pick the amount of memory to set aside for graphics use by typing the letter for that amount. If you change your mind, reenter the graphics option menu by typing A again, then type the letter of your new choice. The option you last pick before you type Q to quit the Option menu is the one that the system will use.

Option B allows you to choose whether Apple II disk routines will be resident in memory or not. If you are using both SOS-format diskettes and Apple II Pascal-format diskettes, then the Apple II disk routines should be resident. If you are using only SOS-format diskettes, you can save some memory space by making the Apple II disk routines non-resident.

The final option, C, selects the format used to display filenames, either the SOS format or the Apple II Pascal format. You may use either format for input, but Option C defines which one will be used by the system to display pathnames. The Apple II Pascal filenaming convention is described in Appendix B. Unless otherwise stated, this manual follows the SOS filename conventions.

Since the system stores the state of all options on diskette, rather than in memory, your selection of options is not lost even if you reboot the Apple III. The only way to change an

option is to recall the Options menu and then select the desired new option.

To exit from the Options command, type Q. The system will reboot if re-booting is necessary for reallocation of space for graphics usage. When this occurs, the system is re-started, and everything in memory is lost. The Pascal operating system is re-loaded into memory, so the diskette file SYSTEM.PASCAL must be in the built-in drive.

Make Exec

The Make exec option is used to create exec files. This option is invoked by typing M from the Command level. A full explanation of exec files is included in Apple III Pascal Program Preparation Tools.

Programs that might require a change in options, such as those using graphics, can be started by an exec file that sets the options before starting the program.

Commands Usable at all Levels

Certain commands can be executed at any level of the system, regardless of what the system is doing at the moment. These commands are listed below; these commands do not appear on any of the system's prompt lines.

CONTROL-\

Pressing the backslash (\ ) key while holding down the CONTROL key interrupts the current program and issues the message

```
Program interrupted by user
S# Ø, P# 9, I#184
Type <space> to continue
```

The second line of the displayed message tells where in the program that execution was halted. Press the spacebar to reinitialize the system. This command should be used when you want to stop a program. Disk operations and assembly-language routines continue to completion before the program will stop.

CONTROL-RESET (or Power Down-and-Up)

Pressing the RESET button while holding down the CONTROL key, or turning the Apple's power switch off and then on again, does a cold boot of the system, just as if the system were being turned on for the first time.

This command will stop any on-going process at the expense of losing whatever is in the computer's memory, and possibly damaging a disk directory. When the system hangs (stops and does not respond to the keyboard), this command will always re-start the system. You should use a control-reset only when at the Command level or the system has crashed. After this command, you will have to repeat the normal startup procedure.

Numeric-Keypad Commands

The following commands are invoked by typing a number key while holding down the CONTROL key. For these commands, the number keys on the numeric keypad must be used. Typing the numbers from the main keyboard will not work.

Control–5

Pressing CONTROL-5 causes output to the display screen to be suppressed. Pressing CONTROL-5 again resumes display updating. This function allows compilations, assemblies, and programs to run faster since no time is spent in updating the screen, and processor time can be used for other tasks.

Note that any program that makes a read to the console will turn the screen back on, just like pressing CONTROL-5 the second time.

Control–6

Pressing CONTROL-6 causes the system to forget any characters that you have typed ahead. This lets you change your mind about any command inputs up to the point that you have actually pressed the return key. If you hold down a key so long that the repeat-key function creates too many commands for the system to keep up with, typing CONTROL-6 will let the system ignore the characters that have been typed ahead.

Control–7

Pressing CONTROL-7 suspends screen output. After you issue this command, the next time a display is sent to the screen, all output will cease and the program halts until you issue the same command again. Output will then continue as if there had been no interruption.

Control–8

Pressing CONTROL-8 makes screen control characters visible on the display. When you issue this command, the screen stops recognizing screen control codes, and instead displays the control character abbreviations of all screen control codes it is sent. When you type another CONTROL-8, the system returns to its normal state.

Control–9

Pressing CONTROL-9 causes subsequent program output to be discarded. This means that the program continues to run, but its output is not sent to the screen or the printer. When the next CONTROL-9 is typed, the system returns to its usual state.

Summary

Command Level Options

File	Invokes the Filer, which is used to save, move, and retrieve information stored on diskettes.
Edit	Invokes the Editor, which is used to create and modify text. Reads the workfile or other specified textfile into the computer for editing.
Compile	Invokes the Pascal Compiler, which converts the text of a program found in the workfile or other specified textfile into executable P-code.
Assemble	Invokes the Assembler, which converts the text of an assembly-language subroutine found in the workfile or other specified textfile into machine language.
Link	Combines external P-code and machine-language subroutines found in SYSTEM.LIBRARY or other specified library codefile into a Pascal host program found in the code workfile or other specified host codefile.
Execute	Loads and runs the specified program codefile and executes EXEC files.
Run	Executes the current workfile, automatically compiling and linking (from SYSTEM.LIBRARY) first, if necessary.
User restart	Attempts to execute the last program or option that was executed.
Initialize	Does a warm boot of the system.
Halt	Does a luke-warm boot of the system.
Options	Enables the user to set the size of the memory area to be used for graphics, and specify the status of the Apple II Pascal format file handlers and the filenaming convention (SOS or Pascal) that will be used to display pathnames.
Make exec	Used to create exec files.

System Commands

The following commands are available at all levels of the system. All numbers used in these commands must be typed on the numeric key pad.

CTRL-\	Interrupts the current program.
CTRL-5	Toggles display refresh on or off.
CTRL-6	Causes the system to forget any characters that have been typed ahead.
CTRL-7	Temporarily stops any program or process. On the next CTRL-7, the program continues.
CTRL-8	Makes control-codes visible on the screen.
CTRL-9	Stops program output to the screen or printer until the next CTRL-9, without stopping the program.
CONTROL-RESET	Causes a cold boot.
Power off-on	Causes a cold boot.

3

The Filer

3
The Filer

Introduction

The Filer manipulates files, which are the fundamental unit of permanent storage on the Apple III. Files can contain different kinds of information—computer programs, letters, lists of data, directories of disk contents, and so on. Some Filer commands pertain only to files stored on disk; others pertain to unblocked device files such as the printer and console.

The Filer is used in coordination with SOS, the Apple III operating system. The underlying structure of the Apple III file system is described in the chapter titled "The System" in the Apple III Owner's Guide. You should read that chapter before going on with this discussion of the Filer.

Here is an overview of some Filer functions and the commands performing them:

Lists the devices and volumes currently on-line	Volumes
Displays detailed information about the files stored in a directory or subdirectory.	List directory Extended-list
Copies a file or volume to another volume or sends the file to a device such as a printer.	Transfer
Creates a subdirectory	Make
Reserves space on a disk for a file to which you later plan to add information.	Make

Allows you to alter three file characteristics as displayed by the Extended directory listing. The alterable characteristics are: Write protect status, date, and filetype.	Alter
Changes the name of a disk directory, subdirectory, or other file.	Change
Removes a subdirectory or other file from a disk	Remove
Removes all files contained in a disk directory or subdirectory	Zero
Changes the default directory name	Prefix
Sets the date and time	Date
Exits from the Filer and returns the system to the Command level	Quit
Designates a specified disk file as the workfile	Get
Saves the current workfile under a given unique pathname	Save
Clears the current workfile so that a new workfile can be created	New
Tells the current state (saved or not) of the current workfile	What
Crunches the files on a disk to increase available space (necessary only on Apple II Pascal diskettes)	Krunch
Tests a disk to see if it is defective	Bad-blocks
Marks defective blocks on a diskette so that information cannot be stored on them (may be used only with Apple II Pascal diskettes)	Examine

The Filer can also handle diskette files formatted for the Apple II Pascal system. See Appendix B for a description of using Apple II Pascal diskettes with Apple III Pascal.

Apple III Pascal Option C allows you to choose between two different filename display conventions: the SOS naming convention and the Apple II Pascal convention. More information about the Options menu is included in Chapter 2 of this manual.

The Options command selected has no impact on the file-naming convention that you may use when specifying a file to the system. You may use either the SOS or Pascal conventions: the option selected determines only which is displayed.

The SOS file-naming convention (described in detail in the Apple III Owner's Guide) is used for most of the examples in this manual and in the accompanying Apple III Pascal manuals. The Pascal file-naming convention and its use with certain Filer commands is described in Appendix B. We encourage you to read that section of the manual.

The overall relationship of the Filer and its commands is displayed in Figure 3-1 below. Each command is entered from the Filer command line, and after completing operation each command returns to the Filer command line.

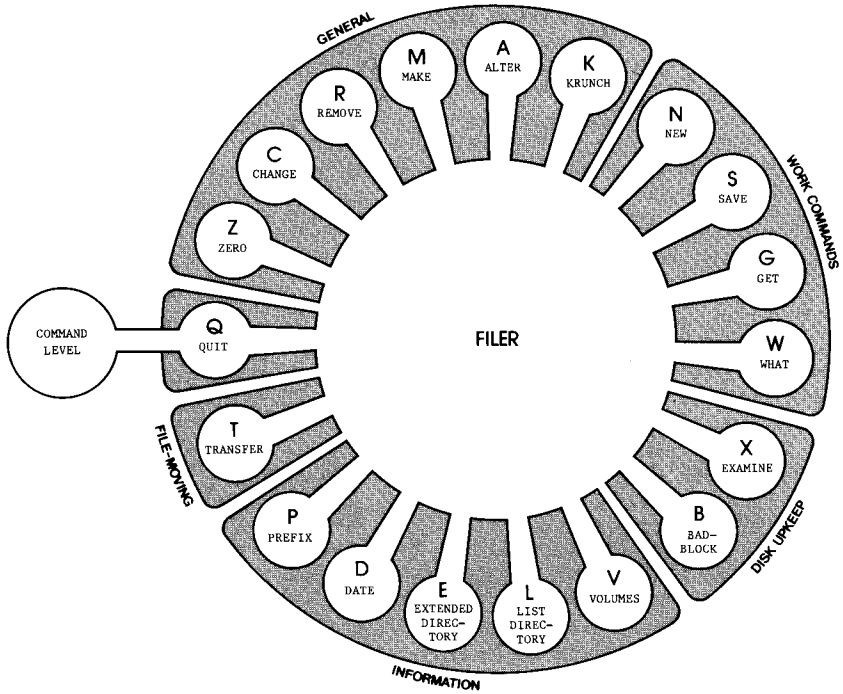


Figure 3-1. The Filer.

Diskfiles Needed by the Filer

The file SYSTEM.FILER must be in a disk drive before you type F to invoke the Filer. When the Filer prompt line appears, SYSTEM.FILER is no longer needed and the diskette containing it may be removed from the disk drive.

Using the Filer

To use the Filer, type F from the Command level. The following prompt line will appear at the top of your screen:

```
Filer: L(dir, E(xt-dir, R(em, T(rans, C(hng, M(ake, D(ate, P(refix, Q(uit, ? [A3/1.Ø]
```

Typing ? in response to this prompt displays more Filer commands:

```
Filer: V(ols, W(hat, N(ew, S(ave, G(et, K(rnch, Z(ero, Bad-b(lks, X(amine
```

To invoke any Filer command, type the first letter of the command that you want to use. For example, typing S invokes the Save command.

Any Filer command that requests a file specification allows you to specify as many files as you wish, by separating the file specifications with commas, and terminating this "file list" by pressing the RETURN key.

Commands using single pathnames will keep reading pathnames from the file list and using them until there are none left. Commands using pairs of pathnames (such as Change and Transfer) will take file specifications in pairs and operate on each pair until only one specification or none remains. If one pathname remains, the Filer will ask you for the second member of the pair. If an error is detected at any point in the list, the remainder of the list will be discarded.

If you press the RETURN key when the Filer prompts you for a filename, the command will be terminated and the Filer command line will be redisplayed.

To erase your response to a Filer prompt while leaving the prompt on the screen, type CONTROL-X. To abort a Filer prompt line and return to the Filer command line, type ESCAPE-RETURN .

General Information

This section contains basic information about the Filer's method of storing disk files. Note that this information applies expressly to systems using 28Ø-block floppy diskettes. Diskette drives (or hard-disk devices) having greater capacity will operate in the same general fashion, but the numbers quoted in the descriptions below will be different for each type of drive.

The Filer stores information on a diskette in 35 concentric tracks. The disk drive's read/write head can be moved in and out to position it over any of these 35 different tracks on the spinning diskette.

Each track on the diskette is divided into 16 sectors. Once the disk drive's read/write head is positioned over a given track, that track's 16 sectors will pass under the head, one after the other, each time the diskette spins around.

Each sector consists of an address field and a data field. The address field identifies which sector of which track the disk drive is about to access. The address fields are written on a diskette just once, when the diskette is formatted. The data field is the portion of each sector used for storing information. Up to 256 bytes of information can be stored in each sector's data field.

The Pascal system stores information in two-sector units called blocks, each containing 512 (also called "1/2 K") bytes of information. Each of a diskette's 35 tracks can thus store eight blocks of information, for a total diskette storage capacity of 28Ø blocks (14Ø K bytes). Although the Filer handles all information storage automatically, low-level routines for storing and retrieving diskette information are also available. (See the Apple III Pascal Programmer's Manual for details).

Not all tracks on a diskette are available for storing your programs or files. Blocks Ø and 1 are reserved for the program that starts up the system. In addition, every disk contains a directory enabling the system to find information stored on that disk. The directory begins on block 2 of the diskette.

In general, the system will begin storing a program or text file wherever it can find an unused block on the disk. When the block is filled, the system finds another free block, perhaps on another track, and continues to record information there. This process continues until the entire file has been stored.

Because files are not stored in contiguous blocks, it is not necessary to "crunch" or consolidate the files stored on a disk. Extensive movement of files on a disk may, however, result in excessive file fragmentation. If this happens, you will notice that the system is operating unusually slowly.

The easiest way to remedy such a situation is to use the Filer's Transfer command to move each file individually to a relatively empty diskette. Because each file will be transferred as a single unit, they will not be fragmented once they have been transferred to the new diskette. For specific information on transferring files, see the section of this chapter on the Transfer command.

If a file is updated (saved again under the same pathname), the new version is saved and verified before the old version is removed. This uses more space on the disk, but guarantees that at least one version of your file is intact on the disk at all times during the saving process. The only command that does not work this way is the Editor's Save command. This command allows you the option of removing the old version of your file before writing the new version, in case you are short on disk space. If you do not choose this option, the new version of the file is written first, as described above.

Device Names and Numbers

A device is something connected to your Apple III to send and/or receive data. A SOS device driver is a program enabling the Apple to communicate with a device. For more information on SOS device drivers, refer to the Apple III Standard Device Drivers Manual.

A device can be referred to by its device filename. A device filename consists of a period followed by the name of the device. Thus, a printer's device filename is .PRINTER . The Apple's built-in disk drive has the device name .D1 ; additional disk drives configured into the system are named .D2 , .D3 , and .D4 respectively.

A block device, such as a diskette, may either be referred to by its device filename or by the volume name of the diskette stored in the appropriate drive. Volume names can be composed of up to 15 letters, numbers, and periods and may not begin with a number or a period. Volume names are always preceded by a slash (/). If you wanted to refer to a diskette named MOOSE in the built-in drive, you could refer to it either as /MOOSE or as .D1 .

Apple III Pascal assigns a number to each device that has been properly configured into the system. All devices are put into one of two categories: standard devices (assigned numbers 1 through 12) and user devices (assigned numbers 128 through 143).

The following table shows standard device numbers, device names, and volume names used by Apple III Pascal.

Apple III Pascal Standard Device Number	SOS Device Name	SOS Volume Name	Description of Device
1	.CONSOLE	—	Screen and keyboard with echo
2	.CONSOLE	—	Screen and keyboard without echo
3	.GRAFIX		Graphics
4	.D1	*disk name	Built-in disk drive
5	.D2	*disk name	2nd disk drive (external drive)
6	.PRINTER	—	Printer
7	.RS232	—	Remote input
8	.RS232	—	Remote output
9	.D3	*disk name	3rd disk drive (external drive)
1Ø	.D4	*disk name	4th disk drive (external drive)
11	.D5	diskname	Disk drive or other block-structured device
12	.D6	diskname	Disk drive or other block-structured device

* Disk or diskette.

Table 3-1. Device Names and Numbers, Volume Names

The assignment of device numbers to devices generally works this way: At boot time the system checks all device drivers, and checks to see if the associated device is a standard device. If it is, the device is assigned the appropriate standard device number. (The standard devices and their associated device numbers are listed on the table above.)

If the system finds that a device is not a standard device, it considers it a user device and assigns it a user device number. User device numbers are assigned consecutively, beginning with number 128, as they are encountered by the system.

A more complete discussion of the method used by Apple III Pascal to assign device numbers is included in Appendix D.

One of the benefits of the Pascal file naming convention is that it lets you use device numbers to specify devices. To do this, you precede the device number with a number (#) sign and follow the device number with a colon. For example, you can refer to the printer as #6: or you can refer to the diskette in the built-in drive as #4: .

More About Files

This section includes more information about files, including the structure of directories and subdirectories and the use of wildcard characters in specifying a set of files stored on disk.

Directories

Every formatted disk has a root directory, starting in block 2, that is a "table of contents" of the files on that disk. Every disk also has a name, called a root volume name, that is the same as the name of the root directory. A "root directory" can contain a maximum of 51 files.

When you use the Apple III Utilities Diskette to format a disk, the newly formatted disk is assigned a volume name that you choose and an empty directory with that name. To change the name of a disk's root volume name (and hence the name of the disk directory), you can use the Filer's Change command described later in this chapter.

Each time a file is stored on a disk, information about that file is automatically entered into the disk directory. The List directory and Extended directory list commands make it possible to view the information stored in a specified volume directory. These commands are explained later in this chapter.

Never issue commands when two disks with the same volume name (and hence the same directory name) are available to the system. If you do, the system may operate on the wrong disk (usually the disk in the higher-numbered drive). If the operation involves updating the disk's directory, the system may store the wrong disk's directory onto your disk, making the files originally on that disk unusable. The same problem may occur if you replace the diskette in a drive by another diskette with the same volume name. The exception to this rule is in certain Filer transfer operations.

Local Filenames

Every file used by Apple III Pascal has its own name called a local filename. The rules for forming local file names are the same as the rules for naming volumes. (This is because a volume name is itself a local file that refers to the root directory file.)

A local filename can be composed of up to 15 letters, numbers, and periods and may not begin with a number. With the exception of a device filename, local filenames may not begin with a period. Most local filenames used by Apple III Pascal end with a suffix (most often .TEXT or .CODE) that specifies the kind of information stored on the file.

You refer to a particular file on a disk by its pathname: a / character, followed by the filename of the disk's directory, another slash, and the local filename you are referring to. For example, if you wanted to refer to a file named MYFILE on disk MYDISK , the complete file specification would be /MYDISK/MYFILE .

A file specification can also begin with the device file name of the disk drive in which the appropriate diskette resides. Thus, if diskette MYDISK was stored in the built-in drive, another way of specifying file MYFILE would be .D1/MYFILE . If you wanted to refer to a file with the local filename AGAIN.CODE that was stored on diskette /DO.IT , in the first external drive, you could refer to it either as .D2/AGAIN.CODE or as /DO.IT/AGAIN.CODE .

A complete file specification is often referred to as a pathname because it specifies the path the system must follow to gain access to a particular file. Refer to the Apple III Owner's Guide for a description of pathnames.

Subdirectories

Subdirectories are used to create a hierarchical system for storing files on disk. The Make command described later in this chapter is used to create subdirectories.

Subdirectories follow the same file-naming rules as other local files. Suppose, for example, that you had a disk named PROGRAMS containing both old and new versions of several program files. You might then create one subdirectory on the disk called OLDSTUFF that included all of your old program files and another subdirectory on the same disk called NEWSTUFF that included all of your revised program files. If you wanted to refer to the old version of the file named SORT.CODE , you would use the pathname specification /PROGRAMS/OLDSTUFF/SORT.CODE ; if you wanted to refer to the new version of the file named SORT.CODE , you would use the pathname specification /PROGRAMS/NEWSTUFF/SORT.CODE .

Subdirectories are most often used on large storage media such as hard disk drives although they also can be used on smaller capacity diskettes. There are no rules governing the number of subdirectories that can be created on a disk although a pathname specification is limited to a maximum of 80 characters, a limit set by the Filer.

For example, the pathname /BIG/BIGGER/BIGGEST/BIGGERSTIL.TEXT refers to a file whose local filename is BIGGERSTIL.TEXT . BIGGER is a subdirectory on disk BIG ; BIGGEST is a subdirectory that is a part of subdirectory BIGGER . The pathname /BIG/BIGGER/BIGGEST/NOTSOBIG.TEXT refers to another textfile in the subdirectory /BIG/BIGGER/BIGGEST .

Subdirectories vary in size according to the number files contained under them. The minimum size for a subdirectory is one block, and it may contain 12 files. Each succeeding block allows an additional 13 files to be added to the subdirectory.

Diskette File Types

The system automatically assigns a file a type when it is created, based on the file's suffix (the characters following the final period in the file's local filename). The most common suffixes are .TEXT for files containing text (natural language, Pascal programs or assembly-language programs) and .CODE for files containing the compiled version of a program. A file's type is displayed by the List directory and Extended directory commands.

Following is a list of the filetypes recognized by the system, the suffix associated with that type, and the way that type is referred to in the displays created by the List directory and Extended directory commands.

Suffix	File Type	Extended directory listing List directory listing
.TEXT	Human-readable text	Textfile
.CODE	Machine-executable code	Codefile
.DATA	Data	Datafile
.ASCI	ASCII character stream	Asciifile
.BAD	Used only with UCSD-formatted diskettes; see the Xamine command in Appendix B.	Bad file

Table 3-2. Disk File Types

BASIC text files are listed on the disk directory as ASCII files even though they may not always have a suffix.

Files referring to a disk directory or subdirectory are listed by the Extended directory and List directory commands as being of type Directory even if they have no suffix.

For more information concerning the internal format of different types of files, see Appendix A.

The filename suffix may be omitted with some commands when entering a file name. The explanations of individual commands later in this chapter tell you when a suffix must be included when specifying a pathname and when the system automatically adds the appropriate suffix for you.

Sometimes (after using the Change command to change a file's name, for example) a file's actual type may not agree with its local filename suffix. The actual type of the file can always be determined by examining the filetype column of the List directory and Extended directory displays.

Wildcards

Wildcards enable you to specify a whole set of files at once. The Filer performs the requested action on all files whose pathnames are included in the set specified. The form of a wildcard specification is as follows:

<string1>=<string2> or <string1>?<string2>

where <string1> and <string2> are set-specifying strings. Either string may be a null string. The set-specifying strings indicate the portion of a pathname that may not be ignored. The wildcard characters, equal sign (=) and question mark (?), stand for any sequence of characters in a pathname that can be ignored. For example, the wildcard specification

/MYDISK/DOC=TEXT

tells the Filer to perform the requested action on all files on disk MYDISK whose local filenames begin with the string DOC and end with the string TEXT . If a question mark is used instead of an equals sign

/MYDISK/DOC?TEXT

the Filer pauses and requests verification before acting on each file in the specified set. At each pause, you may type a "Y" for Yes, "N" for No, or press the ESC key to return to the Command level of the Filer.

Wildcards may be used only when specifying local filenames, and are accepted only by the Alter, List, Extended-directory, Transfer, Change, and Remove commands. A command requiring two filenames demands that both use wildcards if one of the filenames does, except when you are transferring the file or files to a non-block-structured device.

EXAMPLE:

Suppose the disk directory for the disk named MYDISK contains the following files:

```
NEW
MEADOW.TEXT
USELESS.CODE
MEADOW.CODE
NEVERMORE.TEXT
GURUS
```

After typing R for Remove, you will see this message:

```
Remove what file ?
```

Response 1: /MYDISK/N=

Typing this response generates the message:

```
Volume /MYDISK
file NEW removed
file NEVERMORE.TEXT removed
Update directory ?
```

At this point you can type Y to remove all the files listed, or you can type N , in which case the Remove command will abort and the files will not be removed from the disk directory. This gives you one last chance to change your mind before removing the files permanently from your directory.

Response 2: /MYDISK/N?

Typing this response generates the message:

```
Volume /MYDISK
Remove NEW ?
```

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

```
Remove NEVERMORE.TEXT ?
```

Again you may type a response (Y or N), and if you have given any Y responses, the Filer asks:

Update directory ?

As with the previous pattern, this gives you one last chance to change your mind before the files are finally removed.

EXAMPLE:

Again, suppose you have a disk MYDISK with the same directory as in the previous example. After typing L for L(dir (meaning List the disk's directory), you will see this message:

Directory listing of what volume ?

Response: /MYDISK/=TEXT

Typing this response causes the Filer to list the files MEADOW.TEXT and NEVERMORE.TEXT since these are the only files on the disk ending in TEXT .

Wildcards cannot be used to refer to directory or subdirectory names.

Only one wildcard may be used in a pathname specification. The specifications:

/MYDISK/DO?TE?T or D1/=TE=

results in the message:

Illegal wildcard

The Filer commands Transfer and Change both require two file specifications. If the first specification contains a wildcard the second specification must also contain a wildcard. If you forget, you will be given the message

Wildcard not allowed

The only legal exception to this rule occurs when the Transfer command is given the character $ as its second pathname specification.

Either or both of the set-specifying strings may be omitted. For example, a local filename set specification such as =TEXT or

DOC= or even just = is valid. This last case, where both set-specifying strings are omitted, causes the Filer to perform the requested action on every file in the specified disk's directory.

This feature can sometimes be used to act on a file whose local filename is not "recognized" by Filer commands because of illegal characters in the filename, or a slightly damaged directory.

Set-specifying strings may not "overlap". If a character appears in the set-specifying string, that same character must appear in the target string in the same relative position, or no match occurs. The = or ? characters allow any character or sequence of characters to be considered a valid match. For example, the specification GOON=NS would not include the local (pointless) specification for the file GOONS . The specification GOON=NS contains an explicit (non-wildcard) character, in this case an extra "N", that does not occur in the filename GOONS.

The $ character saves the most recently-entered local filename for use as a wildcard, and can be used as part of a longer pathname.

Suppose you want to transfer a file from one directory to another. Using the Filer's Transfer command (described later in this chapter), you could transfer the file named SALAMI from the directory /LUNCHMEATS to the subdirectory /SANDWICH/FILLINGS by typing

/LUNCHMEATS/SALAMI,/SANDWICH/FILLINGS/$

rather than the longer

/LUNCHMEATS/SALAMI,/SANDWICH/FILLINGS/SALAMI

The Filer Commands

The rest of this chapter includes detailed descriptions of each of the Filer's commands listed by function. The section on Information Commands describes those Filer commands allowing you to see what peripherals are connected to your system and to

examine the contents of directories. The section on the Transfer command explains how to move files from one part of the system to another. The section on General Disk File Commands includes a discussion on creating and removing files. The final sections describe commands used to manipulate workfiles and to check diskettes for damage.

Filer commands requiring you to enter a pathname or some other information can be aborted by pressing RETURN instead of the requested information. If you have started entry of the information called for by the command, you can still abort the command by pressing the ESCAPE key. (An exception to this is the Remove command when used with a wildcard.)

Information Commands

Volumes

The Volumes command lists the volume names, device names and device numbers of all input and output devices whose drivers have been configured into the system. This command is invoked by typing V from the Filer command line.

The naming convention used by the Volume command depends on the setting of option C you have chosen from the Command level Options command. If the SOS convention is used, all device and volume names are shown according to the SOS filename convention, if the Apple II Pascal convention is chosen, it will be used. For information about interpreting the Volumes command's display see Appendix B.

A four-drive system, with a QUME-type printer, a SILENTYPE printer, and an RS232 driver for communicating over telephone lines, would give a Volumes display like this:

```
Volumes and devices on-line:
  1   .CONSOLE
  2   .CONSOLE
  4   .D1 (/PASCAL1)
  5   .D2 (/PASCAL2)
  6   .QUME
  7   .RS232
  8   .RS232
  9   .D3 (/MEMOS)
 1Ø   .D4 (/ACCNTINGPROG)
128   .SILENTYPE
System volume is /PASCAL1
Prefix is  /ACCNTINGPROG/REVISION1
```

In this example, the Pascal system diskette is PASCAL1 , in device .D1 . Usually the prefix will be the name of the Pascal system diskette. In this example, the prefix has been changed by the Prefix command to /ACCNTINGPROG/REVISION1 , the name of a subdirectory on the diskette ACCNTINGPROG located on device .D4 . Block-structured devices such as diskette drives are indicated by a root volume name in parentheses.

The first two devices listed are controlled by the .CONSOLE device driver. When the system reads from Device #1, characters are echoed on the screen as they are read. When the system reads from Device #2, characters are not echoed. More information on reading from the console is included in the Apple III Pascal Programmer's Manual.

List Directory

The List directory command gives detailed information about a specified directory or subdirectory and its contents. The command is invoked by typing L from the command level of the Filer.

The list created by the List directory command includes the local filenames of each file that is a part of the listing as well as (in this order) the number of logical blocks each file contains, the date the file was most recently modified, the time the file was most recently modified (if available), the file type, the number of bytes in the last block of the file, and the number of physical blocks used by the file. The last line of the List directory display contains the number of files

included in the directory or subdirectory listed and the number of free blocks available on the disk.

Write-protected files are indicated by an asterisk preceding the local filename. Listings of directories on diskettes formatted by the Apple II Pascal formatter say "AppleII" immediately after the volume name.

A directory listing stops when it has filled the screen. Press the spacebar to continue the listing, or press the ESC key to abandon the listing and return to the Filer command line.

The information displayed by the List directory command differs depending upon which file display convention you have chosen with the Option command's option C: the Apple II Pascal or SOS filename convention. All examples in this section assume that you have selected the SOS convention. For information about interpreting the List directory command's display if you have chosen the Apple II Pascal format, see Appendix B on Pascal format filenames.

The listing generated by the List directory command also changes slightly when an Apple II Pascal diskette is accessed. For a full explanation, see Appendix B.

The List command is often used to list the contents of a diskette directory on the screen. The following display shows a directory listing for a disk named RECORDS .

```
Prompt:    Directory listing of what volume ?

Response:  /RECORDS

/RECORDS             Size  Modified  Time  File type   Eof Phys
  LETTERS.TEXT         32   4-May-81  5:55  Textfile    512   33
  BILLING               2   6-May-81  2:15  Directory   512    3
 *ACCOUNTS.DATA        36  14-May-81  2:15  Datafile    512   37
  WAREHOUSE             2   4-Jun-81  4:41  Directory   512    3
  MINUTES.TEXT         34  28-Jun-81  8:15  Textfile    512   35
 5 files listed, 128 blocks available
```

The List directory command only displays the level of files one level below the directory or subdirectory specified. Thus, based on the listing shown above, we know that the top level of files included in the directory RECORDS contains two text files (LETTERS.TEXT and MINUTES.TEXT), one data file (ACCOUNTS.DATA)

and two subdirectories, BILLING and WAREHOUSE . The asterisk next to ACCOUNTS.DATA indicates that that file is write-protected which means that the file can't be overwritten, removed or destroyed by any Filer operation, except copying an entire volume.

Notice that the listed number of available blocks is based on the number of blocks occupied by all the files contained on the disk and not just on the number of blocks used by the files included in the directory listing.

Usually, the number of logical blocks that a file occupies will be different from the number of physical blocks the file occupies (e.g. /RECORDS/ACCOUNTS.DATA). This occurs because more space is allocated for the file when it is created than the file actually uses. Some of the physical space taken up by the file is used to hold file structure information in addition to the data held by the file.

Now suppose you want to find out what files are contained in the subdirectory WAREHOUSE of the diskette shown above.

```
Prompt:    Directory listing of what volume ?

Response:  /RECORDS/WAREHOUSE

  /RECORDS/WAREHOUSE Size  Modified  Time  File type   Eof Phys
    INVENTORY.TEXT     18   4-Jun-81  4:41  Textfile    512  19
    INVENTORY.CODE      5  14-Jun-81  5:12  Codefile    512   6
    PRODUCTS            2   6-Jun-81  4:41  Directory   512   3
   3 Files listed, 128 blocks available
```

If you wanted to list the contents of the subdirectory /RECORDS/WAREHOUSE/PRODUCTS the transaction would go as follows:

```
Prompt:    Directory listing of what volume ?

Response:  /RECORDS/WAREHOUSE/PRODUCTS

/RECORDS/WAREHOUSE/PRODUCTS
                     Size  Modified  Time  File type   Eof Phys
    NUTS.DATA          18   4-Jun-81  4:41  Datafile    512  19
    BOLTS.DATA          5  14-Jun-81  5:12  Datafile    512   6
    SCREWS.DATA         2   6-Jun-81  4:41  Datafile    512   3
   3 Files listed, 128 blocks available
```

You can list any portion of a directory or subdirectory with the "wildcard" option. For example, suppose that you want to list all of the textfiles included in the directory RECORDS.

Prompt: Directory listing of what volume ?

Response: /RECORDS/=.TEXT:

Typing this response would generate the following display:

```
/RECORDS          Size  Modified  Time  File type   Eof Phys
  LETTERS.TEXT     32   4-May-81  5:55  Textfile    512   33
  MINUTES.TEXT     33  28-Jun-81  8:15  Textfile    512   34
 2 files listed, 128 blocks available
```

Notice that because the List directory command displays only one level of a directory, files ending in .TEXT that are contained within subdirectories are not displayed.

When you use the List directory command, the system sends the listing to the console. You can, however, write the directory or subdirectory (or any portion of it) to a device other than .CONSOLE by giving both a source directory pathname (the name of the directory or subdirectory to be listed) and a destination pathname (where you want the directory listing sent). For example:

This List directory transaction sends a subset of a directory to a printer.

Prompt: Directory listing of what volume ?

Response: /RECORDS/WAREHOUSE/PRODUCTS,.QUME

Typing this response causes this display

```
/RECORDS/WAREHOUSE/PRODUCTS
                  Size  Modified  Time  File type   Eof Phys
  NUTS.DATA         32   6-Jun-81  5:17  Datafile    512   33
  BOLTS.DATA        24   9-Jun-81  5:38  Datafile    512   25
  SCREWS.DATA       21  22-Jun-81  4:17  Datafile    512   22
 3 files listed, 128 blocks available
```

to appear on the printer (assuming that you have a printer whose driver has been properly configured into the system and that the printer is turned on and ready to get data).

A source file specification consists of a directory or subdirectory name and optional subset-specifying strings containing wildcards. A destination file specification consists of a device or file name. If the volume is a disk, you must include a local filename. The source file specification must be separated from the destination file specification by a comma and the destination file specification cannot include a wildcard.

This List directory example involves writing the directory to a diskette file:

Prompt: Directory listing of what volume ?

Response: .D2,.D1/DIRECTORY.TEXT

After typing this response, you will see the message

```
WRITING.........
```

as the Filer lists the directory of the disk in disk drive .D2 onto a file called DIRECTORY.TEXT on the diskette in the built-in drive.

Extended Directory

The Extended directory command provides detailed information about each file stored on the specified directory or subdirectory. This command is invoked by typing E from the command level of the Filer.

The only difference between the List directory and Extended directory commands is that the List directory command displays only one level of files beneath the specified directory or subdirectory while the Extended directory command displays all files that are a part of the specified directory or subdirectory.

The information displayed by the Extended Directory command depends on your choice of filenaming convention chosen with the Option command's option C. For more information on filenaming conventions used with Apple III Pascal, see Appendix B in this manual.

The Extended directory display changes slightly when an Apple II-format diskette is accessed. See Appendix B for information about the Extended directory list generated for an Apple II-format diskette.

The most frequent use of the Extended directory command is to list an entire diskette directory. The following example refers to the same diskette directory that was used in the List directory examples in the previous section.

Prompt: Directory listing of what volume ?

Response: /RECORDS

```
/RECORDS             Size  Modified   Time   File type   Eof Phys
  LETTERS.TEXT         32   4-May-81 11:23   Textfile    512  33
  BILLING               2   6-May-81  7:23   Directory   512   3
   ACCOUNTS.TEXT       38   6-May-81 1Ø:56   Textfile    512  39
   ACCOUNTS.CODE        6   6-May-81  1:15   Codefile    512   7
  *ACCOUNTS.DATA       36  14-May-81  1:Ø3   Datafile    248  37
  WAREHOUSE             2   4-Jun-81 12:38   Directory   512   3
   INVENTORY.TEXT      18   4-Jun-81  3:15   Textfile    512  19
   INVENTORY.CODE       5   4-Jun-81         Codefile    512   6
   PRODUCTS             2   4-Jun-81         Directory   248   3
    NUTS.DATA          32   9-Jun-81  4:Ø7   Datafile    512  33
    BOLTS.DATA         24   9-Jun-81  9:56   Datafile    512  25
    SCREWS.DATA        21  22-Jun-81         Datafile    512  22
   MINUTES.TEXT        33  28-Jun-81  2:Ø4   Textfile    512  34
 13 files listed, 128 blocks available
```

The prompt lines, syntax, and wildcard options are exactly the same for the Extended directory command as for the List directory command discussed previously. For more details and examples, see the description of the List directory command.

General File-Moving Command

All file moving is accomplished by one Filer command. Note that the file itself is never actually moved from the disk: a copy of the contents of the file is moved, either to another part of the disk or another volume in the system.

Transfer

The Transfer command is used to copy the contents of one disk or one or more files from one disk to another. To use the Transfer command, type T from the command level of the Filer.

The Transfer command can be used for many purposes including:

copying individual files from one diskette to another

copying the contents of a subdirectory

copying the contents of an entire disk or diskette

copying files to and/or from a device such as a printer or the console

The Transfer command requires you to supply two file specifications, one for the source file (the file being copied) and one for the destination file, (the place the file is being copied to) separated by either a comma or a RETURN . Wildcards are permitted.

You should avoid having two disks on the system at the same time with the same volume name. The most common exception to this rule is made when you want to back up a disk's files.

The Filer's Transfer command does allow the source and destination volumes to have the same name, but resticts this use to transfers of an entire disk, transfers to and from an Apple II Pascal-format disk, or when the file being transferred is small enough to fit entirely in the Filer's buffer space.

The idea of making back up copies of all your work can hardly be over-emphasized: It should never be necessary for you to have to spend hours or weeks recreating some piece of work that was lost to spilled coffee on a diskette. The Filer's Transfer command make backing up disks too easy for you to be able to afford the luxury of no backups. An example of making a backup file is given below.

EXAMPLE:

Suppose you want to transfer the file STARGAZER.TEXT from diskette MYDISK to diskette BACKUP .

Prompt: Transfer what file ?

Response: /MYDISK/STARGAZER.TEXT

When you press the RETURN key, the system checks to be sure that the specified source diskette is in one of the disk drives. If MYDISK is not in any drive, you will see the message

```
/MYDISK - Volume not found
```

If the source diskette is found in a drive, the system then checks to be sure the specified file is on that diskette. If the diskette MYDISK contains no file named STARGAZER.TEXT , you will see the message

```
File not found
```

In either case, you will be returned to the outer Filer level. Just insert the correct source diskette in any drive and type T again.

Let's assume the system succeeds in finding the source diskette and file. The dialogue continues, asking you to specify the destination for the transfer:

```
Prompt:    To what file ?

Response:  /BACKUP/TEMP.TEXT
```

You could also have given both source and destination specifications in the first response, separated by a comma.

When you press the RETURN key, the system checks to be sure the destination diskette is in a disk drive. If it is, the transfer begins. If it is not, there is a pause; then you are prompted:

```
Insert destination disk
     Type <space> to continue
```

Put the correct destination diskette in any available drive and press the spacebar. If, at this or any other point in the Transfer process, you want to return to the Filer command line, press ESCAPE.

When the transfer is complete, the Filer gives you the message

```
/MYDISK/STARGAZER.TEXT ---> /BACKUP/TEMP.TEXT
```

The Filer has made a copy of STARGAZER.TEXT as found on the diskette named MYDISK , and has stored that copy on the diskette BACKUP under the local filename TEMP.TEXT .

If, in the above example, you had wanted to save the file STARGAZER.TEXT on diskette BACKUP under the same local filename STARGAZER.TEXT, you could have done the following:

```
Prompt:     Transfer what file

Response:   /MYDISK/STARGAZER.TEXT,/BACKUP/$
```

Once the Filer has been summoned, it resides entirely in the computer's memory. Thus, you can summon the Filer, and then remove all system diskettes from the drives in order to use both drives for source and destination diskettes during a transfer. Just remember to replace the Pascal system diskette in the built-in drive when using the Quit command to exit from the Filer.

If you give the same block device name for both source and destination file specifications, the system assumes you are doing a single-drive transfer and are going to change diskettes in that drive. You will see the message

```
Insert destination disk
     Type <space> to continue
```

Files may be transferred to a device other than a disk by specifying a device such as .CONSOLE (for a quick screen listing of a file) or .PRINTER (to print a file) as the destination file.

EXAMPLE:

```
Prompt:    Transfer what file ?

Response   .D2/STARGAZER.TEXT

Prompt:    To where ?

Response:  .PRINTER
```

Typing this response causes the file STARGAZER.TEXT , on the diskette in disk drive .D2 to be sent to the printer (assuming a printer is properly connected and the printer driver has been properly configured to your system).

You can also transfer from an input device other than a diskette, such as the keyboard. A local filename accompanying a non-disk volume name or number is ignored.

EXAMPLE:

Prompt: Transfer what file ?

Response: .CONSOLE

Prompt: To where ?

Response: .PRINTER

After these responses, you can use your keyboard as a typewriter. Nothing will appear on the printer until you type the "End-Of-File" character, CONTROL-C (Note that some printers may accept a CONTROL-C as a command; if yours does, you will have to press the RETURN key before pressing CONTROL-C). Then all you have typed will be sent to the printer.

The wildcard specification is allowed in the Transfer command. When using wildcards, the set-specifying strings in the source pathnames will be replaced by the respective strings (called "replacement strings") in the destination pathnames. The portion of each source pathname accounted for by the = or ? wildcard character is reproduced unchanged in the corresponding destination filename. Remember that the Filer considers the one-character, wildcard-alone file specification (= or ?) to be equivalent to specifying all files in the directory.

EXAMPLE:

Suppose the Prefix directory MYDISK contains these files:

```
PAUCITY
PARITY
PENALTY
```

Further, suppose the destination diskette is named ODDNAMES

Prompt: Transfer what file ?

Response: P=TY,/ODDNAMES/V=S

Typing this response would cause the Filer to reply

```
/MYDISK/PAUCITY ---> /ODDNAMES/VAUCIS
/MYDISK/PARITY ---> /ODDNAMES/VARIS
/MYDISK/PENALTY ---> /ODDNAMES/VENALS
```


EXAMPLE:

Suppose the Prefix directory MYDISK contains these files:

```
CHAP11.TEXT
CHAP12.TEXT
CHAPTER13.TEXT
CHAP14.TEXT
```

Prompt: Transfer what file ?

Response: C=XT

Prompt: To where ?

Response: /BACKUP/OLDC=XT

Typing these responses would cause the Filer to reply:

```
/MYDISK/CHAP11.TEXT ---> /BACKUP/OLDCHAP11.TEXT
/MYDISK/CHAP12.TEXT ---> /BACKUP/OLDCHAP12.TEXT
/MYDISK/CHAPTER13.TEXT ---> not processed
/MYDISK/CHAP14.TEXT ---> /BACKUP/OLDCHAP14.TEXT
```

On the third attempted transfer, the destination filename would have been OLDCHAPTER13.TEXT, which exceeds the 15-character limit for local filenames. Therefore, that file was "not processed".

Using the single character = as the destination pathname specification has the effect of replacing any set-specifying strings in the source specification with nothing.

A brief reminder: in any wildcard specification, the single character ? may be used in place of = . The only difference is that a ? in either specification (or both) causes the Filer to ask you for verification before each file is transferred.

A source or a destination file specification may contain only one wildcard character. A specification such as

```
/MYDISK/?UGH?
```

is not a legal specification. An attempt to use such a specification as either the source or the destination of a transfer will cause the message

```
/MYDISK/?UGH?  - Illegal wildcard
```

If the source file specification contains a wildcard character, and the destination device is a diskette, then the destination file specification must also contain a wildcard character.

EXAMPLE:

Suppose the diskette MYDISK contains the following files:

```
CHAPTER1.TEXT
CHAPTER14B.TEXT
INTRO.TEXT
```

Further, suppose you want to transfer the files CHAPTER1.TEXT and INTRO.TEXT to the diskette BACKUP , retaining the same file names on the backup diskette.

```
Prompt:    Transfer ?

Response:  /MYDISK/?.TEXT,/BACKUP/$
```

Typing this response would cause the screen to clear, and then the following message would appear:

```
Transfer CHAPTER1.TEXT ?
```

Since you want to transfer CHAPTER1.TEXT , type a Y for "Yes". A copy of the file CHAPTER1.TEXT is then transferred from MYDISK to BACKUP . The Filer then proceeds to ask if you want to transfer the next file whose name ends in .TEXT . The complete dialogue might appear as follows:

```
Transfer CHAPTER1,TEXT ? Y
/MYDISK/CHAPTER1.TEXT ---> /BACKUP/CHAPTER1.TEXT
Transfer CHAPTER14B.TEXT ? N
Transfer INTRO.TEXT ? Y
/MYDISK/INTRO.TEXT ---> /BACKUP/INTRO.TEXT
```

Instead of a "Y" or "N" response, you may press the ESC key. This will return you to the command level of the Filer.

Copying an Entire Disk

You can use the Transfer command to copy the contents of an entire disk. When using the Transfer command in this way, you will respond to the Transfer command's prompts with either a volume name or a block device name. This operation erases the previous contents of the destination disk. After

copying, the destination disk becomes an exact, literal copy of the source disk and has the same directory name as the source disk.

EXAMPLE:

Suppose you want an extra copy of the diskette MYDISK and you are no longer interested in keeping the contents of diskette EXTRA .

```
Prompt:    Transfer what file ?

Response:  /MYDISK,/EXTRA

Prompt:    Transfer 28Ø blocks ? (Y/N)

Response:  Y
```

Each diskette used by the system contains at least 28Ø blocks. (Some diskette drives, as well as hard disks, can store more information. Check to see what your system is configured for.) To copy an entire diskette you will always type the response Y. If your system ever gives a message other than 28Ø blocks, the diskette's directory is either damaged or missing or you have a higher-capacity diskette or hard disk.

```
Prompt:    Destroy /EXTRA ?
```

If you type Y , the directory, and therefore your access to the contents of EXTRA , will be destroyed. The diskette named EXTRA will then become an exact copy of MYDISK , even having the same volume name. If you do not wish to destroy the contents of EXTRA type N and you will return to the command level of the Filer.

To copy a diskette using a two-drive system, you should invoke the Filer and then remove all system diskettes from the disk drives. This will enable you to use one drive for the source diskette and one drive for the destination diskette.

When the Transfer command is used to copy the contents of an entire diskette, the Filer transfers each block of information on the source diskette to the same location on the destination diskette. Thus, if files are fragmented on the source diskette, they will remain fragmented on the destination diskette.

If you transfer the contents of the Pascal system diskette to another volume the new volume will be invisible to the system, since it has the same name as the system disk. To copy the Pascal system diskette, always use the wildcard method described below.

A second method of transferring the contents of an entire diskette is to use the " = " wildcard option to transfer each file on the diskette. With this method, each file will be moved as a unit, thus eliminating any fragmentation of files which may be present on the source diskette.

Notice that this method does not destroy the contents of the destination diskette. Before executing this command, you may want to check the destination diskette to make sure that there is adequate space to copy the contents of the source diskette. Note that the transfer of files continues until there is no room on the destination diskette. When this occurs, the message

No room on volume

appears next to the name of the file that was unsuccessfully transferred.

EXAMPLE:

Suppose you want to transfer all of the files on diskette THIS to diskette THAT .

Prompt: Transfer what file ?

Response: /THIS/=, /THAT/=

Messages will then appear, listing the files that have been copied.

Copying a Subdirectory

The Transfer command can also be used to transfer the contents of a subdirectory. Like the full diskette transfer using wildcards described above, the transfer of a subdirectory from one disk to another eliminates file fragmentation at the destination.

When copying a subdirectory, the Filer first creates a new subdirectory on the destination disk and then transfers each of the files listed in the original subdirectory to corresponding files in the new subdirectory. This same process is applied to files within a subdirectory that are themselves subdirectories.

Two types of transfers are prohibited when dealing with subdirectories: Subdirectories cannot be copied over as a root volume directory, and root directories may not be copied to another volume as a subdirectory.

EXAMPLE:

Suppose the diskette MYDISK contained these files:

```
BALLOON.TEXT
BUBBLE
 TEMP.TEXT
 TEMP.CODE
 TEMP.DATA
TUBA.TEXT
```

and that you want to transfer the subdirectory BUBBLE to the diskette YOURDISK

Prompt: Transfer what file ?

Response: /MYDISK/BUBBLE

Prompt: To where?

Response: /YOURDISK/BUBBLE

You could have achieved the same result (and saved some typing) by using the dollar sign ($) as the destination filename.

"Data" diskettes, which are formatted but have no directory, may also be copied with the Transfer command, but operations with data diskettes involve only copies of the entire diskette contents.

When a file is opened to a data disk, transfer of information begins at block zero and continues to the last block of the disk. The sequential nature of data transfer to or from a data disk gives an operational resemblance to a file transfer to a device such as a printer.

General Disk File Commands

The following sections describe several commands that can be invoked from the Filer's command level.

Make

The Make command is invoked by typing M from the Filer's command level and is used to create subdirectories. It can also be used to reserve an area on a disk for a text or code file.

The Make command requires you to type a file name specification and gives you the option of typing a file size specification. The file size is specified by following the pathname with the number of blocks that the file will occupy enclosed in square brackets ([and]). The subdirectory will be assigned one block of space unless you specify a file size.

There are two default file size specifiers, [Ø] and [*]. Their meaning is different, according to the disk format being used. On SOS-format disks, both specifiers imply an initial file size of Ø blocks. On Apple II Pascal-format disks, [Ø] says the file is to occupy all of the largest unused contiguous area on the disk, while [*] means that the file is to occupy either the second-largest contiguous area or half of the largest area, whichever is larger. The difference in usage is explained by the fact that SOS-format files take up space on the disk only as required by their growth, instead of having to allocate contiguous blocks of space on the disk.

When using the Make command to create a subdirectory, you must follow the specified pathname with the directory-specifier character, which is an exclamation mark. This is the only place where the Filer recognizes the directory-specifier character.

The Make command can also be used to reserve an area on the disk for some future use. It also prevents use of that area by other files.

When a file other than a subdirectory file is created using the Make command and no file size is specified, the file is allocated zero logical blocks.

Files with filenames ending in .TEXT must occupy at least four blocks, and must occupy an even number of blocks (see this manual's appendix FILE FORMATS for details). An attempt to use the Make command to create a .TEXT file with fewer than four blocks results in the message "No room on vol". If you use Make

to create a .TEXT file specifying an odd number of blocks, the file will actually be made with one fewer block.

The Make command behaves somewhat differently depending upon whether the file you are creating is on a SOS-formatted or an Apple II Pascal diskette. For information about these differences, refer to Appendix B.

EXAMPLE:

Prompt: Make what file?

Response: /MYDISK/DAFFODIL.TEXT[28]

This response reserves 28 blocks on the volume MYDISK for the dummy file DAFFODIL.TEXT .

EXAMPLE:

Prompt: Make what file?

Response: /MYDISK/MYDIRECTORY![2]

This response creates the empty subdirectory MYDIRECTORY on the diskette MYDISK . The subdirectory will be two blocks long; if no length were specified, the length would be one block.

Apple III Pascal places no limit on the number of diskette subdirectories. If, for example, you had a diskette subdirectory whose pathname was /A/AA/AAA/AAAA , you could create a new subdirectory within that subdirectory this way:

Prompt: Make what file?

Response: /A/AA/AAA/AAAA/AAAAA!

Note that the longest pathname that can be specified to the Filer is 8Ø characters.

Change

The Change command is used to change the name of any file including a directory or subdirectory.

Changing the name of a diskette directory has the same effect as changing the diskette's volume name. You cannot change /PASCAL1, SYSTEM.PASCAL, or SYSTEM.FILER with the Change command. Also, changing a volume name to /PASCAL1 makes that volume invisible to the system.

This command requires two file specifications. The first specifies the name of the file whose name you want to change; the second specifies the file's new name. The first specification is separated from the second specification either by a comma or by pressing RETURN.

If you change the name of the Prefix directory or subdirectory, the directory or subdirectory name that the system supplies as the Prefix is also changed.

EXAMPLE:

The file MICKEY.TEXT is stored on diskette PLUTO .

Prompt: Change what file ?

Response: /PLUTO/MICKEY.TEXT

When you press the RETURN key, the dialogue continues:

Prompt: To what file ?

Response: MINNIE.TEXT

Typing this response changes the name of the file in the directory of diskette PLUTO from MICKEY.TEXT to MINNIE.TEXT . Note that you only have to include the local filename, as opposed to the complete pathname, in the second file specification.

Filetypes (such as TEXTFILE or CODEFILE) are originally determined by the local filename's suffix (such as .TEXT or .CODE). The Change command does not affect the filetype, but

it also does not automatically place any suffix after the new local filename. Consider the following example:

Prompt: Change what file ?

Response: /PLUTO/MICKEY.TEXT

Prompt: To what file ?

Response: MINNIE

In this case, the file is still present and now named /PLUTO/MINNIE , and MINNIE would still be listed as being of type TEXTFILE in an Extended directory list. Note, however, that the Get command (described later in this chapter) searches for the suffix .TEXT in order to identify a textfile as the workfile. Thus in order to successfully use the Get command to access the file MINNIE as the next workfile, you would have to change the name of the file from MINNIE to MINNIE.TEXT .

Wildcard specifications are legal in the Change command. If a wildcard character is used in the first file specification, then a wildcard must be used in the second file specification. The set-specifying strings in the first file specification are replaced by the analogous strings (referred to here as replacement strings) given in the second file specification. The Filer will not change the local filename if the change will result in the creation of a file whose name is too long (more than 15 characters).

EXAMPLE:

The diskette MYDISK contains these files:

```
CHAP14.TEXT
CHAP12.TEXT
CHAPTER13.TEXT
CHAP14.TEXT
APPNDX.TEXT
```

Prompt: Change what file ?

Response: /MYDISK/C=XT,/OLDC=XT

After you type this response (the two parts of the response were separated by a comma this time, but you could also press the RETURN key to separate the responses), the Filer indicates the

following name changes. Only the filenames are changed; the contents of the files themselves are left unchanged.

```
/MYDISK/CHAP11.TEXT ---> /OLDCHAP11.TEXT
/MYDISK/CHAP12.TEXT ---> /OLDCHAP12.TEXT
/MYDISK/CHAPTER13.TEXT ---> not processed
/MYDISK/CHAP14.TEXT ---> /OLDCHAP14.TEXT
```

In the third attempted name change, the "destination" filename would have been OLDCHAPTER13.TEXT, which exceeds the 15-character limit for filenames. Therefore, that file was "not processed". If all the destination filenames exceed 15 characters, none of the files will be processed.

The set-specifying strings may be empty, as may the replacement strings. The Filer considers the one-character file specification = (where both set-specifying strings are empty) to specify every file on the diskette.

EXAMPLE:

Prompt: Change what file ?

Response #1: /PLUTO/=,Z=Z

Typing this response causes every local filename on diskette PLUTO to have a Z added before the first character and after the last character unless the local filename was longer than 15 characters in length.

Response #2: /PLUTO/Z=Z,=

Typing this response causes the initial and terminal Z to be removed from each filename on diskette PLUTO that contains both an initial and a terminal Z.

A disk's volume name may be changed by specifying the current disk volume name or device name followed, after a comma or RETURN, by a new volume name.

EXAMPLE:

Prompt: Change what file ?

Response: /CENTIPEDE,/MILLIPEDE

Typing this response causes the system to give this message:

```
/CENTIPEDE ---> /MILLIPEDE
```

showing that the diskette named CENTIPEDE has been renamed MILLIPEDE . Note that what actually is happening here is that the diskette's directory name is being changed from CENTIPEDE to MILLIPEDE .

Similarly, the names of subdirectories can be changed by specifying an existing subdirectory name and (after a comma or RETURN) a new subdirectory name.

EXAMPLE:

Prompt: Change what file ?

Response: /FUNNY/FUNNIER,FUNNIEST

Typing this response would cause the system to give this message:

```
/FUNNY/FUNNIER ---> /FUNNY/FUNNIEST
```

showing that the subdirectory named /FUNNIER has been renamed /FUNNIEST .

The Change command ignores destination directory and subdirectory names if they are supplied. Thus FUNNIER is changed on volume /FUNNY even if the prefix is not set to /FUNNY .

EXAMPLE:

Prompt: Change what file ?

Response: /ALPHA/FUN,/BETA/NOFUN

This will not result in any change to the contents of the volume named BETA if it is on the system. In the same vein,

```
/ALPHA/FILE,/ALPHA/BETA/PRODUCE
```

ignores the fact that /BETA was even included in the specification.

Remove

The Remove command is used to remove file entries from a directory or subdirectory or to remove empty subdirectories. This command is invoked by typing R when the Filer prompt line is at the top of the screen.

Once files have been removed, they are no longer accessible to the user. While a removed file's contents are still stored on disk, the system acts as if the file has been erased from the disk and considers the area of the disk where the file was stored to be free for other files to write to.

The Remove command requires one file specification for each diskette file that you wish to remove. Wildcards are legal.

The Remove command should not be used to remove the current workfile. Instead, use the Filer's New command described later in this chapter.

You cannot remove a subdirectory until all of the files that are a part of that subdirectory have been removed.

EXAMPLE:

Suppose the Prefix diskette contains these files:

```
AARDVARK.TEXT
ANDROID.CODE
QUINT.TEXT
AMAZING.CODE
```

Prompt: Remove what file ?

Response: AMAZING.CODE

Typing this response tells the system to remove the file AMAZING.CODE from the Prefix diskette's directory.

Before actually removing the filenames of any files specified by the Remove command, the Filer asks if you want to

```
Update directory ?
```

Typing N aborts the command: You are returned to the Filer command level and no files are removed from the disk.

EXAMPLE:

Suppose your Prefix diskette contains the files shown in the previous example.

Prompt: Remove what file ?

Response: A=CODE

Typing this response causes the Filer to remove AMAZING.CODE and ANDROID.CODE from the Prefix diskette directory.

EXAMPLE:

Wildcards also can be used with the Remove command to banish some, or all, of the files in a given subdirectory. Suppose you have a diskette, MYDISK , that includes a subdirectory MEMOS . A listing of the subdirectory MYDISK/MEMOS looks like this:

```
/MYDISK
  MEMOS
     PETER.TEXT
     PAUL.TEXT
     MARY.TEXT
```

Let's say that you want to get rid of the files containing Peter and Paul's memos but that you want to leave the file containing Mary's memos where it is. After typing R for Remove, you will see this message:

Prompt: Remove what file ?

Response: /MYDISK/MEMOS/P=

Typing the above response generates the message:

```
Volume /MYDISK/MEMOS
File PETER.TEXT     removed
File PAUL.TEXT      removed
Update directory ?
```

Typing Y will remove the files containing Peter and Paul's memos but will leave the files containing Mary's memos, as well as the subdirectory MEMOS , intact. If you had wanted to get rid of all of the files in the subdirectory you could have done the following:

Prompt: Remove what file ?

Response: /MYDISK/MEMOS/=

Once this transaction was complete you could go on to remove the subdirectory MEMOS

Prompt: Remove what file ?

Response: /MYDISK/MEMOS

If the ? wildcard character is used, the systems checks with you before removing each file. A response of N causes the system to pass on to the next filename in the directory without acting on the previous one. A response of ESC causes the system to pass directly to the "Update Directory?" prompt.

Krunch

Krunch is only usable on Apple II-formatted disks.

The Krunch command consolidates unused space on a diskette. Because SOS stores text or programs in whatever empty blocks are available on a diskette regardless of whether the blocks are contiguous, it is not necessary to use the Krunch command with SOS-formatted diskettes. If you type K for Krunch and then, when prompted, type in the name of a SOS-formatted diskette, the system will respond with the message

```
Command not allowed on SOS format disk
```

For information about using Krunch with Apple II Pascal diskettes, see Appendix B.

Zero

The Zero command "erases" a specified directory or subdirectory by removing all files contained in it. This command is frequently used to remove all files contained in a hierarchy of subdirectories. The Zero command can also be used to "recycle" an entire diskette; the system forgets anything previously stored on the diskette and the diskette is ready to be used again.

To invoke the Zero command, type Z from the Filer command level.

This command is <u>not</u> the same as the Apple II Pascal Zero command.

EXAMPLE:

Suppose you want to forget all information stored on a diskette named OLDDISK, in drive .D2 so that you can re-use it as a clean, blank diskette.

Prompt: Zero what volume ?

Response: .D2

Prompt: Remove all files on /OLDDISK ? (Y/N)

Response: Y

Typing a Y response to this prompt causes the Filer to respond with the message:

```
/OLDDISK   zeroed
```

A subdirectory also can be Zeroed. This has the effect of erasing all files in a subdirectory, including any subdirectory files. As stated previously, a subdirectory can only be removed if the subdirectory contains no files. Thus an easy way to eliminate a subdirectory is first to use the Zero command to eliminate the files in the subdirectory and then to use the Remove command to eliminate the subdirectory itself.

EXAMPLE:

Imagine that you want to remove a subdirectory called JELLY that is stored on the Prefix diskette PORRAGE. A listing of

the files on diskette PORRAGE along with their file types is shown below:

```
/PORRAGE                  directory
  PRUNES.TEXT             textfile
  JELLY                   directory
    PEACHJELLY.CODE       codefile
    PLUMJELLY.CODE        codefile
    PLUM                  directory
      PLUMJAM.TEXT        textfile
      PLUMJELLY.CODE      codefile
  JUICE                   directory
    APPLEPLUM             datafile
    PLUMAPPLE             datafile
```

Prompt: Zero what volume ?

Response: JELLY

Prompt: Remove all files on /PORRAGE/JELLY ? (Y/N)

Response: Y

The Filer will then respond with the message

```
/PORRAGE/JELLY zeroed
```

The files contained on diskette PORRAGE now include the following:

```
/PORRAGE                  directory
  PRUNES.TEXT             textfile
  JELLY                   directory
  JUICE                   directory
    APPLEPLUM             datafile
    PLUMAPPLE             datafile
```

JELLY now has no files in it. Assuming that you next want to eliminate subdirectory JELLY, you will now type R for Remove.

Prompt: Remove what file ?

Response: JELLY

Typing this response tells the system to remove the subdirectory JELLY from the diskette directory.

Prefix

The Prefix command changes the current default prefix directory or subdirectory name to the directory or subdirectory name specified.

Basically the prefix is a time-saver; it prevents your having to type in a complete pathname specification each time you refer to a particular file. This section will tell you how prefixes can be used with the Apple III Pascal Filer.

When the system is booted, the prefix is set to the name of the Pascal system diskette in the built-in drive. To find out what the Prefix is currently set to, type P. You will be prompted

```
New prefix :
```

If you type a RETURN or ESCAPE, the current prefix is retained.

To change the prefix, type P from the Filer command level. When you are prompted,you should respond with the directory name and any subdirectory names that you want the system to place before each local filename that is specified. Each filename listed must be preceded by a slash. It is not necessary to put a directory (i.e., volume) or subdirectory on line before setting the prefix to it.

Setting the prefix to the device name of a diskette drive causes that drive to become the prefix disk drive, no matter what diskette is in the drive.

When you specify a pathname for some operation (such as a Transfer or Remove), the prefix is attached to the pathname you have entered, unless it is a full pathname. When you enter a local filename, the Filer attaches the prefix to that local filename before acting on it.

The only time that the prefix is ignored is when you specify a pathname beginning with a device name or slash.

EXAMPLE:

An Extended directory listing of the diskette NOVEL reveals the following:

/NOVEL	Size	Modified	Time	File type	Eof	Phys
INTRO.TEXT	6	22-Nov-81	3:22	Textfile	512	7
CHAP1	2	2Ø-Jan-81	2:Ø3	Directory	512	3
PART1.TEXT	22	2Ø-Jan-81	6:42	Textfile	512	23
PART2.TEXT	16	6-Mar-81	4:3Ø	Textfile	512	17
PART3.TEXT	14	19-Mar-81	1:32	Textfile	512	15
CHAP2	2	2-May-81	9:32	Directory	512	3
PART1.TEXT	12	2-May-81	3:11	Textfile	512	13

You are planning to do a lot of editing on Chapter 1 and so decide to set the prefix to the subdirectory CHAP1 on the diskette NOVEL .

Prompt: New prefix ?

Response: /NOVEL/CHAP1

Suppose you now want to change the name of

/NOVEL/CHAP1/PART1.TEXT

to

/NOVEL/CHAP1/FLIMFLAM.TEXT

Since the prefix already has been set to /NOVEL/CHAP1 , you don't have to type the complete pathname. Instead, the command is pleasingly brief:

PROMPT: Change what file ?

RESPONSE: PART1.TEXT

PROMPT: To what file ?

RESPONSE: FLIMFLAM.TEXT

The system ignores the Prefix whenever it sees a root volume name in a file specification. Suppose that the Prefix is still set to /NOVEL/CHAP1 and that you want to remove file /NOVEL/CHAP2/PART1.TEXT . Instead of changing the Prefix you

can cause the system to ignore it simply by giving the complete pathname of the file you want to access.

PROMPT: Remove what file?

RESPONSE: /NOVEL/CHAP2/PART1.TEXT

If you did not specify the complete pathname, but instead only specified CHAP2/PART1.TEXT, the system would automatically add the Prefix and assume that you were looking for file /NOVEL/CHAP1/CHAP2/PART1.TEXT. Since this file does not exist, you would get the message

File not found

Quit

The Quit command, which in invoked by typing Q from the Filer, causes the system to exit from the Filer and return to the main Command level. Remember to have your Pascal system diskette in the built-in drive when you issue this command.

Date

The Date command allows you to set the correct date each day so that a record may be kept of work performed on the system. Each time a volume, directory, subdirectory, or local file is created or updated, the date of creation is included in the directory.

When you invoke the Date command by typing D, you are shown the prompt

```
Today is 27-Apr-83   Time Ø:ØØ
Change date or time? (Y/N)
```

If you respond with Y , the next prompt is added to the screen:

```
Date format: dd-mmm-yy (month, year optional)
New date :
```

Pressing RETURN leaves the date unchanged, otherwise entering the new day, day and month, or full date changes the system's stored date. The month and year are optional, and only have to be changed at need.

After entering the new date, the time prompt appears on the screen:

```
Time format: hh:mm:ss (minutes, seconds optional)
New time :
```

Pressing RETURN leaves the time unchanged, otherwise entering the new time changes the system's stored time.

On systems with an installed clock, this operation updates the clock. On systems having no clock installed, a disk-resident "time stamp" is updated.

In either case, the new time is used when updating directory entries of files being changed, thus pinpointing the most recent update time of the file.

On systems without a clock, updating the time stamp can help you distinguish between files produced in the morning from those produced later in the day. You have to update the time to make this work properly, however.

After entering the time, or pressing RETURN if you have no clock installed, the newly set date (and time, if available) is displayed. If the new date or time is incorrect, type D to invoke the Date command again and reenter them.

Alter

The Alter command allows you to change some file characteristics as displayed by the Extended directory command. The characteristics that can be changed are write-protect status, date of most-recent change, and the filetype.

If you enter a wildcard specifier of = in the file name, all files operated on by Alter will be listed.

You will find the Alter command easier to use if you list the affected directory just before invoking Alter.

When you invoke the Alter command from the Filer, you are shown the prompt

```
Alter dir info of what file?
```


Entering a pathname gives you the prompt

```
Change write-protect status? (Y/N)
```

A reply of Y brings yet another prompt, continuing on the same line, unless the disk referenced is in Apple II Pascal format:

```
Write-protect? (Y/N)
```

Your response is followed by the prompt

```
Change last-modification date? (Y/N)
```

If you reply Y, the prompt

```
New date?
```

is added to the prompt line. Here you may enter the date in the format: dd-mmm-yy (day, month, and year) followed by a return. If the change does not affect the entire date, enter the entire date anyway...it won't be happy until you do.

The final prompt appears next:

```
Change file type? (Y/N)
```

The file types that may be used are:

```
*Asciifile
*Badfile
 Basicdata
 Basicprog
*Codefile
*Datafile
*Directory
*Textfile
```

*(Only the first four character must be entered.)

While you can change the file type listed in the directory, you might not be able to use the file under its new file type.

Responding to any of the Alter prompts with an N or a RETURN results in that operation being skipped and the next promptline being displayed. Entering an ESCAPE aborts Alter and returns operation to the Filer command line.

Workfile Commands

The following commands are used with the workfile. More information about using workfiles is included in Chapter 1 (Introduction) and Chapter 4 (The Editor) of this manual.

Get

The Get command, which is invoked by typing G from the Filer, identifies the designated diskette file for later use as the workfile. The next time you attempt to Edit, Compile, or Run, the designated file will be used.

When using the Get command, although you are told that the specified file has been "loaded", the Get command does not actually transfer the specified file to any other file, it just notes that a workfile has been specified and saves the name of the new workfile.

If there is already a workfile present on the Pascal system diskette when you issue the Get command, you are prompted:

```
Throw away current workfile ?
```

Typing Y for yes will clear the workfile, removing all files SYSTEM.WRK from the Pascal system diskette (if they exist), while N for no returns you to the outer level of the Filer.

Typing the filename's suffix in the file specification is not necessary. Wildcards are not allowed, and the size specification option is ignored.

EXAMPLE:

Suppose the Prefix diskette contains the following files:

```
FILERDOC2.TEXT
ABSURD.CODE
HYTYPER.CODE
FLIM.TEXT
ALPHA
 SORT.TEXT
 SORT.CODE
FILER.DOC.TEXT
FLIM.CODE
```

Prompt: Get what file ?

Response: FLIM

The Filer responds with the message

```
Text & code file loaded
```

since both text and code file exist. Had you typed FLIM.TEXT or FLIM.CODE , the result would have been the same: both text and code versions would have been identified for later use as the workfile. If only one of the versions exists, as in the case of ABSURD.CODE, then that version is identified for later workfile use, regardless of whether text or code was requested. Typing ABSURD.TEXT in response to the prompt would generate the message: "Code file loaded".

The text and code files themselves are not actually loaded at this time; the system loads their pathnames into memory for use when the files themselves have to be loaded.

Save

This command saves both components of the Pascal system diskette's workfile (both .TEXT and .CODE, if both exist) under the filename originally specified with Get or under a different filename if you so specify. This command is invoked by typing S from the Filer command prompt line.

If a file already exists with the specified filename, the workfile is saved under the specified name after removing the old file.

If you are saving the workfile as a file on the root directory of the Pascal system diskette, the workfile (which is already on that diskette) is simply renamed. When you save the workfile on a diskette other than the Pascal system diskette or on a subdirectory of the Pascal system diskette, the system is actually performing a Transfer of the workfile. Thus the workfile is unchanged after the Save is completed.

If saved to the root directory of the same disk, files beginning with SYSTEM.WRK disappear when the Filer's Save command is used to save the contents of the workfile under an original filename or under a new filename. If the system is rebooted before the Filer's Save command is used, the original name of the workfile's contents (as specified by the Get command) will be forgotten, but the file itself will not be affected.

You do not need to include a .TEXT or .CODE suffix when specifying the filename to be used when you save your workfile; the system will add the appropriate suffix for you. The system ignores the .TEXT or .CODE suffix if you do type it.

The Save command cannot be used to create a subdirectory. If, therefore, you want to save a file into a subdirectory that has not yet been created, you first must use the Make command to create the subdirectory.

If the diskette volume name or number is not given, the Prefix diskette is assumed. Wildcards are not allowed, and the size specification option is ignored.

The Apple III Pascal system has two Save commands: one at the Filer level and one at the Editor level. These two commands serve totally different functions.

EXAMPLE:

Suppose that you used the Get command to access the file OLDFILE on diskette MYDISK. After editing and recompiling this file, you decide to save it under the local filename NEWFILE . After typing S you will be prompted

Prompt: Save as /MYDISK/OLDFILE ?

Response: N

Prompt: Save as what file ?

Response: /MYDISK/NEWFILE

This causes the Filer to remove, after asking you for verification, any old file named /MYDISK/NEWFILE and then to save the workfile under that name.

EXAMPLE:

Prompt: Save as what file ?

Response: /RED/EYE

/RED/EYE constitutes a file specification, and this response will tell the Filer to attempt to transfer the workfile to the specified volume and file (see the Transfer command). If you

accidentely specified diskette RED , press the ESCAPE key. The command will be terminated.

If one of your disk drives contains a diskette named /RED , you will soon see the message

```
/PASCAL1/SYSTEM.WRK.TEXT ---> /RED/EYE.TEXT
```

This message tells you that the workfile named SYSTEM.WRK.TEXT , on the Pascal system diskette named /PASCAL1 , has been successfully transferred to the file named EYE.TEXT , on the diskette named /RED . If there is no diskette named /RED in any disk drive, you will see the message

```
Put in /RED
Type <space> to continue
```

This gives you the chance to insert a diskette named /RED , if you have one, into a disk drive.

EXAMPLE:

Suppose you earlier used the Get command to designate the file /MYDISK/LETTER as the next workfile. You then quit the Filer and entered the Editor, causing /MYDISK/LETTER.TEXT to be read into the computer. Finally, you added some new material to the file, and then used the Editor's Quit and Save commands to store the new version of the file under its old name and on the original diskette.

Now, back in the Filer again, you type S for Save and receive this prompt:

```
Save as /MYDISK/LETTER ?
```

If you type a Y , the Filer first asks

```
Remove old /MYDISK/LETTER.TEXT ?
```

Typing another Y causes your previous version of LETTER.TEXT to be removed from MYDISK , and then causes the new version to be saved on MYDISK. The following message will then appear:

```
/PASCAL1/SYSTEM.WRK.TEXT ---> /MYDISK/LETTER.TEXT
```

EXAMPLE:

This example will show you how to use the Make command to create a subdirectory and then save a file into that subdirectory.

Suppose you earlier used the Get command to designate the file /TIC/TOE as the next workfile. You then exited from the Filer and entered the Editor, causing /TIC/TOE.TEXT to be read into the computer. After you add some material you use the Editor's Quit and Update commands to store the new version on the Pascal system diskette as SYSTEM.WRK.TEXT. Back in the Filer, you decide that, because you are going to create several files which are similar in content to TOE.TEXT, it would be helpful to create a subdirectory, TAC, to store those files in. You cannot, however, save your file as /TIC/TAC/TOE because the subdirectory TAC has not yet been created. Instead you must use the Make command to create the subdirectory TAC . Then you can save your file into that subdirectory.

Prompt: Make what file?

Response: /TIC/TAC!

This response creates a one-block subdirectory TAC on the diskette TIC. Now you are ready to save your file. Typing S from the Filer you will be prompted:

Prompt: Save as /TIC/TOE ?

Response: N

Prompt: Save as what file ?

Response: /TIC/TAC/TOE

New

The New command, invoked by typing N from the Filer command level, clears the workfile, so that there is no default file to be used automatically by Edit, Compile, Assemble, and Run. The last file specified as the workfile by the Filer's Get command is no longer so designated. All versions of the workfile saved on the Pascal system diskette are removed from the directory. There will be no workfile on the Pascal system diskette until a workfile is created with the Editor's Update command.

If there is already a workfile SYSTEM.WRK present on the Pascal system diskette when you issue the New command, you are prompted:

Throw away current workfile ?

Response: Y will clear the workfile, removing all files SYSTEM.WRK from the Pascal system diskette, while N returns you to the outer level of the Filer.

Typing Y destroys the temporary workfile (SYSTEM.WRK) and it cannot be retrieved. This command reads from the typeahead buffer, so while it can make work go faster, sloppy typing can result in inadvertantly-destroyed files.

Use the New command to clear away the automatically-loaded workfile before you try to create a new file in the Editor or Compile any file other than the workfile.

What

This command identifies the name and state (saved or not) of the workfile. It is invoked by typing W from the Filer.

If the workfile has been saved onto any disk other than the Pascal system disk, the What command continues to report the workfile as not saved. This is because the workfile still exists on the main system disk. The message

Workfile is not named (not saved)

only indicates that there is no specified workfile on the Pascal system disk.

General Disk Upkeep Commands

Bad Blocks

This command identifies damaged blocks on a disk. To invoke this command, type B from the Filer.

The Bad-blocks command requires that you type a block device or volume name. The specified disk volume must be on-line

(currently available to the system). If the disk drive or disk is not there, the message

```
Volume not found
```

will appear.

EXAMPLE:

```
Prompt:    Bad block scan of what vol ?

Response:  /PASCAL3

Prompt:    Scan 28Ø blocks ? (Y/N)
```

In response you will normally type Y for "Yes", telling the Filer you want to scan the entire disk. If you wish to check only a smaller portion of the disk (a very unusual case), type N and you will be asked to type the number of blocks you want the Filer to scan. Since the number of blocks to scan depends on the type of storage device in use, check the number of blocks available on your disk.

Once the system knows how many blocks to check, it goes ahead and checks each block on the indicated disk volume for errors, and lists the block number of each bad block. In most instances you will see the message:

```
Ø bad block(s)
```

after the bad blocks scan has been completed. If the disk has bad blocks the disk drive will buzz and clatter, and you will see a message similar to this:

```
Scan for 28Ø blocks ? (Y/N) Y
Block 23 is bad
Block 24 is bad
Block 25 is bad
```

After the system has encountered three bad blocks it asks you

```
Continue scan ? (Y/N)
```


If you want to continue the scan, type Y. If you type N, a message like this is displayed:

```
File(s) endangered:
/THISFILE.TEXT
/THATFILE.CODE
```

The last two lines tell you that the three bad blocks are contained partly in the file THISFILE.TEXT and partly in the file THATFILE.CODE .

The system always asks you if you want to continue the scan after it has found three bad blocks. Then, it asks you if you want to continue after every 9th bad block (e.g. after the 9th bad block, the 18th, the 27th, etc).

If you see that one of your disks contains bad blocks, your safest move is to use the Transfer command to copy each of the files that do not contain any bad blocks to a healthy disk, then reformat the disk by using the SOS Utility disk.

The most common cause of reported bad blocks on a disk is caused by bad data written to the disk. Over-writing the information on the block usually cures the problem. If the bad-block error is caused by actual, physical damage or other problems with the disk's recording surface, that particular data is usually unrecoverable.

Dirt and fingerprints are also common culprits. An attempt to store information in such a bad block may result in the loss of that information, and may render the entire file unreadable. To guard against this kind of problem, you should always do the following:

1) Handle your diskettes very carefully, and keep them clean;

2) Do a Bad-blocks scan of every disk whenever you use the Zero command to erase its root directory to re-use it, and at any other times when you have suspicions about the disk.

A less common cause of bad blocks is opening the disk drive door or otherwise disturbing the recording process while the system is trying to store information on the disk in that drive. This will sometimes create an error in the data field of a disk sector.

Occasionally, the address field of a disk sector may be rendered unreadable by something you or the system does. This

problem is reported as a bad block by the Bad-blocks command. This problem must be corrected by reformatting the disk. Reformatting the diskette will erase everything on the disk, so be sure to save the undamaged files, first.

If a directory or subdirectory contains a bad block, the Filer cannot report on the validity of the files contained in it.

Examine

This command cannot be used with SOS-formatted diskettes. See Appendix B for details about how to use this command with Apple II Pascal diskettes.

Filer Command Summary

File Specification

.D2 or /MYDISK Typical volume specification. See Table 3-1, Device Volume Names and Numbers.

.D2/MYFILE.TEXT or MYDISK:MYFILE.TEXT Typical file specification. Unless otherwise noted, Filer commands require complete file specifications, including the suffix.

/MYDISK/MYDIR/MYFILE.TEXT Typical filename that includes subdirectory.

*	Specifies the built-in disk volume name.
:	Specifies the prefix volume name.
!	Creates a subdirectory

Volume name with no filename	Specifies the entire named disk.
Filename with no volume name	Specifies the named file on the prefix directory or subdirectory
=	Wildcard used in specifying a subset of filenames to be acted on. For example, BR=XT specifies all filenames beginning with BR and ending with XT .
?	Specifies use of last-entered filename. Example: BR?XT
$	Specifies a repeat use of the most-previous local filename used.
,	Separates any number of Filer command response fields. Some commands use response fields in pairs.

Information Commands

Volumes: Shows the devices and diskettes currently in the system, by volume number and by volume name.

List-directory: Shows what files are on the specified disk. If desired, list is sent to a second specified file or device.

Extended-directory-list: Shows what files are on specified disk, including the subdirectory structure. The list can be sent to a second specified file or device.

General File-Moving Command

Transfer: Transfers information from the first specified volume or file to the second specified volume or file. Used to move or save disk files, copy entire diskettes, or send files to a printer or other device.

General Diskfile Commands

Make: Creates a disk directory entry with the specified filename and [size]. Primarily used to create subdirectories.

Change: Renames the specified disk or disk file to the second specified name. If second specification is a filename, it need not include the volume.

Remove: Removes the specified file from a disk's directory or subdirectory.

Krunch: Packs all files together on a disk (Can be used only on Apple II Pascal diskettes; see Appendix B)

Zero: Removes the directory and subdirectories of the specified disk.

Prefix: Changes the current default directory or subdirectory name to the directory or subdirectory name specified. Response of : shows current Prefix name.

Date: Tells the current date and time last set for the system and allows them to be updated.

Quit: Leaves the Filer and returns to the outermost Command level.

Alter: Allows you change some file characteristics displayed by the List and Extended-list commands.

Workfile Commands

Get: Designates a specified disk file as the next workfile (no suffix needed: .TEXT and .CODE are supplied automatically). The next Edit, Compile or Run will use this file.

Save: Saves all components of the workfile SYSTEM.WRK under the specified filename (do not specify a suffix: .TEXT and .CODE are supplied automatically).

New: Clears the workfile, removing all SYSTEM.WRK files from the main system disk.

What: Tells the name and state (saved or not) of the workfile.

Disk Upkeep Commands

Bad-blocks: Tests all blocks on the specified disk to see that information has been recorded consistently. Any bad blocks found are reported.

Examine: Attempts to fix disk blocks reported as bad by the Bad-blocks command. Allows you to mark blocks that can't be fixed. (Used only on Apple II Pascal diskettes; see Appendix B)

4

The Editor

4
The Editor

Introduction

The Apple III Pascal Editor aids you in preparing programs and text, by providing the following functions:

Inserting and deleting text.

Moving part of a document from one position in the text file to another.

Merging all or part of one file into another text file.

Moving the cursor to a specified point in the file.

Finding or replacing strings of text throughout the text file.

Adjusting the margins and indentation of paragraphs of text thoughout the file.

The first part of this chapter describes some of the general characteristics of the Editor such as which disk files are needed to use the Editor, how to read the Editor's prompt line, and how the cursor moves. Next is a brief scenario that introduces the Editor, followed by a more detailed description of the Editor's modes of operation. The next section provides a brief explanation of how to use the Editor to edit BASIC text files. The remainder of the chapter consists of a detailed description of each of the Editor's commands.

Diskfiles Needed

To use the Editor, the disk file SYSTEM.EDITOR must be available to the system. SYSTEM.EDITOR is normally provided on diskette /PASCAL2 .

Files that are being edited may be on any diskette in any drive.

A "Window" Into the File

The Apple III Editor is designed for use with the Apple's video display. After a file has been read into the Editor, the Editor displays the beginning of the file on the second line of the screen. If the file is too long for the screen, only the first portion of the file is displayed. The displayed portion is a "window" into your file.

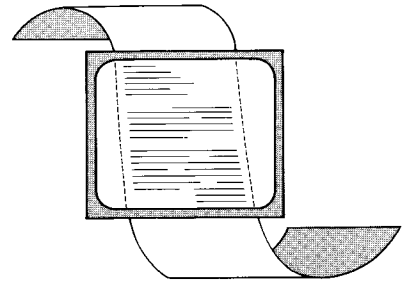


Figure 4-1. Editor "Scroll Window".

Although the whole file is accessible to you through the Editor, only part of the file can be seen through the "window" of the screen. When any Editor command takes you to a position in a file that is not displayed, the "window" is moved to show that portion of the file.

The Cursor

The cursor marks the position in the file where actions performed by the Editor have their effect. The window shows a part of the file near the cursor and moves whenever you move the cursor out of the window through the file.

While the cursor appears to move over characters on the screen, it is actually between the character that appears to its right and the character that the cursor seems to be on. (Don't forget that a space is a character, too.)

Some of the Editor's commands permit additions, changes, or deletions of such length that the screen cannot display all of the text that has been changed. In those cases, the screen shows the portion of the file where the cursor was positioned after the change.

The Prompt Line

The Editor's prompt line lets you know that you are in the Editor rather than in some other part of the system. The language system is complex and these signposts indicate where you are in the system. Secondly, the prompt line reminds you of the commands you can use.

Here is the complete Editor prompt line:

```
>Edit:  A(djst C(py D(lete F(nd I(nsrt J(mp M(rgin Q(uit R(plce S(et X(chng Z(ap
```

Notation

The notation used in this chapter is borrowed from the notation used in the Editor prompt lines. In the Editor prompt lines, a word enclosed between angle-brackets <like this> indicates that you can press a particular key. <ret> means that you can press the RETURN, <esc> means that you can press the ESC key, and <ctrlC> means that you can press CTRL-C. You can use either lowercase or uppercase characters when typing Editor commands.

A Brief Scenario

This scenario demonstrates the use of the Editor, with only a brief explanation of the terms and concepts used. The section called A LITTLE MORE DETAIL is a more complete description of the same concepts. Following that section is a full discussion of all Editor commands.

If a workfile is present, you can clear it by using the Filer (see Chapter 3).

Starting a New File

Now that the workfile has been cleared, press E for Edit from the Command level. Soon, this prompt appears:

```
>Edit:
No workfile is present. File? (<ret> for no file, <esc> to exit editor)
:
```

To start a new file, instead of reading an existing file from disk, press the RETURN key and this prompt appears at the top of the screen:

```
>Edit:  A(djst C(py D(lete F(nd I(nsrt J(mp M(rgin Q(uit R(plce S(et X(chng Z(ap
```

You can now press I for Insert. The following prompt line then appears at the top of the screen:

```
>Insert:  <Text>, <bs> a char, <del> a line, <ctrlC> accepts, <esc> escapes
```

The prompt tells you what you can do while typing in text in insert mode. If you type in a wrong character, backspacing over the character with the left-arrow key will remove it. After removing the offending character, continue typing again wih your text. If an entire line offends you, typing CONTROL-X will remove that line. The cursor will then appear at the end of the preceding line.

If you decide it was all hopeless anyway, press the ESCAPE key to throw out all the text that you have typed since last entering insert mode. Otherwise, after you have entered all the text you want, type CONTROL-C to save the text and leave insert mode.

When you wish to add more text, press I again and the Insert prompt line reappears. Further typing inserts text at the cursor position, until you terminate the latest insertion by pressing CONTROL-C.

For example, you might press I for Insert, and then type

```
PROGRAM EXAMP;
BEGIN
  WRITE('AN APPLE A DAY')
END.
```

ending each line by typing RETURN. Accept this insertion by pressing CTRL-C.

Saving Your Work

Finally, when the text is the way you want it, press Q for Quit and then press W for Write. The system prompts you for a pathanme for your new file to be stored in on disk. For example, you could name your new file

/MYFIRST/PROGRAM

and your file would be saved on the disk named /MYFIRST in a file name PROGRAM.TEXT .

If you wish to leave the Editor, press E to exit and return to the Command line. If your file is a Pascal program, you can now press R for Run, and the system will ask you for a pathname. Enter /MYFIRST/PROGRAM and press RETURN. The system will then ask for the pathname of the codefile to be created. Enter a $, followed by RETURN, and the system will then attempt to compile and run the file, storing the compiled version of your program (if compilation is successful) as PROGRAM.CODE .

To successfully compile your program, the Compiler must be in one of the system's drives.

If you wish to change your file for any reason, simply press E to Edit again. Now the system will ask you for a pathname: Type

/MYFIRST/PROGRAM

and the Editor will read in the file PROGRAM.TEXT from your disk named /MYFIRST , ready for more editing.

A Little More Detail

The first part of this chapter gave a superficial description of the steps required to edit a text file. This section continues the chapter with a detailed description of the Editor's operation and commands.

Entering the Editor

When the Command prompt line is on the screen and your Pascal system diskette is in the built-in drive (device .D1), press E for Edit. If the system already has a text workfile (see the

section on workfiles in Chapter 1), that file is automatically read into the Editor, ready for work. If the system does not have a workfile yet or if only a code workfile exists, this prompt line appears when you first enter the Editor:

```
>Edit:
No workfile is present.  File?  (<ret> for no file, <esc> to exit editor)
:
```

There are three ways to answer this opening prompt line's question:

1. You can answer by entering the pathname of any text file that already exists on diskette.

   For example, you might enter

   /HOHUMM/PROGRAM1

   When you press the RETURN key, the file named PROGRAM1.TEXT is retrieved from diskette /HOHUMM , and the text of that file appears on the screen. Note that you do not need to enter the .TEXT at the end of any of your text files. However, be sure to enter a period (.) at the end of any files whose pathnames don't end in .TEXT .

2. You can answer by pressing the RETURN key.

   This tells the system that you are starting a new file. The only thing visible on the screen after doing this is the normal Edit prompt line:

   ```
   >Edit: A(djst C(py D(lete F(ind I(nsrt J(mp M(rgin Q(uit R(plce S(et X(chng Z(ap
   ```

   A new file has been started and currently has nothing in it. Type I for Insert to begin inserting your program. No permanent version of this new file exists until you use the Quit command to exit from the Editor and the Write command or the Update command.

3. You can answer by pressing the ESC key:

   This causes the Editor to return you to the system Command level, a useful option when you didn't mean to press E to Edit.

Moving the Cursor

To edit, you must move the cursor. There are four "arrow-keys" on the keyboard that move the cursor up and down, left and right. You can move the cursor only when one of these prompt lines is at the top of the screen: Edit, Delete, or Adjust.

Vertical motion of the cursor is made without regard to the text on the page, otherwise the cursor will remain in the text of the program. For example, suppose the cursor appears after the "N" in "BEGIN" :

```
PROGRAM EXAMP;
BEGIN
  WRITE('AN APPLE A DAY')
END.
```

(Actually, the cursor is "between" the invisible RETURN character that ends every line and the "N" in BEGIN.) If you press the right-arrow key, the cursor moves to the "W" in "WRITE" :

```
PROGRAM EXAMP;
BEGIN
  WRITE('AN APPLE A DAY')
END.
```

Similarly, pressing the left-arrow key now moves the cursor back after the "N" in "BEGIN".

If it is necessary to change the third line, "WRITE('AN APPLE A DAY')", to "WRITE('AN ORANGE A DAY')", you must first move the cursor to the correct spot.

For example, if the cursor is on the "P" in "PROGRAM EXAMP;", move down two lines by pressing the down-arrow twice. After you press the down-arrow once, the cursor is on the "B" in "BEGIN"; after you press the down-arrow key a second time, the cursor is in front of the "W" in "WRITE".

```
PROGRAM EXAMP;
BEGIN
  WRITE('AN APPLE A DAY')
END.
```

Now, using the right-arrow key, move the cursor until it appears on the "A" in "APPLE".

Note that the cursor may at times appear to be outside the text when moved with downward and upward cursor moves. In the last illustration, the cursor appears to be in the blank space before the "W" in "WRITE". As far as the Editor knows, however, the cursor is actually immediately before the "W" in "WRITE". So do not be surprised when you first press the right-arrow key and the cursor jumps to the "R" in "WRITE". When the cursor appears to be outside the text, from the Editor's viewpoint it is actually adjacent to the character nearest the cursor.

Inserting Text

The Edit prompt line shows the command option I(nsrt. To insert text, first move the cursor to the correct position, and then press I. Always move the cursor to the correct position BEFORE pressing I. Earlier, you moved the cursor to the A in APPLE. Now, when you press I , an insertion will be made just to the left of the A. The rest of the line, starting with the A, will be moved to the righthand side of the screen.

If the insertion is lengthy, the righthand portion of the line (beginning with "A") will be moved down to allow room on the screen for more inserted text to appear. After you have pressed I, the following prompt line should be displayed on the screen:

```
>Insert:  <Text>, <bs> a char, <del> a line, <ctrlC> accepts, <esc> escapes
```

If that prompt line did not appear at the top of the screen, you could not insert characters. You might have pressed a wrong character. If so, press ESCAPE to bring up the Edit line prompt, then type I for Insert.

If the cursor was at the A in APPLE when you pressed I , the Insert prompt line appeared and the remaining portion of that line (beginning with "A") was pushed to the righthand edge of the screen. You can insert ORANGE by entering those six letters. They will appear on the screen as they are pressed.

There remains one more step: Accepting or rejecting the final result of your editing. The choice at the end of the prompt line indicates that pressing CONTROL-C accepts the insertion while pressing ESCAPE rejects the insertion, leaving the text as it was before you began the insertion.

(Portion of screen before typing I)

```
PROGRAM EXAMP;
BEGIN
  WRITE('AN APPLE A DAY')
END.
```

(Portion of screen after typing I for Insert)

```
PROGRAM EXAMP;
BEGIN
  WRITE('AN                                  APPLE A DAY')
END.
```

(Portion of screen after typing "ORANGE")

```
PROGRAM EXAMP;
BEGIN
  WRITE('AN ORANGE                           APPLE A DAY')
END.
```

(Portion of screen after insertion followed by CONTROL-C)

```
PROGRAM EXAMP;
BEGIN
  WRITE('AN ORANGEAPPLE A DAY')
END.
```

(Portion of screen after insertion followed by ESCAPE)

```
PROGRAM EXAMP;
BEGIN
  WRITE('AN APPLE A DAY')
END.
```

It is legal to insert a carriage return. This is done by pressing the RETURN key while the Insert prompt line is at the top of the screen.

Deleting Text

Making deletions is similar to making insertions. Now that you have inserted the word ORANGE into the EXAMP program and have pressed CTRL-C, you must now delete APPLE. Move the cursor so that it is placed directly on the first character that you wish to delete. Then press D for Delete. The following prompt line appears:

```
>Delete:  <Moving keys>, <ctrlC> accepts, <esc> escapes
```

Each time you press the right-arrow key, the character on which the cursor is positioned disappears. Pressing the left-arrow key erases the next character to the left of the cursor. In this example, pressing the right-arrow key five times causes the word APPLE to disappear. To terminate the deletion, you have the same choice you had with Insert. Press CONTROL-C to make the proposed deletion permanent. Press the ESCAPE key to cancel the proposed deletion and restore the original text.

It is legal to delete a carriage return. When the cursor is at the end of the line, press D for Delete. Then press the right-arrow key until the cursor moves to the beginning of the next line. After completing the deletion by pressing CONTROL-C, you may find that the line extends beyond the end of screen with a "!" at the rightmost position in the line. The text is not lost, but is not displayed. The ! indicates that there are characters remaining in the line beyond the 80th position.

Leaving the Editor

It is a good idea to exit temporarily from the Editor and then use either the Update or Save commands about every 15 minutes or so. This way, in case of accident (such as the power going out or your mistakenly deleting an important part of your file), you won't lose more than 15 minutes worth of text entry.

To leave the Editor, press Q for Quit. The following prompt appears:

```
>Quit:
        To leave Editor, type
                E(xit to main command line

        To store text file on disk, type
                W(rite to a new file name
                U(pdate /NEWPASCAL2/SYSTEM.WRK.TEXT
                S(ave /MYFIRST/PROGRAM.TEXT

        To continue editing, type
                R(eturn to same file
                C(hange to another file
```

If you choose the E(xit option, any changes you have made to the file since the last time it was saved to a disk file will be thrown away forever. The Editor courteously asks if you wish to

```
Are you sure you want to throw away changes since last update?
```

if any such changes exist. You have the option of either saving changes made since the last time you saved the file (type N for no), or of not saving them and exiting to the Command level (type Y for yes).

One way to save a copy of your present file is to press U for Update. This saves your file on the Pascal system diskette under the filename SYSTEM.WRK.TEXT.

If you use Update, you should also use the Filer's Save option to save the default system workfile under its own filename before using the Editor to modify or create another file.

Remember that the Filer's New command will erase the workfile SYSTEM.WRK.TEXT any time it is used, and that the Editor's Update command always stores the just-edited file under the same filename SYSTEM.WRK.TEXT. You will not want SYSTEM.WRK.TEXT to be your only copy of a file once you are through working on it.

Another method of saving your present file onto diskette is to use the Editor's Write command. Assume that you have created a file in the Editor that you want to save. After pressing Q to Quit the Editor, you press W for Write. When the filename prompt appears, you enter the file's pathname, which, of course, includes the name of the diskette on which the file will be

stored (for example, as /MYDISK/PROGRAM2). If you currently have no file by that name on the designated diskette, the file is stored as whatever filename you enter. However, if you have a file by the designated same name on that diskette, you see the prompt:

```
/MYDISK/PROGRAM2.TEXT already exists.  Delete before Write?-->
      Y(es to delete old file before starting new one.
      N(o to delete old file after new one is complete.
      <esc> to cancel W(rite.
```

Entering Y deletes the old version of the file stored on disk before writing the new version to the disk. The only time that this is necessary is when there is too little free space on the diskette to fit both the old and new versions of the file.

Entering N deletes the old file only after the newest version has been stored on the disk. This is a little safer than using the Y(es option, but will only work when there is sufficient room on the disk for both files. You should use the N(o option whenever possible, especially on dark and stormy nights: a black-out while your file is being saved when you have specified the Y option may leave you with no file at all, not just part of the new one.

If you change your mind, and don't want to save the file now, you can press ESCAPE to return to the Quit menu.

The Q(uit prompt line reappears after the Write finishes. You can then enter E to Exit entirely from the Editor to the Command level, R to Return to the Editor to continue editing the same file, or C to Change to another file. This last choice allows you to begin working on another file.

Note that when the Quit prompt line appears after you have specified the name of the current file, the Save option appears in the second set of options headed

```
    To store file on disk, type
```

You now can press S to Save your current file under the same name during each editing session. This saves you the trouble of having to enter the entire pathname every time you want to store more information in the same file.

The Save command operates like the Write command, except that you don't need to specify the pathname each time you write to disk.

Editing an ASCII File

The Editor normally deals with Pascal textfiles having the special format described in Appendix B. However, it can also be used to edit a file without this format, called an asciifile. This ability is useful when editing a file transported from a non-Pascal system such as Apple III Business Basic, or Visicalc III.

To edit an asciifile, just give the editor the correct pathname when entering the Editor. The name of most asciifiles do not end in .TEXT so you will have to add a period at the end of the pathname. Now edit the file as you would a regular Pascal textfile.

When you are finished editing the file, the normal Quit process will create an asciifile as output; here, however, the Editor knows it will be creating an asciifile and so will not automatically append a .TEXT to the file's pathname with the Quit Write command.

The Quit Update command should not be used as the workfile should not be an asciifile. Also, the Assembler and Compiler cannot handle an asciifile correctly.

All Visicalc III files, including templates and sheets, are asciifiles and may be read and manipulated by the Editor. For more information, see the Apple III Visicalc Users Guide.

To convert an asciifile to a textfile, read it in to the Editor; type

Set Environment Asciifile False (SEAF) CONTROL-C

to change the type of file, then Write it out in the normal fashion to a new text file. Similarly, to convert a textfile to an asciifile, read it in, set Asciifile to True and Write it out.

Editing BASIC Files

A Business BASIC text file corresponds to a Pascal asciifile. Thus any BASIC text files your program creates can be edited

with the Pascal Editor. In particular, you can create an BASIC text file version of a BASIC program using the LIST command with output directed to a file, use the Editor to edit that file, then EXEC it back into BASIC and save it as a BASIC program file. An example follows using the READCRT program included on your Apple III Business BASIC Disk.

In BASIC type:

```
)LOAD .d1/readcrt
)OPEN #1, .d2/readcrt: OUTPUT #1: LIST: CLOSE #1
)
```

Now boot the Pascal system, go into the Editor and in response to the File? prompt type:

```
>Edit:
No workfile is present. File? (<ret> for no file, <esc> to exit editor)
: .d2/readcrt.
```

Do any editing you wish to do, then type Quit Save No to write the file back. Finally, boot BASIC again and type

```
)EXEC .d2/readcrt
```

After you have done this, one) character will appear on the screen for each line of your file.

The Editor Commands

The following sections describe, in detail, each of the commands used by the Editor.

The relationship of all the Editor commands to each other and the Editor command level is show in Figure 4-2 below.

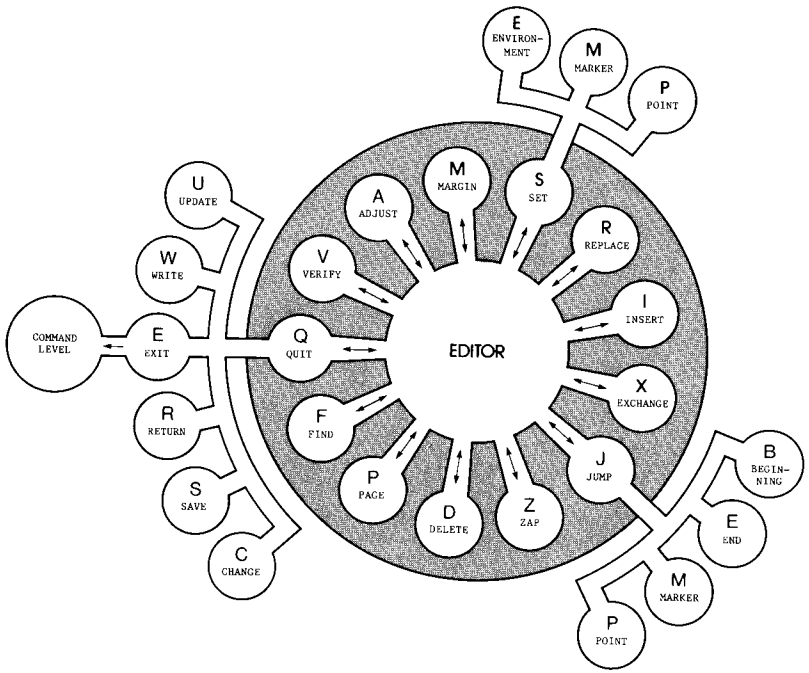


Figure 4-2. The Editor.

The Cursor

You spend almost all of your editing time either following or directing the movement of the cursor. The cursor is the white rectangle displayed on the screen to indicate your position in the file. How to make the cursor do what you want it to do is described here.

Cursor Movement

In general, special cursor moves (see below) are used in the Editor to move the cursor through the text and place it just where you want the next command to have its effect.

Notice that not all commands affect the character on which the cursor actually appears to be placed.

As you can see below in Figure 4-3, the actual position of the cursor is between the character is appears to be on and the next character to its left. If the cursor movement (see Direction Indicators) is toward the end of the file, which is toward the lower-right corner of the screen, most Editor commands can act on characters following the character the cursor appears to be on as well as the character it appears on. If the cursor direction is set toward the beginning of the file, commands will act on characters to the left and above the character the cursor appears on.

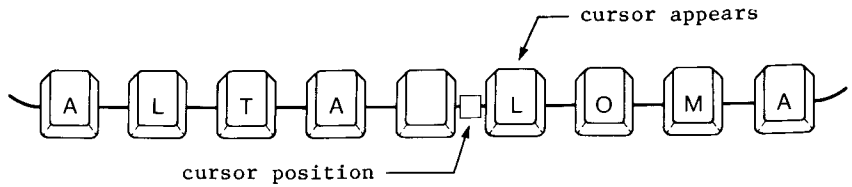


Figure 4-3. Cursor Positioning and Action.

Repeat-factors

Some of the command options (and the cursor moves) allow you to use repeat-factors. A repeat-factor is a number that is entered immediately before issuing a cursor move or command that causes the cursor move or the command option to repeat for the number of times indicated by the repeat-factor.

For example, pressing 2 and then pressing the down-arrow key causes the cursor move to be executed twice, moving the cursor down two lines. Cursor moves and commands allowing a

repeat-factor assume the repeat-factor to be 1 if no number is typed.

Explicit repeat-factors may range from Ø to 9999. Typing a slash (/) before a cursor move or a command indicates an "infinite" repeat-factor, and causes the move or command to be repeated as many times as possible in the file.

The Direction Indicator

The first character displayed on most Editor prompt lines is a direction indicator showing the set direction. A greater than (>) character indicates forward direction. A less than (<) character indicates backward (or reverse) direction. When the Adjust, Edit, or Delete prompt lines are showing, you can type the > or the < key (with or without the SHIFT key) to change the direction that the cursor moves.

Forward operations begin at the current cursor position and proceed toward the end of the file. When the set direction is set to the reverse direction, operations begin at the current cursor position and proceed toward the beginning of the file. Commands whose operation is affected by the set direction are noted as such in the detailed command description.

The set direction does not affect the operation of the cursor-moving up-, down-, right-, and left-arrow keys.

Cursor Moves

Repeat-factors are allowed with all cursor moves. The cursor moves are outlined below.

If you press	The cursor moves
up-arrow key	up by lines.
down-arrow key	down by lines.
right-arrow key	right by characters.
left-arrow key	left by characters.
space	in the set direction by characters.
TAB key	in the set direction, to the next tab-stop (tab-stops are set every eight spaces across the screen).
RETURN key	in the set direction, to the beginning of the next line.
P(age	in the set direction, one full screen.

If the cursor appears to the right of a line of text, the Editor acts as though the cursor is positioned immediately after the last character in the line.

If the cursor appears to the left of a line of text, the Editor acts as though the cursor were immediately before the first character in the line. When any action is taken, the cursor immediately moves over to the character at the nearest end of the line and begins performing the operation.

Be careful if you are using the fast-repeat feature with a cursor move: When you press firmly on the repeat key, the repeat function speeds up and cursor moves will then occur faster than they can be updated to the screen. This causes you to overshoot your intended destination. If this begins to happen, press CONTROL-6 on the numeric keypad to cancel the rest of a runaway cursor move.

Moving Commands

The commands described below, Jump and Find, allow you to go to a particular point in the file, defined either by markers placed in the file or by particular words or character strings found in the file.

Jump

The Jump command moves the cursor to the beginning of the file, to the end of the file, to a marker that you have placed in the file, or to a point.

To use the Jump command, press J for Jump while at the Edit level. The following prompt line then appears:

```
>Jump:  B(eginning, E(nd, M(arker, P(oint, <esc> escapes
```

Typing B for Beginning moves the cursor to the beginning of the file, displays the Edit prompt line and the first page of the file. Typing E for End (after you have again pressed J for Jump) moves the cursor to the end of the file, displays the Edit prompt line and the last page of the file. Typing M for Marker causes the Editor to display the prompt line:

```
Jump to what marker?
```

If you respond by typing the name of a marker (described under Set Marker in this chapter) that exists in the file, when you press the RETURN key, the cursor moves to the marker position in the text. Pressing the ESCAPE key instead of RETURN aborts the jump. If you press a marker name that does not exist in the file, you see this message:

ERROR: Marker not there. Please press <space> to continue.

and the cursor does not move. Placing markers in the text is explained under Set Marker in the section on Miscellaneous Commands.

Typing J for Jump and P for Point moves the cursor to the last point designated by a Find, Replace, Insert, or Set Point command in your file. The point and cursor are simultaneously exchanged so that typing J for Jump and P for Point again returns the cursor and Point to where they were positioned just before you first pressed Jump Point.

Find

The Find command searches through a file for a specified string of characters (target string), finds the specified number of occurrences of that string, and places the cursor at the end of the string.

To use the Find command, press F for Find while at the Edit level. One of the following prompt lines then appears, depending on the setting of the Environment's Token default option (see the Set Environment section under Miscellaneous Commands):

>Find[1]: L(it <target> =>

if the Environment's Token default option is set to True, or

>Find[1]: T(ok <target> =>

if the Environment's T(oken default option is set to False.

Direction

The Find command searches for the specified occurrence of the target string, starting at the present cursor position and scanning through the text in the set direction (indicated by the arrow at the beginning of the prompt line).

A target string can be found only if it appears in the text between the cursor and the end of the file toward which the search is progressing.

If you wish to search in the reverse direction, you must set the direction indicator before typing F to find something. See the section on the direction indicator (under Cursor Movement in this chapter) for information about setting the direction. If the required occurrence of the target string is not found by searching through the text in the set direction, this message appears:

ERROR: Pattern not in rest of the file Please press <space> to continue.

Notice that the search does not "wrap around". There is no search in the portion of the file between the cursor and the end of the file in the direction OPPOSITE the set direction.

Repeat-factor

You may specify a repeat-factor just before typing the F for Find. It appears on the prompt line in square brackets: [1] , for example. If you enter a repeat-factor of 6, the cursor appears after the 6th occurrence of the target string. If there are only four occurrances of the target string in the file,

ERROR: Pattern not in rest of file. Please press <space> to continue.

is displayed. If you don't enter a repeat-factor, a repeat-factor of one is assumed. A repeat factor of / finds, and stops after, the last occurrance of the target.

Target String and Delimiters

You can search for strings containing any characters including control characters. The target string must be set off by delimiters. Attempting to use a delimiter that occurs in the target string causes the Find to execute immediately. This means that you will search for only part of your intended target string.

The Editor allows you to choose any character that is not a letter or a number to be a delimiter so long as the character is not in the target string. The most common choice is the slash (/) because it is a character that is not

often searched for in the text, and is easy to type. Examples of delimiter usage are given below.

```
/UNLIKELY/
/An honest man/
```

or

```
!2/3!
```

Multi-line targets and targets containing control characters are allowed, except for the control characters NULL, BS, CONTROL-X, and ESCAPE.

If you forget to precede the target string with a correct delimiter character, you see the message:

```
ERROR: Invalid delimiter for Find. Please press <space> to continue.
```

Just try again, this time beginning with a correct delimiter.

Uncase Option

Normally, Find will only count target string matches as good of the potential target is all lowercase. Obviously, this will miss any occuring at the beginning of sentences or as part of a proper noun.

Uncase allows you to match any string having the proper sequence of characters, regardless of their case.

If you press a "U" immediately after bringing up the Find command line, all occurances of the target string will be found, regardless of case.

Literal or Token Search

The target string is treated differently, depending on whether you select Literal search or Token search. When you select Literal search, the Editor looks for the occurrences of a string of characters that exactly match the target string, even if it occurs within a word. When you select Token search, the Editor looks for occurrences of the target string using rules similar to the Compiler's to match a string.

The default setting of the search option is set in the Environment. The Find prompt line displays the alternative search mode; either L(it or T(ok. If you do not specify a

search option, the default search option (the one NOT mentioned on the prompt line) is used.

To use Token search when the default is Literal search (prompt line says T(ok), press T after the prompt line and before the target string. To use Literal search when the default is Token search (prompt line says L(it), press L before typing the target string. Note: nothing appears to happen when you press L or T ; the letter just appears where you are about to press the target string. See Set Environment in Miscellaneous Commands for more information about Literal and Token search options.

ESC Option

At any point during your response to the Find prompt, you can abandon this command and return to the Edit level by pressing the ESC key.

Same-string Option

Pressing S instead of typing a delimited target string tells the Editor to use the same string last specified for the target string (the target string may have been specified either in Find mode or in Replace mode). From the Editor, entering the command

```
FS
```

causes the previously specified target string to be used again and is displayed at the top of the screen when you type the S.

Note: The Environment displays the current <target> string that will be invoked by pressing S as a string response if you have forgotten what would be used (see Set Environment under Miscellaneous Commands).

EXAMPLE:

Suppose you are editing, with Token Search selected, a file containing the following text:

```
PROGRAM STRING1;
BEGIN
  WRITE('TOO WISE ');
  WRITE('YOU ARE');
  WRITELN(',');
  WRITE('TOO WISE ');
  WRITELN('YOU BE.')
END.
```

In the STRING1 program, with the cursor at the P in the line

```
PROGRAM STRING1;
```

enter F to select Find. When the Find prompt line appears, enter

```
/WRITE/
```

You MUST enter the two slashes (or two of some other delimiter). The prompt line should now appear as:

```
>Find[1]: L)it <target>  =>/WRITE/
```

until you press the last slash; the cursor then jumps immediately to the first character following the E in the first occurrence of the Token target string

```
WRITE
```

EXAMPLE:

Again in the STRING1 program, with the cursor at the E of END. , enter:

```
<2F
```

This prepares the system to find the second pattern (you entered a repeat-factor of 2) in the reverse direction (you changed the set direction by entering <). When the prompt line appears, enter

```
/WRITELN/
```


The prompt line should read:

```
<Find[2]: L)it <target>  =>/WRITELN/
```

When you press the last / , the cursor moves immediately to the W in the second nearest occurrence (searching backward through the file) of the Token string WRITELN . (The fifth line of the program).

After typing JB to Jump to the Beginning of the file, enter

```
F/WRITE/
```

This locates the first occurrance of the Token string WRITE , searching forward (to the third line). Now, typing

```
>FS
```

makes the prompt line flash:

```
>Find[1]: L(it <target>  =>/WRITE/
```

and the cursor appears at the next occurrance of WRITE.

Text-Changing Commands

The commands listed in this section (Insert, Delete, Zap, Copy, Exchange, and Replace) enable you to change the content of the text that you are working on.

Insert

The Insert command allows you to put new information into the text you are creating or editing.

To insert text, position the cursor where you want to insert and enter I for Insert while at the Edit level. The following prompt line appears:

```
>Insert:  <Text>, <bs> a char,<del> a line, <ctrlC> accepts, <esc> escapes
```

When the Insert prompt line appears at the top of the screen, the characters that you enter are inserted between the character on which you placed the cursor and the character that was immediately to the cursor's left.

The Insert prompt line also reminds you how to correct errors while you are entering text: The <bs> stands for the

left-arrow key (BACK SPACE), used to delete text a character at a time, and <del> stands for CONTROL-X , used to delete text back to the most recent RETURN character.

To save time, the Editor does not constantly re-write the entire screen as you insert each new character. Instead, it makes a gap in the text, just where your insertion will appear, and then waits for you to enter.

If you want to see exactly how the insertion will look in its final form, you can accept your insertion by pressing CONTROL-C .

If you make a mistake while inserting, just use the left-arrow key to backspace over your inserted characters. To delete the entire line that you are in the process of inserting, back to and including the previous RETURN character, enter a CONTROL-X . You can erase only the text that you have added during the current insertion.

The set direction does not affect the Insert command.

At any time during an insertion, you can cause the Editor to accept the insertion as it stands (making it a part of your file) by pressing CONTROL-C . Until you press that CONTROL-C , you can cause the Editor to discard everything you have entered since beginning the insertion by pressing the ESC key.

Whether the insertion is made and accepted (using CONTROL-C) or made and rejected (by pressing the ESCAPE key), that insertion is still available, in the Copy buffer, until the next insertion or deletion, for use by the Copy command. You can use this to duplicate your last insertion as many times as you wish. Remember that all of the text may not be available if there is not enough room in memory for all of it.

If you have mistakenly rejected the insertion by pressing ESCAPE, you can still recover the text using the Copy Buffer command.

The maximum size of a file that you can edit on a 128K Apple III is approximately 28,ØØØ characters, or about 54 diskette blocks. When your file can hold only about a thousand more characters, you will receive this warning as you begin entering more text:

ERROR: Please finish up the insertion. Please press <space> to continue.

When you respond by pressing the space bar, the Insert prompt line will still be at the top of the screen. You can continue your insertion, but you have been warned that your file is almost full. You should start a new file right away or split the present file into two parts. If you continue inserting, you will eventually receive this more urgent message, when your file has exceeded the amount of text it can hold:

ERROR: Buffer overflow!!!! Please press <space> to continue.

The Editor accepts your insertion, and any further attempts to insert text cause this message:

ERROR: No room to insert. Please press <space> to continue.

The Edit prompt line reappears on the top of the screen, and you are not able to add any more text to your file.

Text Formats

There are two basic ways that text can be formatted as you insert it; programming mode and document mode. The formatting mode is basically determined by the settings of two options in the Environment; Indent-auto and Filling (see Set Environment under Miscellaneous Commands).

You select programming mode by setting Indent-auto to True and Filling to False. You select document mode by setting Indent-auto to False and Filling to True. For example, this manual was written in document mode, and the programming examples found in the Apple III Pascal system manuals were written using programming mode.

Inserting in Programming Mode

This is the usual setting of the Environment when you are writing computer programs, building tables, writing outlines, or writing poetry. During an insertion, the margins set in the Environment are ignored. Instead, you must terminate each line yourself, and start a new line, by pressing the RETURN key. Each new line automatically starts at the same indentation as the first non-space character of the preceding text line.

When you first create a document in the Editor, the Environment for that document is set for programming mode. If the document Environment settings have been changed to document mode and you

wish to set it back to programming mode, you can set the programming mode by typing

SEFFIT

followed by RETURN. SE allows you to set the Environment options, FF selects Filling and sets it to false, and IT selects Indent-auto and sets it to true.

You can change indentation of a line by typing spaces or tabs (to indent farther) or by pressing the left-arrow key as the first character in a line (to indent less). Pressing CONTROL-Q as the first character of any new line sets the indentation of that line to zero, or type CONTROL-X to discard a line and return to the end of the previous line. You can also use the Adjust command to create a new indentation for a line after you leave Insert mode. (Adjust is described in the section entitled Formatting Commands.)

The Editor allows you to insert new lines of text into a file without disturbing the indentation of existing lines. Beginning an insertion at the start of a line and ending with a RETURN CONTROL-C will not change the indentation of that line.

If you try entering text beyond position 71, your Apple III beeps to warn you that you are approaching your right margin. If you enter text beyond position 8Ø, an exclamation mark (!) appears at the rightmost position on the screen. This character at the end of any line indicates that the line contains more than the 8Ø characters that can appear on the screen. Additional characters entered into that line are not lost, but they are not displayed.

To see the hidden characters, you can insert a RETURN character anywhere in the visible portion of the line; or use the Margin command. The Margin command will ask for confirmation of your intent, since Filling has been set to false.

EXAMPLE:

In Programming mode, type I for Insert and then type the following sequence of keys:

```
ONE<return>
<space><space><space>TWO<return>
THREE<return>
<left-arrow>FOUR<return>
```


This should create the indentations shown at the left below:

```
ONE       Original indentation
   TWO    Indentation changed by <space><space><space>
   THREE  <return> causes indentation to level of line above
  FOUR    <left-arrow> changes indentation from level of line above
```

Inserting in Document Mode

This is the normal setting of the Environment when you are writing text such as letters, user manuals, or other documents. The Editor forces all insertion to be between the margins set in the Environment. The instant a new word (as you are entering it) exceeds the right margin, a RETURN character is automatically inserted before the word, and the entire word (or as much of it as you have entered at that point) is placed beginning at the left margin defined in the Environment.

In the Editor, a word is any text character or characters bounded by any two word delimiters, where a word delimiter is a space, a RETURN character, the beginning or end of the file, or the beginning or end of the current insertion (before you press CONTROL-C). The hyphen is not a recognized word delimiter. If you enter two or more RETURN characters in succession, the next text begins at the set Paragraph margin.

This setting of the Environment also causes the Editor to adjust the margins on the portion of the paragraph following an insertion (but not the paragraph portion preceding the insertion). The Editor considers a paragraph to be any text bounded by any two paragraph delimiters, where a paragraph delimiter is a blank line (created by two RETURN characters), a line beginning with the Command character (set in the Environment), or the beginning or end of the file.

The automatic re-margining following an insertion can sometimes cause you much grief. If you are editing in or near a diagram, table, or other carefully formatted portion of text, it is a good precaution to set Filling temporarily to False (just enter SEFF<ctrlC>). This prevents an insertion from scrambling your beautiful diagram into a paragraph of meaningless text.

EXAMPLE:

In document mode, the Left margin at Ø, and Right margin at 13, enter I for Insert and then enter the following:

```
WISH I WEREN'T A WASH-AND-WEAR WARRIOR
```

This should create the text format shown at the left, below:

```
WISH I          Auto-returned when next word would exceed margin
WEREN'T A       Auto-returned when next word would exceed margin
WASH-AND-WEAR   Auto-returned at first possible break, even though
WARRIOR         beyond margin
```

Inserting with Indent-auto and Filling both True

With this setting of the Environment, Indent-auto controls the left margin, ignoring the settings of the Left margin and Paragraph margin. Filling inserts RETURN characters as before, to keep lines from exceeding the set Right margin.

However, Filling operates only to keep the CURRENT insertion from exceeding the Right margin. Any text on the same line, but to the right of the cursor, may extend beyond the Right margin or even beyond the 8Ø characters visible on the Apple's display. The existence of characters beyond position 8Ø is indicated by an exclamation mark (!) displayed at the rightmost position on the screen. To see the hidden characters, insert a RETURN character anywhere in the visible portion of the line, or Margin the paragraph.

Changing the indentation can be done as before, by entering a space or left-arrow, but only as the FIRST character in a new line (not likely, since Filling generally begins a line with the last entered word). You can delete a line and indent to the left margin by using CONTROL-Q. This setting of the Environment is not usually very helpful, as its effects can be better obtained in other ways.

Inserting with Indent-auto and Filling both False

With this setting of the Environment, the Editor ignores the margins set in the Environment. You must enter into the text all margins, indentations, and RETURN characters.

As in Programming Mode, if you attempt to enter text beyond position 71, the computer beeps to warn you. If you attempt to enter text beyond position 8Ø, an exclamation mark (!) appears

at the rightmost position on the screen. This character at the end of any line indicates that the line contains more than the 8Ø characters that can be displayed on the screen. Additional characters entered into that line are not lost, but they are not displayed.

To see the hidden characters, you can insert a RETURN character anywhere in the visible portion of the line; or you can use the Margin command.

Delete

The Delete command removes text from the file.

To delete text, enter D for Delete while at the Edit level. After you enter D, the following prompt line appears:

```
>Delete:  <Moving keys>, <ctrlC> accepts, <esc> escapes
```

Prior to entering D, make sure that the cursor is in the correct position to begin the deletion. Editor commands begin their action immediately to the left of where the cursor appears to be positioned. (See Figure 4-5 below.)

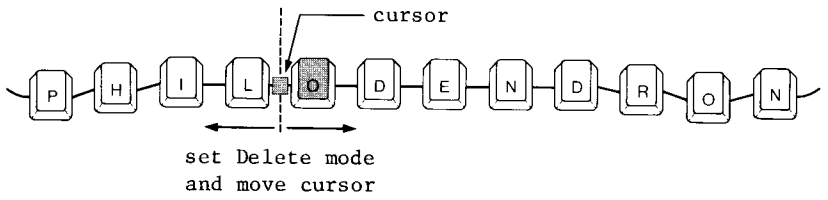


Figure 4-4. Cursor Action with Delete Command.

If you position the cursor at the middle of a line of text, moving the cursor to the right will delete characters beginning with the first character that the cursor appeared on. Moving the cursor to the left will begin deleting text with the first character to the left of the cursor.

Remember, the cursor is always between the character it appears to be on and the next character toward the beginning of the file.

The Editor remembers where the cursor is when you enter D for Delete. This position is called the anchor. As the cursor moves away from the anchor in any direction, using the normal cursor moves, all text between the cursor and the anchor disappears. When you move the cursor toward the anchor, the erased characters reappear. The repeat-factor can also be used to delete or undelete several lines at once, by prefacing a RETURN or any other cursor move with a repeat-factor while the Delete prompt line is showing.

To accept the deletion at any point, press CONTROL-C. To undo the entire deletion at any time before pressing CONTROL-C, press the ESCAPE key.

Unlike inserting text, deleting text does NOT cause re-margining of the portion of the paragraph following the deletion, even if the Environment's Filling option is set to True and Auto-indent is set to False.

After a deletion that included a RETURN character, the line containing the cursor may extend beyond the 8Ø-character limit of the system's display. The invisible portion of the line is indicated by an exclamation mark (!) in the last visible character-position of the line. To see the rest of the line, insert a RETURN character anywhere in the visible portion of the line, or use the Margin command to reformat the entire paragraph.

All the text between the cursor and the anchor position is stored in the Copy buffer, ready for use by the Copy command, not only after you accept the deletion with CTRL-C, but also after you reject the deletion by pressing the ESCAPE key. This last fact is useful when you want to duplicate some text in another location, or when you are moving some text to another location but wish to keep a backup copy of the text until the move is successfully completed. If you mistakenly press D to delete, press ESCAPE. The copy buffer will not be changed.

The Copy buffer may be unable to hold all the deleted text if you attempt to delete too much text at one time. The maximum amount of room in memory for the Copy buffer varies somewhat, depending on how large your file is at the moment. If you try to delete more text than will fit in the Copy buffer, this message appears:

```
There is no room to copy the deletion. Do you wish to delete anyway? (y/n)
```

If you enter Y for "Yes", the text between the cursor and the anchor position is deleted, but that text does not go into the Copy buffer. If you enter N for "No", the deletion is not performed, and the text is not placed in the Copy buffer. After a response of either Y or N , the Copy buffer remains with the same text it contained before you initiated the Delete command.

EXAMPLE:

Suppose you are editing the following text:

```
PROGRAM STRING2;
BEGIN
  WRITE('TOO WISE ');
  WRITELN('TO BE.')
END.
```

1) Move the cursor onto the E in END.
2) Type D for Delete.
3) Type < . This changes the set direction to backward.
4) Press the RETURN key twice. After the first RETURN the line "WRITELN('TO BE.')" disappears. After the second RETURN, the line "WRITE('TOO WISE')"; disappears.
5) Now press CTRL-C. The program after deletion appears as shown:

```
PROGRAM STRING2;
BEGIN
END.
```

The two deleted lines have been stored in the Copy buffer, and the cursor has returned to the anchor position. Now type CB to place the contents of the Copy buffer back in the file, then type CB again several times to write multiple copies of the deleted lines to the file:

```
PROGRAM STRING2;
BEGIN
  WRITE('TOO WISE ');
  WRITELN('TO BE.')
  WRITE('TOO WISE ');
  WRITELN('TO BE.')
  WRITE('TOO WISE ');
  WRITELN('TO BE.')
  WRITE('TOO WISE ');
  WRITELN('TO BE.')
END.
```

This leaves you with something like the example above, depending on how far you get carried away with the command.

Note: after pressing CTRL-C, if you immediately copy the deletion without moving the cursor, the deleted material is just replaced. This gives you one more chance to recover from a mistaken deletion.

Zap

The Zap command deletes all text between the current cursor position and the point. The point is the text position of the first character of the most recent Find, Replace, Insert, or Set Point command.

You execute the Zap command by typing Z for Zap while at the Edit level. Because several different commands can move the point, you can check the beginning and end of the text to be Zapped by typing JPJP for Jump Point Jump Point.

If you Zap more than 8Ø characters, the Editor asks for verification:

```
WARNING!  You are about to zap more than 8Ø chars, do you wish to zap?  (y/n)
```

Repeat-factors and Zap: If the Find or Replace command is used with a repeat-factor, only the last string found or replaced will be deleted by Zap. All the other strings remain as found or replaced.

All the text that is deleted by using the Zap command goes into the Copy buffer, where it is available for use with the Copy command (until the next insertion or deletion).

If you attempt to use Zap to delete too much text at one time (the maximum amount varies somewhat, depending on how large your file is at the moment), the Copy buffer may be unable to hold all the deleted text. In that case, when you enter Z to Zap a very large block of text, first this message appears:

```
WARNING!  You are about to zap more than 8Ø chars, do you wish to zap?  (y/n)
```

and then, if you enter Y for "Yes", this message appears:

```
There is no room to copy the deletion. Do you wish to delete anyway? (y/n)
```

If you enter Y for "Yes", the text between the cursor and the point is deleted, but that text is not placed in the Copy buffer. If you enter N for "No", the deletion is not performed, and the text is not placed in the Copy buffer.

Copy

This command copies text from the Copy buffer or from a diskette file. To use the Copy command, enter C for Copy at any time the Edit prompt line is showing. When you enter Copy mode, the following prompt line appears:

```
>Copy:  B(uffer, F(ile portion, M(arkers, <esc> escapes
```

Copying From a File

Enter C for Copy and then F for File portion, to copy text stored in a file (including the one you are working in) and insert it in your file. When you do this another prompt line appears:

```
>Copy: From what file?
```

Any on-line file may now be specified. Use of the Copy command does not change the contents of the file from which you are copying. You may enter the filename's .TEXT suffix or not, as you wish, since .TEXT is automatically supplied if you do not enter it into your pathname. To suppress this feature (when copying from a file whose name does not end in .TEXT), enter a period following the complete pathname.

To copy only part of a file, two markers must have been set (from the Editor) to bracket the desired text. (See the section titled Set Marker under Miscellaneous Commands.) After you enter F for File pathname, you see the prompt:

```
>From what marker?
```

Enter the name of the first marker that you had set in that other file during an earlier editing session.

If your response to the prompt line above is just a <return> character, the copy will start with the beginning of the file. You then see the following prompt:

```
>To what marker?
```

Enter the terminal marker of the information you wish to copy into your current file. Entering RETURN now will copy all the file from the previously-specified point to the end of the file. You can abort the Copy operation by pressing ESCAPE at any time before the final RETURN is typed.

If you have not placed the markers that you thought you had, you see the following message:

```
ERROR:  Marker not there.  Please press <space> to continue.
```

Pressing the space bar returns you to the Edit level.

On the completion of the Copy command, after text has been copied from the specified file, the Editor places the cursor before the first character of the text that you copied.

If your present file cannot contain all the additional text that you are attempting to copy into it (maximum size of a file on a 128K Apple III is about 54 diskette blocks), the Editor copies in as much of the additional text as it can. Then it gives this message:

```
ERROR: Buffer overflow.  Please press <space> to continue.
```

When you press the space bar, the copy is complete, and your file now contains as much of the additional text as the Editor could fit into your file. Remember that none of the target file's Environment information is brought over with the copied text.

You can copy from an asciifile regardless of the type of file being edited. There will be no requests for markers since an ascii file has no markers. The Copy command will copy the entire asciifile in to your file currently being edited.

EXAMPLE:

Suppose the diskette named /MYDISK/ contains a file named SUPERMART.TEXT , which has two markers placed in its text: ALPHA and BETA . Further suppose that you are now in the Editor, editing a file, and you wish to insert the text of SUPERMART.TEXT , bounded by markers ALPHA and BETA, at the current cursor position.

In response to the Editor prompt line, you first enter C to enter Copy mode, and then enter an F to select copying from a File portion. This prompt line then appears:

```
>Copy: From what file?
```

To cause the planned insertion, enter

```
/MYDISK/SUPERMART<return>
ALPHA<return>
BETA<return>
```

Copying From the Copy Buffer

Each time you insert or delete text, you are also storing that text in the Copy buffer, sometimes called the insert-delete buffer. To use the text in the Copy buffer, enter C to enter Copy mode and then enter B for Buffer. The Editor immediately copies the contents of the Copy buffer into the file at the current location of the cursor (that is, between the character on which the cursor sits and the first character to the cursor's left). Use of the Copy command does not change the contents of the Copy buffer.

Upon completion of the Copy command, after text has been copied from the Copy buffer, the cursor is placed before the first character of the text that you copied.

Unlike inserting text, copying text does NOT cause re-margining of the portion of the paragraph following the Copy, even in document mode. After copying, some lines may extend beyond the 8Ø-character limit of the Apple III's display.

After an insertion has been made, you can use the Copy command to duplicate the section of text just inserted as many times as desired. Use the Copy command, too, to move text from one location in the file to another. Delete the text from its present location, then move the cursor, and copy the deleted text into its new location.

The contents of the Copy buffer are affected by the following commands:

1) Delete: When you accept a deletion (with CONTROL-C), the Copy buffer is loaded with the deleted text. When you reject a deletion (by pressing the ESCAPE key), the Copy buffer is loaded anyway, with the text that would have been deleted had you accepted the deletion. However, if you enter D to Delete some information but then enter the ESCAPE key before moving the cursor over any characters, there will be no effect on the Copy buffer. It will retain the information from your previous insertion, deletion, or zap. Similarly, if you enter D to Delete and actually move the cursor through several characters but then backspace to your point of origin and press ESCAPE, your Copy buffer will not be affected.

   If there is not enough room in memory, the Copy buffer will not be loaded by pressing ESCAPE and you will lose the deleted text forever.

2) Insert: When you accept an insertion with CONTROL-C, or reject it with ESCAPE, the Copy buffer is loaded with the inserted text. If there is not enough room in memory, the copy buffer may not be loaded after pressing ESCAPE.

3) Zap: When you delete text using the Zap command, the Copy buffer is loaded with the deleted text.

4) Margin: Using the Margin command destroys the contents of the Copy buffer.

If you have used most or all of the available file workspace (about 54 blocks with a 128K Apple III), if an attempted Copy causes either

```
ERROR: Invalid copy. Please press <space> to continue.
```

or

```
ERROR: No room to copy. Please press <space> to continue.
```

to be displayed, it is probably time to either begin a new file or to split the file you are working on into two parts.

Exchange

You use the Exchange command to replace characters in a line by entering new characters.

Press X to use the Exchange command. After doing this, the following prompt is displayed:

```
>eXchange: <text>, <bs> a char, <right arrow> copies, <ctrlC>, <esc>
```

As you enter characters from the keyboard, the cursor moves to the right along the line of text and replaces existing characters with the characters you are entering. After

finishing your change to a line of text, you can accept the changes by pressing CONTROL-C or reject them by pressing ESCAPE.

You can use the right-arrow key to copy over existing text without changing it, and use the left-arrow key to back up and restore old characters one by one. Exchange is not affected by the set direction, and can be used for only one line at a time.

Replace

The Replace command replaces a specified string with a specified substitute string. Its operation is very similar to Find, except that the target string can be replaced with a substitute string after being found.

To use the Replace command enter R for Replace while at the Edit level. One of the following two prompt lines will then appear, depending on the setting of Token default in the Environment (see Set Environment in the Miscellaneous Commands section):

```
>Replace[l]: L(it V(fy <targ> <sub>  =>
```

if the Environment's Token default is set to True, or

```
>Replace[l]: T(ok V(fy <targ> <sub>  =>
```

if the Environment's Token default is set to False.

The Replace command searches through a file in the set direction to find the specified number of occurrences of the specified target string of characters, and replaces each of those occurrences (after verification, if that option is chosen) with the specified substitute string of characters. When finished, it places the cursor at the end of the last string found and/or substituted.

Set Direction

The Replace command searches for a specified number of occurrences of the target string beginning at the present cursor position and scanning through the text in the set direction (indicated by the arrow at the beginning of the prompt line). The number of occurrences searched for is determined by the repeat factor that you specify. An occurrence of the target string will be found only if it appears in that portion of the text which lies between the cursor and the end of the file toward which the search is progressing. See the section on the

set direction (in this chapter, under General Information) in order to change the set direction arrow. If the end of the file is reached before the specified replacement can be carried out, this message appears:

```
ERROR: Pattern not in rest of the file.  Please press <spacebar> to continue.
```

Notice, however, that the search does not "wrap around". That portion of the file between the cursor and the end of the file in the direction OPPOSITE the set direction is not searched.

Repeat-factor

The repeat-factor is allowed when using the Replace function.

Literal or Token Search

The target string is treated differently, depending on whether Literal search or Token search is selected. When you select Literal search, the Editor will replace any occurrence of a string of characters that exactly matches the specified target string. If Token search is selected, the Editor looks for isolated occurrences of the target string, ignoring spacing. See Set Environment in Miscellaneous Commands for more details about the difference between Literal and Token search.

The default setting of the search option is set in the Environment. The Replace prompt line indicates only the alternate choice of search mode. If you do not specify a search option, the default search option (the one which is NOT mentioned on the prompt line) is used. To use Token search when the default is Literal search (prompt line says T(ok), enter T after the prompt line and before the target string. To use Literal search when the default is Token search (prompt line says L(it), enter L before entering the target string.

Note: Nothing appears to happen when you enter L or T ; the letter just appears where you are about to enter the target string.

Target Strings and Delimiters

The Editor has two string storage variables. The first string variable, called <target> or <targ> by the prompt line, contains the "target" string, and is used both by the Find command and by the Replace command. The target string is the sequence of characters which will be searched for by the Find command, or searched for and replaced by the Replace command. The second

string, used only by the Replace command, is called <sub> by the Replace prompt line and is the "substitute" string. In the Replace command only, the substitute string is the sequence of characters that will replace the target string when the target string is found.

To allow the target and substitute strings to contain almost any characters (including RETURN characters), each string must be entered using special rules. In particular, each string must be set off by characters called "delimiters". Both delimiters of a string must be the same character. One delimiter must precede the first character of the string, and the same delimiter must follow the last character of the string.

The Editor allows almost any normal printing character that is not a letter or a number to be a delimiter as long as it does not occur in the string delimited. This lets you choose the delimiter. The most common choice is the slash (/) because it is a lower-case character that is not commonly found in text, and it is easy to enter.

Once you have entered the initial delimiter character for either the target or the substitute string, you cannot backspace (using the left- arrow key) to erase that character or any of the preceding characters in your response. If you forget to precede either the target string or the substitute string by a correct delimiter character, you will be told.

```
Error: Invalid delimiter.  Please press <spacebar> to continue.
```

You will get the same message if you try to backspace (by pressing the left-arrow key) immediately after entering the target string's final delimiter. Just try the whole command again, and this time use correct delimiters.

Verify Option

The Verify option (shown as V(fy on the Replace prompt line) permits examination of each target string as it is found, before the replacement is carried out. You can then decide whether this occurrence of the target string is to be replaced or not. To select the Verify option when using Replace, enter V before entering the target string. Nothing will appear to happen when you enter V, but the Verify option will be selected anyway. The following prompt line appears whenever the Replace command has

found an occurrence of the target string in the file and Verify has been requested:

```
>Replace: <esc> aborts, 'R' replaces, ' ' doesn't
```

Typing an R at this point will cause the specified replacement to be carried out, while pressing the spacebar will cause the system to search for the next occurrence of the target string, provided the specified repeat-factor (or the end of the file) has not been reached. Entering <esc> aborts the Replace operation. The repeat-factor specifies the number of times an occurrence of the target string will be found, not the number of times you actually cause its replacement. Use / as the repeat-factor to examine every occurrence of the target string in the set direction.

Literal or Token Search

Replace uses the Literal and Token search modes exactly as does the Find command.

ESCAPE Option

At any time during your response to the Replace prompt, you can abandon this command and return to the Edit level by pressing the ESCAPE key.

Same-string Option

Typing S in place of the delimited target string tells the Replace command to use the target string that was last specified. The target string may have been specified either by the Find command or by a previous use of the Replace command. Similarly, entering S in the place of the delimited substitute string tells the Replace command to use the same substitute string that was last specified by a previous use of the Replace command. For example, with the Replace prompt line at the top of the screen, entering

```
S/<any-string>/
```

causes the Replace command to use the previous target string (and a new substitute string), while entering

```
/<any-string>/S
```

causes the previous substitute string to be used (and a new target string). From the Editor, entering the command

RVSS

says "Do it again": it causes the next occurrence of the previously specified target string to be replaced (after verification) with the previously specified substitute string.

If you have not previously specified a substitute (or target) string, you will be informed

'no old target'

or

'no old substitute'

Note: The Environment (see Set Environment, in Miscellaneous Commands) shows you the current <target> and <subst> strings which will be invoked by entering S as a string response.

EXAMPLE:

Suppose you wish to replace the next three occurrences of the target string APPLE with the substitute string BANANA, assuming that the set direction is >, and Literal search is true :

With the Edit prompt line showing, you would enter

3R

to indicate a repeat-factor of 3 and then to select the Replace command. In response to the Replace prompt line:

>Replace[3]: T(ok V(fy <targ> <sub> =>

you could enter

/APPLE/)BANANA)

In this example, first the character / is used as the beginning and ending delimiter for the target string, and then the character) is used as the beginning and ending delimiter for the substitute string. In the example, two different delimiters

were used for pedagogical purposes. In practice you would be more likely to use

/APPLE//BANANA/

If you now wish to Replace five more occurrences of the target string APPLE, but this time with the substitute string PAPAYA , just enter, with the Edit prompt line showing,

5RS?PAPAYA?

After a brief flash of this prompt line

>Replace[5]: T(ok V(fy <targ> <sub> =>/APPLE/?PAPAYA?

the requested replacements will be carried out.

EXAMPLE:

Now assume that Token mode is true; if you enter

RL/QX//YZ/

when the Edit prompt line is showing, this prompt line should appear:

>Replace[1]: L(it V(fy <targ> <sub> =>L/QX//YZ/

If the cursor is before the V in VAR, this command will change the program line

VAR SIZEQX:INTEGER;

to

VAR SIZEYZ:INTEGER;

You must select the non-default Literal search option (by entering L before entering the target string) because the string QX is not a Token but is part of the Token SIZEQX.

Formatting Commands

Adjust

The Adjust command adjusts the indentation of a line or a whole group of lines.

To use the Adjust command enter A for Adjust while at the Edit level. The following prompt line will then appear:

```
>Adjust: L(just R(just C(enter <moving keys>, <ctrl C> accepts, <esc> escapes
```

The right-arrow and the left-arrow keys can be used to push the line right and left, or you can adjust the line to the Left margin, the Right margin, or the Center. Moving the cursor up or down makes the same adjustment to lines above or below. Use of a repeat-factor is valid with all cursor moves.

Each time the right-arrow key is pressed when the Adjust prompt line is at the top of the screen, the line the cursor is sitting on moves one space to the right. The line can be moved beyond the Right margin set in the Environment. Characters moved beyond the 8Ø-th character position are not displayed, but their existence is indicated by an exclamation mark (!) in the 8Ø-th character position of the line.

Each time the left-arrow key is entered, the whole line moves one position to the left. The line can be moved beyond the Left margin set in the Environment, but the leftmost character cannot be moved beyond the left edge of the screen display character position zero.

When the line is adjusted to the desired indentation press CONTROL-C .

Note: ESCAPE can be used to cancel the adjustment of the current line. You accept the adjustment by pressing CONTROL-C or by moving to another line.

In order to adjust a whole sequence of lines, first adjust the top or the bottom line, then BEFORE entering (CONTROL-C) use the up or down arrow keys. The line above or below will automatically be adjusted by the same amount when the cursor jumps to that line. Finally, when the entire sequence has been adjusted, enter CONTROL-C.

Repeat-factors, including / , are valid when used before any of the cursor moves while in Adjust mode.

The Adjust command can also be used to center text on the page and to left-justify or right-justify text (force all the lines to make a smooth left margin, like this page, or a smooth right margin). Entering L for L(just while the Adjust prompt line is showing causes the line containing the cursor to be

left-justified by moving the leftmost non-space character to the Left margin set in the Environment.

Similarly, entering R for R(just right-justifies the line by moving the rightmost text character to the set Right margin. Entering C for Center causes the line to be centered between the set Left and Right margins. Entering the up-arrow or down-arrow key before CONTROL-C is entered will cause the line above or below to be adjusted to the same specification (left-justified, right-justified or centered) as the previously adjusted line.

EXAMPLE:
Now insert a line to practice shoving it around with the Adjust command:

```
    My name is Caspar Milquetoast.
```

After you have typed CONTROL-C to accept the insertion, type AL<CONTROL-C>, moving your line to the left margin:

```
My name is Caspar Milquetoast.
```

Now type AR<CONTROL-C> to shove the line to the right margin:

```
                                    My name is Caspar Milquetoast.
```

Now type AC<CONTROL-C> to center the line:

```
                    My name is Caspar Milquetoast.
```

Now type AL<left-arrow><left-arrow><left-arrow><CONTROL-C> to drag the line to the fourth column from the left margin:

```
   My name is Caspar Milquetoast.
```

Now that you've satisfied the brute within you, let us continue on.

Margin

The Margin command adjusts a paragraph by expanding it as much as possible without exceeding the the margins set in the Environment. You execute Margin by entering M for Margin while at the Edit level.

Margin is an Environment-dependent command that has three parameters, all set in the Environment: Right margin, Left

margin and Paragraph margin. See Set Environment under Miscellaneous Commands for how to set the margin values.

In addition to this, Margin assumes that if you have Filling set to false, or Indent-auto set to true, that you won't want to remargin a paragraph by accident. If you press M for Margin while not in document mode, the Editor will display:

Inappropriate environment. Margin this paragraph anyway? (y/n)

You would find this most useful while you are creating a document with diagrams or tables (which don't need or even want to be margined!) included with normal text.

The Margin command affects only the paragraph containing the cursor. A paragraph is defined to be any text bounded above and below by paragraph delimiters, where a paragraph delimiter is a blank line (created by two consecutive RETURN characters), the beginning of the file, the end of the file, or a line which starts with the Command character that is currently set in the Environment. Unless you change it (see Set Environment), the Command character is by default the carat(^).

To margin a paragraph, move the cursor to anywhere in that paragraph and enter M . The screen blanks while the Margin command is busy shuffling the paragraph. When doing an exceptionally long paragraph, it may take several seconds before the routine is ready to redisplay the screen. When breaking lines to avoid exceeding the right margin, the Margin command recognizes all spaces as possible points to break the line. All other characters in sequence are considered words, and will not be broken.

The Margin command does not recognize hyphens as possible line break points, nor does it know how to correctly introduce hyphens into words that do not already contain them.

The following characters: period, question mark, colon, exclamation point, or any of those characters immediately followed by a close-parenthesis or double quote, will be followed by exactly two spaces after a Margin command. This might cause some inconvenience with abbreviations.

The Margin command will compress all groups of spaces between words into single spaces.

EXAMPLE:

The paragraph below has been typed with these Environment parameters set:

Left margin	Ø
Right margin	64
Paragraph margin	8

When you operate a skateboard in excess of 15Ø miles per hour, certain problems are encountered. First of all, the number of traffic citations becomes excessive, unless your skateboard is equipped with either a working radar detector or set of flashing red lights. Secondly, goggles and knee protectors often blow away and skateboards have been known to become airborne. Lastly, you may have to endure the ire of Porsche and Ferrari drivers, since they become depressed, angered, and sometimes say uncomplimentary things when passed by a person on a skateboard.

Next, the same paragraph is shown after being margined with these parameters set in the Environment:

Left margin	1Ø
Right margin	5Ø
Paragraph margin	Ø

When you operate a skateboard in excess of 15Ø miles per hour, certain problems are encountered. First of all, the number of traffic citations received gets out of hand, unless your skateboard is equipped with either a working radar detector or set of flashing red lights. Secondly, goggles and knee protectors often blow away and skateboards have been known to become airborne. Lastly, you have to endure the ire of Porsche and Ferrari drivers, since they become depressed, angered, and sometimes say uncomplimentary things when passed by a person on a skateboard.

Miscellaneous Commands

Verify

The Verify command verifies the contents of the Editor by redisplaying the screen.

The Verify command is executed by entering V for Verify while at the Edit level. There is no indication of the Verify command on the Edit prompt line. The status of the Editor is verified by redisplaying the screen. The Editor attempts to adjust the window so that the cursor is at the center of the screen. This command can also be psychologically helpful. Enter it whenever you are unsure that the screen really corresponds to what is in your file. After entering V the screen reflects what is really in your file.

Set

The Set command is used to either access the Environment or to set a marker in the text.

To use the Set command enter S for Set while at the Edit level. The following prompt line will appear:

```
>Set:  E(nvironment, M(arker, P(oint, <esc> escapes
```

Note that there is no indication of the Set command on the Edit prompt line.

Set Point

Set Point allows you to position the marker called Point anywhere in the file that you want to place it. Point is used or modified by the Insert, Find, Zap, or Replace commands.

Set Marker

When you are editing a large file, it is particularly convenient to be able to jump directly to certain places in the file by using markers that have been set in the desired places. Once set, it is possible to jump to these markers at any time, by using the Marker option with the Jump command (see Moving Commands).

The Copy File portion and Copy Marker commands can also make use of markers that have been placed in the text of a file. When you are editing one file, the marked portion of a second file that is stored on diskette may be copied into the file you are editing (see Text Changing Commands).

Move the cursor to any spot in the text where you want to place a marker. When the cursor is in the desired spot, type SM for Set Marker. The following prompt line appears:

```
Set what marker?
```

Now type the name of the marker (up to eight characters) that you want placed at the current cursor position, terminated by pressing the RETURN key. Any printable character except a carriage return may be used in a marker name, but all lower-case letters are converted to upper-case letters.

If you have already placed a marker with the specified name in the file at an earlier time, the old marker is moved to the current cursor position without comment, and the old position is lost.

Only ten markers are allowed in a file at any one time. If you attempt to place an eleventh marker, the following message appears:

```
Marker overflow.  Which one to replace?
Ø) name1
1) name2
.  ...
.  ...
9) name1Ø
```

You must eliminate one of your existing markers before you can place the new one. Choose a number from Ø through 9, enter that number and its place in the list will now be available for your new marker name. You can use this method to rename or re-place an existing marker, but you can never simply remove a marker from your file, even if you delete all the text that contained the marker.

Set Environment

The Editor lets you set various aspects of the editing "environment" to suit the task at hand. From the Edit level, enter S for Set and then enter E for Environment. The screen

display is replaced with a prompt similar to the one shown below:

```
>Environment: <options>,<ctrlC> accepts, <esc> escapes
    I(ndent auto  True
    F(illing      False
    L(eft margin  Ø
    R(ight margin 79
    P(ara margin  5
    C(ommand ch   ^
    T(oken def    True
    A(scii file    False
    N(ame of file /GNUDISK/ANTELOPE.TEXT

    File is 488 characters long with 26647 more available.
    Currently at character position 5Ø.

    Patterns:
      <target>= /APPLE/, <substitute>=  /BANANA/

    Markers:
        START          PART3          SUMMARY
        INTRO          MAINPARA       BIBLIOG
        ACKNOWL        PART 5         INDEX

    Date Created: 4-Ø1-84   Last used:  7-28-85
```

By entering the appropriate first letter, any or all of the options listed in the upper portion of the display may be changed.

The portion of the display showing the Patterns: <target> and <subst> will not appear unless you have used the Find or Replace commands since entering the Editor this time. The portion of the display showing the markers currently in the file will not appear unless you have at some time used the Set Marker command to place a marker in the text.

The information stored in the Environment (with the exception of the <target> and <subst> strings) is saved in the file header each time you save the file on diskette, so the system can "remember" that environment each time you work on that file again.

The Editor will not accept Environment option choices having an improper format.

If you enter a non-numeric choice for L(eft. R(ight, or P(aragraph margins the Apple will beep and the message

Digit, <bs>, <ret>, or <esc>

is displayed. You can then press either RETURN or Left-arrow to clear the error and try again or press ESCAPE to leave the option at its original value.

If you give any answer to I(ndent auto, F(illing, T(oken search, or A(scii file, except T (for true) or F (for false), the message

T, F, or <esc>

show you your operating choices. Enter the correct response or press ESCAPE to leave the option in its original state.

Do as the prompt suggests, and either exit the Environment menu, or type correct values and then exit by typing CONTROL-C to accept the changes you have made, or by typing ESCAPE to reject the changes.

Each of the following options must be accessed from the Editor's Environment. To enter the Environment, enter S for Set and then E for Environment.

Indent-auto
Indent-auto affects only the Editor commands Insert (under Text Changing Commands) and Margin (under Formatting Commands). See the discussions of those commands for more details and examples.

The Indent-auto option is set to True (each new line is automatically started at the same indentation as the first non-space character of the previous line) by your entering AT .

The Indent-auto option is set to False (new lines begin at the screen's left edge or at the set Left margin and Paragraph margin) by your entering AF . Unless Indent-auto is False (and Filling is True), the Insert command will not cause re-margining of the portion of a paragraph following an insertion.

Indent-auto should generally be True for writing and editing programs, and False for writing and editing natural language text.

Filling
Filling affects the Editor's Insert command (under Text Changing Commands).

When the Filling option is set to True lines are automatically broken between words -- at spaces -- when they are entered in. This prevents lines from exceeding the right margin. To set Filling to True, enter FT. Unless Filling is True (and Indent-auto is False), the Insert command will not cause re-margining of the portion of a paragraph following an insertion.

The Filling option is set to False (the set margins are ignored; you must end each line yourself) by entering FF.

Filling should generally be False for writing or editing programs, and True for writing or editing natural language text. However, if you are editing a table, diagram, or other carefully formatted portion of text, it is a very good safety precaution to set Filling to False (from the Edit level, just enter SEFF<ctrlC>). This will save you the frustration of having your text completely re-formatted following an insertion.

Left margin
Right margin
Paragraph margin

In Document Mode, the margins set in the Environment are the margins which are used by the Insert command (see description under Text Changing Commands) and the Margin command (see description under Formatting Commands). These margins also affect the Center, Left, and Right justifying commands in the Adjust command (see description under Formatting Commands). See the discussions of those commands for more details and examples.

To change the value for the Left margin option, enter L followed by an unsigned integer, and then press RETURN. The value that you enter replaces the old value for the Left margin in the prompt display shown at the beginning of this section.

To change the value for the Right margin option, enter R followed by an unsigned integer, and then press RETURN. Similarly, you can change the value of the Para margin option by entering P followed by an unsigned integer, and then press RETURN.

All unsigned integers with four or fewer digits are valid margin values. If you attempt to assign a margin value of more than four digits, the value will be truncated to the first four digits entered. To create normal text displays whose characters are all visible on the screen, you should use margin values from Ø through 79, and the Left and Paragraph margin values should be less than the value of the Right margin.

Command character
The Command character affects the Margin command (under Formatting Commands) and re-margining in the Insert mode (under Text Changing Commands). See the discussions of those commands for more details.

To change the setting of the Command ch option, enter C followed by any printing character. For example, entering C* will change the set Command character to * . This change will be reflected in the Environment screen.

If the Command Character appears as the first non-blank character in a line of text, then that line is protected from the Margin command, and from re-margining following an insertion. That line is also treated as a paragraph delimiter for margining purposes. The normal Command character is the caret or circumflex accent (^). Unless you have some special use for the caret character in your text, you should generally leave it as the set Command character.

Token default
Token default affects the search option used by the commands Find and Replace (under Text Changing Commands). See the discussions of those commands for more details and examples.

In the Environment, Token def is set to True (the default search option is Token search) by entering TT, and to False (the default search option is Literal search) by entering TF.

When the Literal search option has been selected, the Editor will look for ANY occurrence of a string of characters that exactly matches the <target> string. When the Token search option has been selected, the Editor will look for ISOLATED occurrences of the <target> string. The Editor considers a string isolated if it is surrounded by any combination of delimiters, where a delimiter is any character that is not a number or letter.

For example, in the sentence "Put the book in the bookcase.", using the <target> string "book", the Literal search option

will find two occurrences of "book" while Token search option will find only one, the word "book" isolated by the delimiters <space> <space>.

When the Token search option has been chosen, you can find an occurrence of the <target> string, even if the occurrence has more spaces or fewer spaces (including zero) corresponding to each space in the specified <target> string. For example, suppose you are searching the following text, which contains four slightly different occurrences of the words "APPLE PIE":

```
I'LL HAVE SOME A PPLEPIE, SOME APPLE
PIE, SOME APPLEPIE, AND THEN
SOME AP PLE PIE, TOO.
```

If you use the <target> string "APPLEPIE", a Token search will find only the third occurrence. With the <target> string "APPLE PIE" , a Token search will find both the second occurrence (which has more spaces, but at the right place in the string) and the third occurrence (which has fewer spaces, and none in the wrong place). With the <target> string "A P P L E P I E" , a Token search will find all four occurrences.

However, only a Literal search would find an occurrence of "APPLE PIE" that was buried in the word "CRABAPPLE PIE". That's because the "B" would not constitute a proper isolating delimiter.

When editing natural language text, it is a good idea to use Literal search (set Token default to False). When editing programs, it is usually more useful to use Token search (leave Token def set to True).

Ascii File
BASIC programs can be edited, but the files used are not standard text files, but Ascii files. See the description of editing BASIC files and editing Ascii files earlier in this chapter.

Name of File
You can change the name of the file you want to save your current editing session under. The name of the file shown in the Set Environment display was set either by the Filer's Get command, the Editor's Quit Update command (as SYSTEM.WRK.TEXT), or declared by you when you started the edit session with no file specified for Get. If you change the file name here, the file name used by the Save option of the Quit command will be

changed to the new file name. If you are in textfile mode, .TEXT will be appended to the name, but if you are in asciifile mode, the name will remain as entered.

Number of Characters
This line tells you how long your file currently is and how much space remains in the Editor's file space. The current position of the cursor is also displayed, giving you an idea of where you are in the file.

Quit

The Quit command is used to exit from the system's Edit level.

To use the Quit command type Q for Quit while at the Edit level. The following message will then appear:

```
>Quit:
      To leave Editor, type
            E(xit to main command line

      To store text file on disk, type
            W(rite to a new file name
            U(pdate /NEWPASCAL2/SYSTEM.WRK.TEXT

      To continue editing, type
            R(eturn to same file
            C(hange to another file
```

You select the desired option by entering the first letter of the option as given in the display. Another option, Save becomes available after you have once written a file to a pathname or if you have set a file in the Environment to write to.

Update

This tells the Editor to erase all previous versions of the main system diskette's workfile (SYSTEM.WRK.CODE as well as SYSTEM.WRK.TEXT). Then it saves on the main system diskette, under the filename SYSTEM.WRK.TEXT, a backup copy of the file currently in memory.

If you are using SYSTEM.WRK.TEXT as your text file, the Update command should be used at least every 15 minutes, in order to prevent accidental loss of your efforts. From the Editor, every so often, just type QUR. In a few seconds, the main system diskette's file SYSTEM.WRK.TEXT will contain the latest

version of your file, and you will again be in the Editor, ready to continue working on your backed-up workfile.

Exit

This causes the system to leave the Editor without saving the file in memory that is currently being worked on. This means that any modifications made since last writing to the file are irretrievably lost. After answering Y, or if there have been no changes, the system returns to the Command level.

Return

This option lets you return directly to the Editor without updating. The cursor is returned to the exact place in the file it occupied when Q was typed. Usually this command is used after unintentionally typing Q, or when saving changes made to the file to disk.

Change

This lets you switch from the file you are presently working on to another file without leaving the Editor. After typing C for change, this prompt appears:

```
>Edit:
File? (<ret> for no file, <esc> to exit editor)
:
```

If you want to start a brand new file, press RETURN. Otherwise type the pathname of the file that you want to begin working on, and continue as if you had entered the Editor from the main Command level.

If you change your mind, press ESCAPE to leave the Editor and return to the Command level.

When you bring up a new file to edit, the Find command and Replace command's buffers are retained. This allows you to go through a series of related files and search for some particular item by using FS, or replace it with RSS. (See the Find and Replace commands in this chapter.)

Depending on the amount of memory used by the new file, the Copy buffer will contain all or most of what it held when you left the previous file.

Write

The Write command saves the file presently in memory to the pathname specified.

Selecting this option causes a further prompt to be displayed if a file with the same pathname already exists:

```
/MYDISK/PROGRAM.TEXT already exists. Delete before W(rite?-->
      Y(es to delete old file before starting new one.
      N(o to delete old file after new one is complete.
      <esc> to cancel W(rite.
```

The file in memory may now be saved under any pathname. You do not need to specify the .TEXT suffix; it will be supplied automatically. If you want to suppress the suffix, end the pathname with a period.

If you change your mind and wish to return directly to editing the file currently in memory, without saving it, just press the RETURN key instead of typing a pathname.

Type "Y" to continue. After your file has been saved on diskette, the Editor displays a message similar to this:

```
Writing to /MYDISK/PROGRAM.TEXT.....
Your file is 1984 bytes long.
```

After writing to the file, the system displays the Quit menu and gives you your choice of action.

You may also write to unblocked files, such as to .QUME or .SILENTYPE to print a copy of your file.

Save

When you choose this option, your new file will have the same name as the file that was most recently called by the Editor. If you are unsure of the name of your file, you can check on it by either typing SE or Q.

After you invoke the Save option, you will be asked if you want to purge your original file. For example, if you use the Get command to access MYDISK:MYFILE from the Filer, edit the file, quit the Editor and then save the updated text with the Editor's Save command, the following message will appear:


```
/MYDISK/PROGRAM.TEXT already exists.  Delete before S(ave?-->
          Y(es to delete old file before starting new one.

          N(o to delete old file after new one is complete.
          <esc> to cancel S(ave.
```

If you type Y the old file will be removed from the disk before the new file is written out. This may cause the new file to overwrite the old file. If you have no backup of the original file and it is a large file, it would be safer to type N. When you type N the old file will not be overwritten and only will be removed when the new file is successfully written to the disk. If there is not room to copy the new file before destroying the old one, or if an error occurs while writing to the disk, the message

```
ERROR:  Writing out the file.  Please press <spacebar> to continue
```

will appear. Pressing the spacebar will return you to the Quit menu.

Do not press RESET after you have given the system permission to purge your original file; doing so may destroy both the old and new versions of your file. Also remember that if a power failure occurs while writing to the file, the file being written to may be lost, as well. If you have room on the disk, it is always safer to use the N option.

After your file has been saved on diskette, the Editor displays a message similar to this:

```
Writing to /MYDISK/PROGRAM.TEXT
Your file is 1984 bytes long.
```

Editor Command Summary

Cursor Moves

right-arrow key	Moves repeat-factor spaces right.
left-arrow key	Moves repeat-factor spaces left.
up-arrow key	Moves repeat-factor lines up.
down-arrow key	Moves repeat-factor lines down.
spacebar	Moves repeat-factor spaces in set direction.
TAB key	Moves repeat-factor tab positions in set direction.
RETURN key	Moves to start of line that is repeat-factor lines away, in set direction.
P(age	Moves repeat-factor screens-full in set direction.

Repeat-Factor

An integer from Ø through 9999 typed before a move or command. If repeat-factor is / the move or command is repeated as many times as possible in the file.

Set Direction

<	,	-	Change set direction to backward
>	.	+	Change set direction to forward

Moving Commands

Jump: Jumps to file's Beginning, Point, or End, or to a previously-set Marker.

Find: Looks in the set direction for the number of Literal or Token occurrences of the <target> string set by the repeat-factor. Must be typed with delimiters. S means use the same string as before.

Text-Changing Commands

Insert: Inserts text. Use left-arrow key to backspace over insertion. CTRL-Q deletes back to the most recent RETURN character in the current insertion. CTRL-X acts like CTRL-Q except that it also deletes the RETURN character.

Delete: Deletes all text moved over by the cursor. Back up the cursor to recover deleted characters.

Zap: Deletes all text between the current cursor position and the point (at the start of the latest text found, replaced or inserted).

Copy: Copies a diskette file, or the copy buffer, or between two markers in the current file, to the file at the position of the cursor.

Exchange: Replaces the character under the cursor with the character typed. Each line must be done separately. Pressing the left-arrow key causes the original character to reappear. Pressing the right-arrow key allows you to copy over existing text.

Replace: Looks in the set direction for the next Literal or Token occurrence of <targ> string, and replaces it with <sub> string. Continues repeat-factor times. Both strings must be typed with delimiters. Verify option asks for permission to replace. S means use the same <targ> or the same <sub> string as before.

Formatting Commands

Adjust: Adjusts indentation of the line the cursor is on. Left arrow and right-arrow keys move the line left and right. Cursor up or down adjusts lines above or below by same amount.

Margin: Starting at the cursor position, realigns all text between two blank lines (one paragraph) to the margins which have been set.

Miscellaneous Commands

Set: Sets the Point or a Marker of the specified name at the current cursor position. Sets options in the Environment for Auto-indent, Filling, margins, default search mode, and Command characters.

Verify: Redisplays the screen with the cursor centered.

Quit: Leaves the Editor. You may Update the workfile, Save a current copy of the file being edited, Exit without updating, Return to the Editor, or Write to any diskette file.

A

File Formats

A
File Formats

Text Files

Diskette text files begin with a two-block (1Ø24-byte) header page, reserved for the sole use of the Editor. The Pascal I/O subsystem creates the header page each time a user program opens a TEXT file, and REWRITEs or RESETs the file with a name ending in .TEXT to ease editing of input and output data. The file-handler transfers the header page only during disk-to-disk transfers; all transfers to non-block devices, such as those to .CONSOLE or .QUME always omit the header page.

Following the initial header page, the text appears in 1Ø24-byte text pages (two blocks each) on the diskette, where a text page is defined:

[DLE] [indent] [text] [CR] [DLE] [indent] [text] [CR]...[nulls]

Each DLE (Data Link Escape) is followed by an indent-code, a byte containing the value 32+(number of spaces to indent). The nulls at the end of the page always follow a CR (return character) and fill any remainder of a 1Ø24-byte page (because the compiler wants integral numbers of lines on a page). The Data Link Escape and corresponding indentation code are optional. In a given text file, some lines will have the codes, and some will not.

Data Files

The system does not "know" anything about the internal format of data files. The formats for data files are up to the user and must be defined in the user's program.

ASCII Files

ASCII files have no internal format. Like data files, they are streams of bytes known to contain ASCII characters and therefore readable by the Editor. BASIC text files are listed as ASCII files in the Filer's Extended directory listing.

B

SOS and Apple II Pascal File Format and Pathname Compatibility

B

SOS & Apple II Pascal File Format & Pathname Compatibility

This appendix tells you how to use Apple II Pascal diskettes with the Apple III Pascal system. It also includes a discussion of the Filer commands that operate differently depending upon whether SOS or Apple II Pascal diskettes are being used, and Apple II Pascal pathname conventions.

A full explanation of the Compiler option allowing you to use your Apple III Pascal system to write programs to run on an Apple II Pascal system is included in the Apple III Pascal Programmer's Manual.

A SOS diskette is any diskette formatted by the formatter on the Apple III Utilities Diskette. An Apple II Pascal diskette is any diskette formatted by the Apple II Pascal formatter on an Apple II or an Apple III. If you are formatting Apple II Pascal diskettes using an Apple II, use the formatter program that comes with Apple II Pascal. If you want to use an Apple III to format Apple II Pascal diskettes, you should use the AIIFORMAT program supplied on PASCAL3.

You cannot boot Apple III Pascal with an Apple II Pascal boot diskette. The Apple III files SOS.KERNEL, SOS.DRIVER, SOS.INTERP, SYSTEM.PASCAL, and SYSTEM.MISCINFO may not be used on Apple II Pascal-format disks. Other system files may be used on disks of either system.

General Considerations

Most software written for the Apple II Pascal system can be run on the Apple III Pascal system after being recompiled. If you want to run an Apple II program on an Apple III Pascal system, you should keep in mind the following points:

Apple II Pascal disketts may be used freely in the Apple III Pascal system with the exception of system diskettes.

Apple III Pascal diskettes cannot be used on the Apple II.

Apple II programs may be run on the Apple III after being recompiled, possibly with some changes in specialized areas such as graphics.

Apple III codefiles may be executed directly on an Apple II. They will have improved memory use if they are either recompiled on an Apple II or recompiled on an Apple III with the compiler's A2 option set.

Make sure that you keep Apple II disk routines resident by setting option B of the Options menu before you use an Apple II Pascal diskette with Apple III Pascal.

Be careful when editing text files on an Apple II which were originally created on an Apple III. The Apple III Pascal Editor can handle text files which are too large to fit in the Apple II Pascal Editor's workspace. When this happens, you will fill the Apple II Editor's workspace with the first part of the file, and the rest of the file will be lost when you write it back to disk after editing. Setting the Apple II Pascal system's swapping option to ON will give you some additional room to reduce this problem somewhat.

One advantage of writing programs on SOS diskettes is that it enables you to use features of Apple III Pascal, such as subdirectories, that are not available when Apple II Pascal

diskettes are used. You should note, however, that files written on SOS diskettes use one more block of space than files written on Apple II Pascal diskettes.

The following table provides a general outline of the differences between the way files behave when SOS and Apple II Pascal diskettes are used.

	SOS	APPLE II PASCAL
file storage (See Filer command Krunch)	files are stored on whatever blocks are available on the diskette	files must be stored in contiguous blocks
directory structure (See Filer commands List direcory and Extended directory below)	supports subdirectories	does not support subdirectories
Bad blocks (See Filer command Examine described below)	Does not mark blocks reported as bad by the Bad-blocks Filer command	Uses Xamine command to mark blocks reported as bad by the Bad-blocks command
Disk space overhead	Uses "overhead" blocks for house-keeping	No extra blocks used per file

Table B-1. File Differences: SOS vs Apple II Pascal

The Apple II Pascal system is unable to read SOS files. Thus, you cannot use an Apple II to transfer a file from a SOS diskette to an Apple II Pascal diskette.

Note also that Apple II Pascal and Apple III Pascal have different rules regarding legal characters in local filenames. Apple II Pascal permits most characters in local filenames. Apple III Pascal stipulates that the only characters that may be contained in local filenames are letters, numbers, and periods and that local filenames may not begin with a period.

If you attempt to transfer an Apple II Pascal file whose name contains an "illegal" character to a SOS diskette, Apple III Pascal displays an error message. Apple III Pascal also will not let you edit such a file. Thus, before attempting to use an Apple II Pascal file that contains an illegal character for the Apple III SOS pathname, you must use an Apple II to change the file's name.

Naming Conventions

The naming convention used throughout this manual has been that of Apple III SOS. It is described in detail in the Apple III Owner's Guide.

Since both the SOS convention and the Apple II Pascal file naming convention are supported with Apple III Pascal, a description of the Apple II Pascal convention's differences is included here.

Pathnames in the Apple II Pascal convention provide for a volume directory name and a filename, with a colon (:) used as a separator character. For example,

```
/PASCAL2/SYSTEM.EDITOR
```

and

```
PASCAL2:SYSTEM.EDITOR
```

are specifications in both conventions for the same file.

The Apple II convention on the Apple III Pascal system is extended to support subdirectories. The SOS pathname

```
/FARMING/FRUIT/BERRIES
```

would be given in the Apple II convention as

```
FARMING:FRUIT/BERRIES
```

Where the slash character is used as in the SOS format to separate subdirectories and local file names. Volume names under the Apple II Pascal convention are given as the volume name followed by a colon, as in MYDISK: .

The Apple II Pascal use of the characters * and : are also supported as root directory names. Both of them may be used as substitute references to the built-in drive's root name, unless another volume has been defined by the Filer's Prefix command, in which case the colon (:) will refer to the newly defined prefix name.

Volume device numbers in the two naming conventions are as follows:

Device	SOS	Apple II
Console (echo)	.CONSOLE	#1: (or CONSOLE:)
Console (no echo)	.CONSOLE	#2: (or SYSTERM:)
Built-in drive	.D1	#4:
First external drive	.D2	#5:
Second " "	.D3	#9:
Third " "	.D4	#1Ø:
Printer	.PRINTER*	#6: (or PRINTER:)
Remote Input	.RS232	#7: (or REMIN:)
Remote Output	.RS232	#8: (or REMOUT:)

*Note: You can always refer to the system printer as #6: or PRINTER:, but its name in the SOS system may be .PRINTER, .QUME, or .SILENTYPE .

If you are making extensive use of Apple II Pascal format disks, and are using Apple II's extensively, see the Apple II Pascal Operating System Reference Manual for a fuller description of the Apple II Pascal file naming convention.

If you are making only occasional use of Apple II Pascal formatted disks, you can use the SOS filename convention, and system display all the time.

Apple II Pascal Filer Commands

Following is a description of Filer commands that treat Apple II Pascal-format or SOS-format diskettes differently, or else can be used only with Apple II Pascal-format diskettes.

List Directory, Extended Directory

The List directory and Extended directory displays differ depending upon whether the specified directory belongs to a SOS or an Apple II Pascal diskette. This difference is a result of the systems' methods of storing files. Apple III Pascal begins storing a file wherever it can find an unused block on the diskette. When a block is filled with its 512 bytes of information, the system finds another free block, perhaps on another track, and continues to record information there. Files stored on Apple II Pascal diskettes are always stored in contiguous blocks, so if there are too few contiguous blocks on a diskette on which to save a particular file, an error message is given and the file is not saved.

Here is an example of an Extended directory list for a Apple II Pascal diskette, PEARTRE .

```
/PEARTRE (AppleII)  Size  Modified  Loc  File type  Eof
  PARTRIDGE           16   4-May-81    6  Datafile   512
  TURTLE.DOVE         1Ø   4-May-81   22  Datafile   512
  < Unused >          1Ø              32
  CALLINGBIRD          6  22-Jun-81   42  Datafile   512
  FRENCH.HEN           8  14-Jun-81   48  Datafile   512
  GEESE.TEXT           4  18-Apr-81   56  Textfile   512
  MAIDS.MILK          18   8-Jul-81   68  Datafile   512
  < Unused >           4              78
  DUCKS.TEXT           4  17-Jul-81   82  Textfile   512
  SWANS.CODE           6  17-Jul-81   86  Codefile   512
  < Unused >         198              92
  8 files listed, 212 blocks available, 198 in largest
```

The empty or unused blocks scattered throughout the diskette are marked as < Unused >.

The information contained on the List directory and Extended directory lists of Apple II Pascal diskettes includes, from left to right, the file name, file block length, last modification date, starting block address, file type and number of blocks used in the last block of each file.

The last line on the directory listing tells the number of files contained in the directory, the total number of free blocks available on the diskette, and the number of blocks in the largest unused area of the diskette.

The word AppleII appears after the volume name, thus indicating that the diskette is an Apple II Pascal diskette.

Note that because Apple II Pascal diskettes cannot contain subdirectories, the List directory and Extended directory command displays are the same whenever they refer to Apple II Pascal diskettes.

You can either use SOS- or Apple II Pascal-format filenames when referring to files on Apple II Pascal diskettes. In the above example Option C had been set to display SOS-format filenames; thus, files were displayed using the SOS format.

Krunch

The Krunch command consolidates all unused space on an Apple II diskette, and is used by typing K from the Filer.

Before using the Krunch command, you may want to use the List directory or Extended directory command to see the unused areas of the diskette that need to be consolidated.

The Krunch command requires that you type a diskette volume name, block-device name, or device number. The specified diskette volume must be on-line (currently available to the system). If you suspect that your diskette is damaged, you may want to perform a bad block scan of the volume before using the Krunch command. This will prevent your writing files over bad areas of the diskette. If bad blocks are found, they should be marked using the Examine command described later in this section.

Do not touch the disk, the power switch or the disk-drive door until Krunch tells you it has completed its task. If you do, you may make the information on your diskette unreadable.

EXAMPLE:

Suppose you wish to crunch diskette TOYBEAR .

Prompt: Crunch what vol ?

Response: /TOYBEAR

Prompt: From end of disk, block 28Ø ? (Y/N)

Typing the response Y initiates the normal Krunch. Typing an N will cause the prompt:

Prompt: Starting at block # ?

If you type a block number in response to this prompt, the Filer attempts to make room for new files in the area surrounding the block number that you specified. It does this by moving files which are below the specified block forward (toward lower block numbers), and moving files which are above the specified block backward (toward higher block numbers). This feature allows you to re-arrange files by placing them at diskette locations other than the end of the diskette.

Note: If you specify a Krunch starting block that is within an existing file, the Filer assumes that you want to move files away from the block immediately preceeding that file.

Make

The Make command creates a diskette directory entry with the specified pathname. Because Apple II Pascal does not support subdirectories, you cannot use Make with Apple II Pascal diskettes to create subdirectories.

Regardless of the type of diskette used, the Make command permits you to specify the number of blocks you want the file being created to occupy. When SOS diskettes are used, if the file being created is not a subdirectory, and no file size is specified, the new file is allocated zero logical blocks. When Apple II-formatted diskettes are used, the following file size specifications can be used:

[Ø] - Equivalent to omitting the size specification. The file is created using all of the largest unused area.

[*] - The file is created using all of the second largest area, or half of the largest area, whichever is larger.

If you know enough information about the location of a file before it was removed, and if nothing has been written over that area of the diskette since the removal, you can use the Make command to recover a removed file.

When you make a file, you create a diskette directory entry, without in any way changing the actual information stored on the portion of the diskette to which that directory entry refers. If you forget that the file is a "dummy" file, you can attempt to read into the Editor whatever information may have been stored on the diskette in that location.

Suppose, for example, that you have just used the Remove command to eliminate a 19 block file, which started at block 134. An Extended directory list of the diskette may show the "hole" where that file used to be, as a 19 block <unused> area starting at block 134. If you now use the Make command to create a file (of any name) that exactly occupies the blocks the removed file occupied, the new file will contain exactly the same information the removed file contained.

Because files on Apple II Pascal diskettes are stored in contiguous blocks and files on SOS diskettes are not, this method of retrieving files is far more likely to be successful with Apple II Pascal diskettes than with SOS diskettes.

Examine

The Examine command is invoked by typing an X from the Filer, and attempts to "fix" suspected bad blocks on a Apple II Pascal diskette. (Bad blocks are found by using the Bad-blocks command.)

This command requires that you type a diskette volume name, block-device name, or device number. The specified diskette volume must be on-line (currently available to the system) and one using the Apple II Pascal format.

EXAMPLE:

Suppose you have just done a Bad-blks scan of the diskette named MYDISK, and the Filer has given you the following message:

```
Scan for 28Ø blocks ? (Y/N) Y
Block 23 is bad
Block 24 is bad
Block 25 is bad
3 bad blocks
File(s) endangered:
THISFILE.TEXT       18    24
THATFILE.CODE       25    29
```

Now you have typed X to initiate the Examine command.

Prompt: Examine blocks on what volume ?

Response: /MYDISK

Prompt: Block-range ?

At this point, you should enter the block number returned by the bad block scan. If more than one bad block was reported, type the number of the first bad block, followed by a minus sign, followed by the number of the last bad block.

Response: 23-25

If any files are stored on the area of the diskette occupied by the blocks you are about to examine, you will be told the name of each such file and its beginning and ending block numbers:

```
Prompt:   File(s) endangered:
          THISFILE.TEXT 18 24
          THATFILE.CODE 25 29
          Try to fix them ?
```

Note: The files shown are endangered merely by containing bad blocks, NOT by the Examine process. Also, the question "Try to fix them ?" refers to the specified bad blocks, not to the files.

An N response to this prompt returns you to the outer level of the Filer. If you type a Y in response to the above prompt, you will

cause the Filer to examine the blocks in the range you specified. The Filer will then usually return a message like this:

```
Block 23 may be ok
Block 24 may be ok
Block 25 may be ok
```

in which case the bad blocks have probably been fixed. The Examine process attempts to fix the block-sector track data on the disk as well as the data in the file. Usually the file data will be unrecoverable.

```
Block 23 may be ok
Block 24 is bad
Block 25 is bad
File(s) endangered:
THISFILE.TEXT       18    24
THATFILE.CODE       25    29
Mark bad blocks  ? (Files will be removed !) (Y/N)
```

in which case the Filer is offering you the option of marking the block(s) which it could not fix. If you type a Y response to this prompt, the Filer first removes all files containing those bad blocks that could not be fixed. It then creates a special file on the diskette, named BAD , which exactly covers the bad blocks (or more than one such file, if the bad blocks are not contiguous). This message then appears:

```
Bad blocks marked
```

and you are returned to the outer Filer level. On the diskette, there is now a new directory entry saying

```
BAD.ØØØ24.BAD
```

Blocks in a file marked .BAD will not be used to store any of your files, and will not be shifted during a Krunch. These dangerous areas of your diskette are thus rendered effectively harmless.

A block that has been "fixed" may still contain useless garbage. The message "May be ok" should be translated as "is probably physically ok". Fixing a block means that the information stored in the block is read into the computer, is stored again at the same spot on the diskette, and is then read again. If the same information is read from the block both times, that spot on the diskette is probably not physically damaged (some kinds of damage cause inconsistent recordings). In that event, the message "May be ok" is given. However, if the two readings are different, the block is declared bad and may be marked as such to protect you from using that spot on your diskette.

Since disk drives are mechanical devices, they tend to have a range of actual adjustment values. Sometimes a diskette recorded on one drive will be unreadable on another, or may be readable with only certain others. Before giving up completely, try reading your "bad" diskette on several drives.

C

Summaries

C
Summaries

All Levels

The following commands are available at all levels of the system:

CTRL-\	Break signal; does a warm boot.
CTRL-9	Stops program output to the screen or printer until the next CTRL-9, without stopping the program.
CTRL-8	Makes Control-codes visible on the display.
CTRL-7	Temporarily stops any program or process. On the next CTRL-7, the program continues.
CTRL-6	Flushes data or commands out of the type-ahead buffer.
CTRL-5	Toggles video display on or off. When the display is off, programs run faster since no time is used on updating the display. The next CTRL-5 switches the display back on.
CTRL-RESET	Does a cold boot.
Power off-on	Does a cold boot.

Command Level

1. The Command level is reached automatically each time the system is booted, RESET, or initialized. It is also reached when any program, including any part of the operating system, finishes executing.

2. Use the Command level options to select any of the main subdivisions of the language system.

These are the Command level options:

File	Deals with the disks and disk files.
Edit	Helps you create and change text files.
Compile	Converts Pascal program text into executable P-code.
Assemble	Converts 65Ø2 assembly text into 65Ø2 machine code.
Link	Combines external routines into a program.
Execute	Loads and runs a utility program or other P-code file.
Run	Executes the workfile, automatically compiling and linking first, if necessary.
User-restart	Re-executes the last program or option executed.
Initialize	Re-initializes the system.
Halt	Does a warm boot of the system.
Options	Lets you set aside space for graphics. Also lets you control whether displays will use SOS or Apple II Pascal filenaming conventions and whether the system will accept Apple II Pascal files.
Make exec	Used to create exec files.

The Filer

These are the Filer commands:

Volumes	Shows which devices and diskettes are connected to the system.
List-dir	Shows what files are in the top most layer of a a directory or subdirectory.
Ext-dir	Shows all files contained in a directory or subdirectory.
Transfer	Copies a file or entire diskette to another diskette or device. Source diskette must be in a drive to begin.
Make	Creates a subdirectory.
Change	Renames a file or diskette.
Remove	Erases a file from a diskette directory or subdirectory.
Krunch	Compacts an Apple II Pascal format diskette. (Not used with SOS diskettes.)
Zero	Removes directory or subdirectories from a disk.
Prefix	Sets the default directory or subdirectory name.
Date	Sets and reports the current date.
Quit	Leaves the Filer and returns to Command level. Be sure your main system diskette is in the built-in drive.
Get	Designates a file to be used as the next workfile.
Save	Saves the workfile on diskette.
New	Clears the workfile.
What	Tells the original name of the current workfile.
Bad-blks	Tests diskette information for correct recording.
Examine	Attempts to fix bad blocks on Apple II Pascal-format diskettes. (Not used with SOS diskettes.)
Alter	Used to change the write-protect state, date, or file type as displayed by the directory commands.

Editor

These are the Editor commands:

Jump	Moves cursor to file's Beginning, End, Point, or preset Marker.
Page	Moves cursor one page.
Find /x/	Moves cursor to next "x".
Insert	Inserts typed text at cursor.
Delete	Moving cursor erases text.
Zap	Erases all text from cursor to start of last Find, Replace, or Insert.
Copy	Inserts buffer (last insert or deletion) or marked text at cursor.
Exchange	Replaces character at cursor by typed character.
Replace /x//y/	Replaces next "x" by "y".
Adjust	Moves line at cursor right and left.
Margin	Formats all text between two blank lines (one paragraph).
Set	Places a Marker at cursor, or sets Environment options for Indent-auto, Filling, Point, margins, etc.
Verify	Redisplays screen.
Quit	Leaves the Editor. You may Update the workfile, Exit without updating, Write to any diskette file before returning to Command level or Save to your original file.

D

Notes and Tables

D
Notes and Tables

Apple III Pascal Device Number Assignments

Apple III Pascal assigns a device number to every device driver configured into the system. When using Apple II Pascal format filenames, you can refer to a device by its device number. More complete information about device drivers is included in Apple III Standard Device Drivers.

SOS, the operating system used by Apple III Pascal, uses a set of special programs called device drivers, stored in the file SOS.DRIVER , to communicate with all peripheral devices. Each time you add, remove, or change a device driver, you must change the system configuration information in the SOS.DRIVER file by using the System Configuration Program contained on the Apple III Utilities diskette.

Each time Apple III Pascal is booted, the system examines the SOS.DRIVER file and assigns each driver contained in it either a standard or a user device number. Standard device numbers range from 1 to 12; user device numbers from 128 to 143.

Device number assignment is a two-pass process:

In the first pass, the system attempts to match each device driver with a SOS standard device driver name. If a match is made, the device driver is assigned the standard device number associated with the appropriate device driver name. Table D-1 lists SOS standard device driver names and their associated Apple III Pascal standard device numbers.

Some devices may be unassigned after the first pass. In the second pass, the system first checks an unassigned driver's device type value. (The device type value is part of the system configuration included in the SOS.DRIVER file.) If the device type value is one of the values listed on Table D-2 and the standard device associated with that value is unassigned, the system assigns that device to that standard device number. User device numbers begin at 128 and are assigned consecutively.

A device whose device type value is not one of the values listed in Table D-2 is automatically assigned a user device number. Any remaining devices are consecutively assigned the remaining user device numbers.

Here's an example. Suppose that the System Configuration Program indicates that the following drivers have been added to your system:

1. .CONSOLE
2. .PRINTER
3. .SILENTYPE
4. .AUDIO

When making its first pass, the system finds two device drivers that have SOS standard device driver names: .CONSOLE and .PRINTER . Based on the information in Table D-1, .CONSOLE is assigned Apple III Pascal standard device numbers 1 and 2 and .PRINTER is subsequently assigned Apple III Pascal standard device number 6 .

The first device that is encountered in the second pass is .SILENTYPE . This device was not assigned a device number during the first pass because there is no standard device named .SILENTYPE . In the second pass, the system checks the standard device type value of .SILENTYPE and finds that it is 41. Next it checks to see if a device has already been assigned the associated standard device number (6). Since a device already has been assigned this number, the system must instead assign .SILENTYPE a user device number. Because no user device numbers have been assigned, .SILENTYPE is assigned user device number 128. If standard device number 6 had been unassigned, SILENTYPE would have been assigned that number instead.

Note that SILENTYPE can always be referred to by its device name, .SILENTYPE, regardless of whether it is assigned a standard device number or a user device number.

Finally the system encounters the AUDIO driver. The driver remains unassigned during the first pass because there is no SOS standard device named .AUDIO . In the second pass the system checks to see if the device has one of the device type values listed on Table D-2. Because it does not, the AUDIO driver gets assigned the next available user device number which, in this case, is 129.

If the system encounters two devices with the same standard device type and the appropriate standard device number is unassigned after the first pass, Pass 2 will assign the first driver it encounters to the appropriate standard device number and subsequent drivers to available user device numbers. Devices are encountered in the order in which they appear in the System Configuration Program menu.

SOS Standard Device Driver Name	Standard Device Number
.CONSOLE	1
.CONSOLE	2
.GRAFIX	3
.D1	4
.D2	5
.PRINTER	6
.RS232	7
.RS232	8
.D3	9
.D4	1Ø
.D5	11
.D6	12

Table D-1. SOS Standard Device Names and the Associated Apple III Pascal Standard Device Numbers

Device Type Value (Hex)	Kind of Device Associated With Device Type Value	Standard Device Number
61	console	1 and 2
62	graphics	3
41	printer	6
C1, E1	disk	lowest available among 4,5,9,1Ø,11,12

Table D-2. Device Type Values, Their Associated Devices, and Apple III Pascal Standard Device Numbers

When to Use .TEXT and .CODE

A filename normally ends with a suffix that tells the system what type of data is stored in the file. The most common suffixes are .TEXT , for files that contain natural-language text, Pascal program text, or 65Ø2 assembly-language text, and .CODE , for files containing compiled P-code or assembled 65Ø2 machine code.

Many commands require you to specify a file by typing its filename. When the file being acted on can be of only one type (.TEXT , .CODE , etc.), the system allows you to type the filename either with or without the suffix: If you omit the suffix, the system will add it for you. If you want to override this feature, type a period at the end of the filename. When a command may apply to files of more than one file type, the following rules apply:

1. All Filer commands except Get and Save must have complete filenames, including the .TEXT and .CODE suffix, and use the filename exactly as you type it, without adding any suffix

2. The Get and Save commands automatically supply the correct suffix (.TEXT or .CODE) to the part of the workfile acted. Therefore, when using these commands, you must not specify a suffix.

3. The Librarian does not specify a suffix for its output file. When you specify one of these files, you must type the suffix.

Apple III Pascal System Diskettes

Definitions

The first stage of a bootstrap operation is the loading of the operating system (files SOS.KERNEL and SOS.DRIVER) and the P-code interpreter (SOS.INTERP). The second stage loads the Pascal command processor (SYSTEM.PASCAL and SYSTEM.MISCINFO).

Any diskette used in a bootstrap process may be called a boot diskette. A diskette containing the Pascal command processor is normally kept in the built-in drive during Pascal system operations and is called the main system diskette.

A cold boot occurs when you press CONTROL-RESET and operates as if you had just turned on the power. The second-stage boot, sometimes called a luke-warm boot, is the same as what happens when you invoke the Halt command. The effect of the Init command (warm boot) is to initialize system variables and to reload SYSTEM.MISCINFO. The effects of these three commands are summarized in the following table.

Command	Files Reloaded
Init	SYSTEM.MISCINFO
Halt	SYSTEM.PASCAL SOS.MISCINFO
CONTROL-RESET	SYSTEM.PASCAL SOS.INTERP SOS.KERNEL SOS.DRIVER SOS. MISCINFO

Table D-3. SOS/Pascal Boot Levels

Pascal System Diskettes

/PASCAL1 is both the boot diskette and the main system diskette for running Pascal applications programs. This diskette contains all of the system files necessary for a bootstrap load of the Pascal system. Since there are very few blocks available on this diskette for the application code and for data, a diskette in an external drive must be used if more diskette space is needed.

/PASCAL2 is the program development diskette.

/PASCAL3 is the diskette containing the Apple II-diskette formatter utility, Library, Libmap, and Setup files.

In general, it is easiest to cold-boot the system with diskette /PASCAL1 in the built-in drive, since it contains both the SOS files and the Pascal interpreter. For program development, see the recommended diskette configurations given in the Apple III Program Preparation Tools manual.

Making a Turnkey Diskette

The Apple Pascal system allows you to set up a turnkey system. This means the Apple will automatically begin running a particular program when you insert the turnkey diskette and turn the Apple on.

To set up a turnkey system based on a Pascal program, first make a copy of diskette /PASCAL1 with the SOS Utilities diskette's copy utility and change the copy's name to something you will recognize. For example, you might name this diskette TURNKEY . Now Transfer a copy of your Pascal program codefile onto the turnkey diskette, giving this new copy of your program the filename SYSTEM.STARTUP . Make sure your turnkey diskette contains the following files:

```
SOS.KERNEL
SOS.DRIVER      (only drivers needed by SYSTEM.STARTUP)
SOS.INTERP
SYSTEM.PASCAL   (Apple II drivers resident if needed)
SYSTEM.MISCINFO
SYSTEM.LIBRARY   (containing needed intrinsics only)
SYSTEM.CHARSET   (if needed by your STARTUP program)
SYSTEM.STARTUP
```

To run your turnkey program, put the turnkey diskette in the built-in drive and start a cold boot by pressing CONTROL-RESET.

Soon, with no further action on your part, SYSTEM.STARTUP is executed. Thereafter, SYSTEM.STARTUP will be executed each time the system is re-booted, as long as this disk is in the built-in drive.

The System Diskette Files

The following tables list the files that are normally found on each of the system diskettes needed for program development on the Apple III. The order of the files on any diskette is unimportant. When most files are needed by the system, it is only necessary that the file be present on any diskette in any drive. For exceptions to this rule, see table, The System Diskette Files: By Command, below.

The System Diskette Files: By Diskette

The files making up the Apple III system reside on four diskettes. The following table lists the system files that are found on each diskette.

/PASCAL1	/PASCAL2	/PASCAL3
SOS.KERNEL	SYSTEM.EDITOR	AIIFORMAT.CODE
SOS.DRIVER	SYSTEM.SYNTAX	LIBRARY.CODE
SOS.INTERP	SYSTEM.COMPILER	LIBMAP.CODE
SYSTEM.PASCAL	SYSTEM.ASSMBLER	SETUP.CODE
SYSTEM.MISCINFO	OPCODES.65Ø2	
SYSTEM.LIBRARY	ERRORS.65Ø2	
SYSTEM.FILER	SYSTEM.LINKER	

Table D-4. Apple III Pascal System Diskette Files

Apple III Pascal System Console Configuration

At boot time, the Pascal system establishes the console configuration by setting the Apple III system console options as follows:

New Line -- Disabled -- Read requests do not terminate when they encounter the new-line character.

New-Line Character -- Carriage Return (ASCII $ØD).

Keyboard Mode -- Console

Buffer Size -- 128-byte typeahead buffer

Attention Event -- Generates an interrupt on receiving the character CONTROL-\ . The Pascal system uses this interrupt to stop execution of the currently executing program.

Any Key Event -- disabled

No Wait Input -- disabled. Read operation terminates only when specified number of characters have been read.

Screen Echo -- disabled. Pascal performs its own echoing.

Character Copy -- disabled. Inhibits character copy capability of right-arrow key.

Character Delete -- disabled. Console driver treats backspace as any other character. Pascal handles the character delete function implied by backspace.

Line Delete -- disabled. The line deletion function implied by CONTROL-X is ignored by the console. Pascal handles this function itself.

Escape Functions -- All escape functions of the console are disabled.

Text Options --

Auto Advance -- enabled. Console advances cursor one position to the right after displaying each character.

Auto Line Feed -- disabled. Console does not automatically line feed after a carriage return.

Auto Return -- enabled. Console moves the cursor to the beginning of the next line or end of the previous line whenever it is advanced past the right or left edge of the viewport.

Scroll -- enabled. Data in the viewport is scrolled whenever the cursor is moved past the bottom of the viewport.

A console "Restore Environment" call is made each time a program ends execution on the Apple III Pascal system.

The SETUP Utility

The Setup utility is provided to allow you to set certain system configuration options such as graphics space allocation, Apple II disk routine residency, file name display format, and some Editor key assignment.

You run the Setup utility by typing X from the command level, and when prompted

```
Execute what file?
```

respond by typing

```
/Pascal3/SETUP
```

The menu display that SETUP produces is almost exactly the same as that of the Option command. (See the description of the Option command in the chapter The Command Level in this manual.) Operation of SETUP is identical to the Option command with the exception noted below.

Option D of SETUP allows you to change Editor key assignments. When you press D, the display will read

```
Key Options: A, B, C, D, <esc>, Q(uit
```

Option	Character function	Current Value
A)	Accept key	Ø3 => <ctrl>-C
B)	Escape key	Ø27 => escape
C)	Delete Line key	Ø23 => <ctrl>-W
D)	End of File key	Ø33 => <ctrl>-C

If you should want to change any of these key assignments, press the appropriate key and then enter the new value that you want, from ØØØ to 127. If you enter a number greater than 127, it is evaluated Modulo 128, and assigns the result to that key.

After making your new assignment(s), or if you don't want to change anything, type Q to return to the main SETUP menu. When you have completed any changes there that you might want, type Q again to return to the Command level menu. If any changes to system configurations have been made by your choices, they are accomplished at this time.

As with the Option command, changes to graphics allocation cause the system to be rebooted. All other changes are saved in memory and SYSTEM.MISCINFO so that when you reboot the system, the configuration you have chosen will be the new system configuration.

E

ASCII Character Codes

E
ASCII Character Codes

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
Ø	ØØ	NUL	32	2Ø	SP	64	4Ø	@	96	6Ø	`
1	Ø1	SOH	33	21	!	65	41	A	97	61	a
2	Ø2	STX	34	22	"	66	42	B	98	62	b
3	Ø3	ETX	35	23	#	67	43	C	99	63	c
4	Ø4	EOT	36	24	$	68	44	D	1ØØ	64	d
5	Ø5	ENQ	37	25	%	69	45	E	1Ø1	65	e
6	Ø6	ACK	38	26	&	7Ø	46	F	1Ø2	66	f
7	Ø7	BEL	39	27	'	71	47	G	1Ø3	67	g
8	Ø8	BS	4Ø	28	(	72	48	H	1Ø4	68	h
9	Ø9	HT	41	29	)	73	49	I	1Ø5	69	i
1Ø	ØA	LF	42	2A	*	74	4A	J	1Ø6	6A	j
11	ØB	VT	43	2B	+	75	4B	K	1Ø7	6B	k
12	ØC	FF	44	2C	,	76	4C	L	1Ø8	6C	l
13	ØD	CR	45	2D	-	77	4D	M	1Ø9	6D	m
14	ØE	SO	46	2E	.	78	4E	N	11Ø	6E	n
15	ØF	SI	47	2F	/	79	4F	O	111	6F	o
16	1Ø	DLE	48	3Ø	Ø	8Ø	5Ø	P	112	7Ø	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	5Ø	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
2Ø	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	12Ø	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	9Ø	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	6Ø	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
3Ø	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Table E-1. ASCII Character Codes

F
Transporting Programs Between Apple III and Apple II Pascal Systems

F

Transporting Programs Between Apple III and Apple II Pascal Systems

This appendix tells you how to transport (or "port") Pascal programs between the Apple III and Apple II Pascal systems.

A full explanation of the Compiler option allowing you to use your Apple III Pascal system to write programs to run on an Apple II Pascal system is included in the Apple III Pascal Programmer's Manual.

General Considerations

Many Pascal programs developed by either the Apple III or Apple II Pascal systems are interchangeable. Their similarities and differences are described in the rest of this appendix.

Diskette Compatability

The Apple III Pascal system will happily accept any diskette from an Apple II Pascal system with the exception of system diskettes: You can't boot an Apple III with Apple II diskettes.

The reverse is not true. Apple III diskettes, which use the SOS directory and file structure, cannot be read by the Apple II Pascal system.

Disk read/write routines that handle the Apple II Pascal format are supplied with the Apple III Pascal system. If you are going to use Apple II Pascal diskettes on your Apple III system, you will find that disk access is much faster if you make sure, by setting Option B as described in chapter 2 of this manual, that the Apple II disk routines are resident in

memory. If the Apple II disk routines are not resident the system will operate noticeably slower.

Textfiles can be transported between systems with the restriction that some Apple III textfiles may be too big to fit in the Apple II Pascal Editor's workspace.

Program Compatability

Apple II Pascal programs may be run on the Apple III after they have been recompiled on the Apple III Pascal system. Some parts of some programs may need to be changed slightly to run on the Apple III as described later in this section.

Apple III programs may be run directly on the Apple II, with some exceptions, but they will run more efficiently if they are either recompiled on the Apple III using the compiler's "A2" option or recompiled on an Apple II.

The Apple III compiler's "A2" option causes the compiler to generate the same codefile that would be generated by the Apple II's compiler. An Apple III Pascal codefile has some additional information in it that does not presently affect execution on an Apple II system, but future versions of the Apple II Pascal system may not be so accomodating.

The syntax of the Apple III and Apple II versions of Pascal are almost identical except for the BYTESTREAM, WORDSTREAM, and OTHERWISE enhancements to Apple III Pascal. The compiled code from these enhancements will execute on an Apple II system (assuming that you have compiled programs using those constructs on an Apple III using the "A2" compiler option.

Common Factors to Consider

You will need to consider the factors described below on many programs transported between the Apple III and the Apple II Pascal systems:

1. Since your Apple III will generally have more memory available than an Apple II, not all programs that run on an Apple III can be guaranteed to run on an Apple II.

   If this problem appears, try recompiling the program on an Apple II or on your Apple III with the A2 option set for the compiler.

2. The Apple III PGRAF graphics unit is not supported by the Apple II. Graphics in an Apple III program will have to be converted to equivalent TURTLEGRAPHICS functions and recompiled. Remember that not all the functions of the Apple II TURTLEGRAPHICS unit are supported by Apple III Pascal.

3. Assembly-language procedures will not work the same on both machines. This is primarily due to the banked memory features of the Apple III. In particular, parameter passing should be closely checked for compatibility.

4. The TRANSCEND unit requires the REALMODES unit on the Apple III, but is not required on the Apple II. Note that the results of real arithmetic may be different, this is due to the increased accuracy of the Apple III real number routines. Added features of the REALMODES unit are not available on the Apple II.

5. Compiler options that work on the Apple III may have to be changed: The Apple II compiler will not allow options to be written out in full, unlike the Apple III compiler. For instance, {$RANGECHECK} as well as the Apple II's {$R} options may be used on the Apple III.

6. When you present more than one identifier in an option statement, the Apple III requires that the identifier be repeated. The Apple II compiler option {$R SEG1, SEG2} would have to be rewritten as {$R SEG1, R SEG2} for the Apple III compiler.

7. Conditional compile options are not available on the Apple II.

8. The Apple III CLOSE procedure options WPROTECT and UNPROTECT are not available on the Apple II.

9. Apple III Pascal programs are not limited to the 26 segment maximum of Apple II Pascal. Also, the program library concept is not supported by Apple II Pascal.

Occasional Program Transport Factors

Some programs that you want to transport between the Apple III and Apple II will require attention to the factors given below, but most don't.

1. The use of UNITREAD and UNITWRITE for disk access may have to be altered to conform to either Apple II- or SOS-format disks.

2. The MEMAVAIL function works differently on the Apple III than the same-named function on the Apple II Pascal system. This is because the Apple III Pascal system sets aside a part of memory as code space, while the Apple II Pascal system assumes that all unused space is available memory.

3. Device contentions can occur on the Apple III if two devices try to use the same hardware elements. Apple III driver documentation points out potential problem areas for specific drivers.

4. The console device works differently on the two machines. Programs using information from SYSTEM.MISCINFO (which is the same format for both machines) to format their display screens will have no problems. Programs depending on either an 8Ø- or 4Ø-column screen must be changed to accomodate the screen width of the target machine. Special characters and features of the Apple III console driver are not available to the Apple II.

5. Apple III programs that depend on the non-contiguous format of SOS-format disks may have to be modified to work on the Apple II.

6. Apple II programs that have been cleverly written to handle diskette free space collection ("krunching") can have that feature removed since it is not needed on the Apple III when using SOS-format diskettes.

Glossary

block: Unit of storage on a disk or diskette. A block contains 512 bytes. A standard Apple mini-floppy diskette can store 280 blocks.

block device: A device that stores and retrieves blocks of data. A disk drive is a block device.

byte: Eight bits of data. The main unit of information storage of the Apple III computer.

character device: A device that sends or receives a stream of single characters (as opposed to sending and receiving a block of characters at a time as a block device does). The console and printer are both character devices.

codefile: A file containing P-code, the compiled version of a Pascal program, or 6502 machine code, the assembled version of an assembly-language program.

device: A piece of hardware used for data input or output. A disk drive, video screen, and speaker are all commonly-used Apple III devices.

device driver: Software interface to a device that enables the Apple to communicate with that device. Before your Apple III can carry on such communication, you must use the System Configuration Program to configure the appropriate device driver into your system.

device file: The data stream being sent to or from a character device.

device filename: The name used to refer to a device file. A device filename always begins with a period. .PRINTER and .CONSOLE, for example, are legal device filenames.

device number: A number assigned by the Apple III Pascal system to each I/O device on the system. Used to avoid conflicts in I/O processing.

directory: A file containing information about the files stored on a disk. A directory includes the local filename of each file that is on the disk as well as each file's length, last modification date, and file type. Each time a file is created or modified, information about the file is recorded in the diskette directory. The Filer's Extended directory and List directory commands display the information in a disk directory.

disk: A generic term for mass storage devices using rotating magnetic storage media. This includes diskettes and fixed disk storage.

file: A stream of bytes. A file may contain text (information, such as documents or computer programs), compiled P-code, assembled 65Ø2 machine-code, or any other data that can be used by the Apple.

local filename: The name given to a file. Local filenames may be composed of up to 15 letters, numbers, and periods and may not begin with a number. Files other than device files may not begin with a period.

P-code: (Also known as Pseudo-code) The compiled form of a Pascal program. Pseudo code is a machine-independent intermediate code that is interpreted by a specific machine-dependent interpreter at execution time.

Pascal system disk: A diskette that contains the file SYSTEM.PASCAL . The same Pascal system disk should be in the built-in drive each time the system returns to the main Command level.

pathname: A series of local filenames, each beginning with a slash, and all joined together, that specifies the path you take from directory to subdirectory to gain access to a certain file. /FEE/FI/FO is an example of a legal Apple III Pascal pathname.

prefix: A shorthand specification for a series of directories and subdirectories. Prefixes are used to prevent your having

to specify a complete pathname each time you want to refer to a file that is part of a complex subdirectory structure. The prefix is set with the Filer's Prefix command.

pseudo-code: See P-code.

root directory: The highest-level directory on a diskette. A standard Apple III mini-floppy diskette may only have one root directory. A diskette's root directory name is the same as the diskette's volume name.

segment: 1. A unit of a Pascal program that can be swapped in or out of memory as required for operation. This allows larger applications to be run with limited available memory. 2. (biol.) The basic unit of organization of the phylum Annelida, which contains earthworms, leeches, etc.

Sophisticated Operating System: See SOS.

SOS: The operating system used by the Apple III.

subdirectory: A file containing a listing of files that have been grouped together because of similarity of function. A subdirectory includes the local filename of each file that is a part of the subdirectory as well as each file's length, last modification date, and file type. The Filer's Make command is used to create subdirectories. The Filer's List Directory command lets you examine a subdirectory.

textfile: A file containing human-readable text such as documents or source programs.

volume: A unit of storage on a block device. A block device may contain more than one volume, or it may contain different volumes at different times. The volume name of a standard 28Ø-block floppy disk is the name of the diskette's root directory.

workfile: A file that can be easily modified. A workfile may be identified by the local filename SYSTEM.WRK or it may be given its own local filename.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z