SOFTWARE MANUAL

AMOS USER'S GUIDE

DWM-00100-35

AMOS
user's guide

DWM-00100-35
Revision A01

This document reflects AMOS versions 4.3and later

NOTE: This printing of the manual contains the contents of Change Page Packet #1 for the AMOS User's Guide, (DWM-00100-57), which may be ordered separately from Alpha Micro.

Second Printing: October 15, 1979

ALPHA MICROSYSTEMS
17881 Sky Park North
Irvine, CA 92714

IMPORTANT NOTICE

There are some simple software installation procedures that must be followed to get AMOS up and running for the first time on your particular hardware. DO NOT attempt to start up your system until a System Operator has done this necessary initialization of the system software.

NOTE: Throughout this document we frequently refer to the manual "Introduction to AMOS." This manual is in the process of being written, and we ask for your patience until we are able to print and distribute it.

This document was created using the Alpha Micro text editor VUE and the Alpha Micro text formatter TXTFMT.

* NOTE TO THE SYSTEM OPERATOR:

If you are to be the System Operator (the person in charge of system management, adding user accounts, adding terminals to the system, etc.), you MUST read the documents "The System Initialization Command File" and "Setting Up the Line Printer Spooler" in the "System Operator's Information" section of the AM-100 documentation packet before you try to use the system.

PREFACE

This manual is aimed at the general user of the system. We do not assume that you have had any great prior experience with computers. However, we do assume that you are familiar with such common computer concepts as "files" (known to IBM users as "data sets"), "disks," "jobs," "text editors," and "commands."

This manual contains very little theory or discussion of the inner workings of the operating system; instead, it is a practical guide to system operation. If you are interested in a broader system view than this manual provides (e.g., discussions on files, the operating system structure, the programs available on the system, organization of files on the disk, jobs, memory partitions, etc.), refer to the manual "Introduction to AMOS." If you are new to computers, you will probably want to read "Introduction to AMOS" before beginning to use the AMOS system; in it you will find definitions for many of the terms we use in the "AMOS User's Guide."

If you are an experienced AMOS user, you will want to just skim the "AMOS User's Guide," and refer to the "AMOS System Commands Reference Sheets" for answers to your questions about specific commands.

We would also like to draw your attention to the document titled "A Guide to the AMOS Software Documentation Library" which lists all of the software documentation currently offered by Alpha Micro; it tells you what information each manual contains, and also indicates the kind of reader at which each manual is aimed. If you want information on a specific facet of system operation (such as using the macro-assembler, BASIC or one of the text editors), see "A Guide to the AMOS Software Documentation Library" to find out what manuals cover those topics.

Table of Contents

CHAPTER 1

INTRODUCTION TO THE MANUAL

If you have been a user on a large, timesharing main-frame system before, you will notice a remarkable kinship between these sophisticated systems and the Alpha Micro microcomputer. In fact, the distinction between main-frames, minicomputers and microcomputers has become increasingly blurred during the last few years. In many ways, your Alpha Micro computer is closer in philosophy of operation to a large-scale timesharing system than it is to the microcomputers in the personal computer field; yet it retains the cost advantage of a small system. Take a look at some of the features of the Alpha Micro Operating System (AMOS), most of which are new to microcomputers:

TIMESHARING

AMOS supports several users on the system at the same time, all running the same or different programs.

USER ACCOUNTS

All users are assigned an account and (optionally) a password. A user must present his or her account number and password to get onto the system-- this protects your system from unauthorized use.

MULTITASKING

The system can handle more than one task per user at the same time. For example, you are able to print one file at the same moment that you are editing another file.

MULTIPROGRAMMING

Unlike some timesharing systems, AMOS oversees multiple users at the same time who are running DIFFERENT programs. Every user has a fixed partition of memory in which to run his own task, so that all of the tasks of multiple users reside in memory at the same time.

COMMAND LANGUAGE

You can create a file which contains a sequence of AMOS commands, command options and special symbols. Every time you invoke that file (called a command file), AMOS reads and processes the commands in it. A special type of command file (called a DO file) allows you to pass arguments to the file. You can also create your own AMOS commands by writing assembly language programs.

1.1 MANUAL CONTENTS

To make it easier to access the information in this manual, we've divided it into two general areas: •

PART I - GETTING STARTED

The chapters in Part I will get you started using the system. You'll learn about turning on your machine, using the keyboard, logging into and out of the system, mounting disks and specifying files. This section also contains a brief system demonstration that walks you through some of the major system programs in just a few minutes.

PART II - THE AMOS SYSTEM COMMANDS

Part II is meant to be much more of a reference guide than is Part I; this section contains comprehensive discussions of many of the major system commands, along with information on command files, DO files and disk backup procedures.

1.2 CONVENTIONS USED IN THIS MANUAL

To make our examples concise and easy to understand, we've adopted a number of graphics conventions throughout our manuals:

PPN	A Project-programmer number; that is, the number that identifies a user's account (e.g., [100,2]). Also shown as [p,pn].
devn:	Specifies a physical device type and a logical unit of that device. Such devices are almost always disks. For example, in a two-drive floppy disk system, the device DSK0: indicates the first drive of that physical device.

filespec — A file specification. A filespec identifies a file, and usually has these four elements:

devn:filename.extension[p,pn]

default — Information assumed by the system when you omit necessary information. For example, if you do not specify the account in which a file is located, the system usually looks for the file in your own account; in this case, the default is your own account.

{} — Optional element of a command line. For example:

.DIR {/switch} ↵

tells you that the element "/switch" is an optional element on the DIR command line.

—— — Underlined characters indicate those characters that AMOS prints on your terminal display. For example, throughout this manual you will see that most examples begin with an underlined dot (.); the dot is the AMOS prompt symbol that it displays when it is ready for you to type a command. The characters in the examples in this manual that YOU are supposed to type are not underlined.

↵ — A carriage return symbol. The curly arrow indicates the place in your keyboard entry to type a RETURN (hit the RETURN key). For example: ".LOGOFF ↵" tells you "After an AMOS prompt, type LOGOFF and a RETURN."

^ — Indicates a control character. As you enter characters from the keyboard directly to AMOS, the system usually displays those characters on your terminal display-- if you type a Control-C, you see a ^C on your terminal. (For a discussion of Control-characters, refer to Section 3.3, Control-characters.)

PART I

GETTING STARTED

Even before we begin to discuss the major system commands, you will want to get started using the system so that you can develop a feeling for how the system works.

Part I, Getting Started, is a practical, how-to-do-it guide to turning the system on and off, logging into the system, communicating with AMOS and specifying files. We'll also walk through a quick, brief demonstration of system use.

Unless you have the chance to put what you learn into practice, reading manuals can be dry work. So, if you have the opportunity, sit down in front of the machine with this manual and try out the examples in Part I; by the time you start learning about the system commands, you'll already have a good feeling for how to use the system.

CHAPTER 2

TURNING THE SYSTEM ON AND OFF

The usual computer system consists of the computer itself, one or more terminals, mass storage devices and a printer. A terminal is the device you use for communicating with AMOS, and it has a keyboard and a display of some kind. If the terminal displays data on a video screen, we call it a CRT terminal (e.g., a SOROC terminal or a HAZELTINE). If the terminal displays data by printing it on paper, we call it a hard-copy terminal (e.g., a TELETYPE). The mass storage devices can be one or more floppy-disk drives or hard-disk drives.

NOTE: Check with your System Operator, or refer to the instructions included with your hardware for information on turning on and off the terminals and disk drives. This procedure varies between different brands of devices, as does the procedure for changing disk cartridges and floppy disks.

Before you begin to use your system, make sure that the System Operator has installed the system software so that it works with your system's specific hardware configuration. Also ask the System Operator to assign you an account on the system; that will be the account you will use from now on when logging into the system and when creating files.

2.1 TURNING ON THE SYSTEM

Turn on whatever terminals you are going to be using. If your terminal has a REMOTE/LOCAL switch, turn the switch toward REMOTE.

If you are going to be using a printer, turn it on. If it has a REMOTE/LOCAL switch, turn the switch toward REMOTE. If it has an ONLINE button or switch, turn it to ONLINE. (You may turn the terminals or printer on and off at any time before or after system start-up.)

* NOTE: As a general rule, remember that disk drives are the last component of the system to be powered up, and the first item to be shut off.

Now--

If you have a hard-disk based system (that is, the programs that comprise AMOS reside on a hard-disk drive):

1. Holding down the RESET button, turn on the power to your computer. (In most cases, the RESET button is on the computer's front panel.)

2. Turn on the power to the disk drives. If your disk has a removable cartridge, make sure that the cartridge in the drive is the one that you want to use; if it is not, you may change it at this time following the instructions supplied with your drive. (Also, see Section 2.3, Changing Disk Cartridges and Floppy Disks.)

   Now, perform the steps needed to get the drives up and ready; follow the instructions provided by the manufacturer of your disk drive. For example, in the case of the CDC Hawk Drive (the disk used with the AM-500 Hard Disk Controller) you must push the START/STOP button. When the light behind the READY button comes on, the disk is up and ready to use. This procedure usually takes a minute or so.

3. The system now brings itself up by transferring into memory copies from the System Disk of those programs that comprise the operating system.

4. If you are going to be using floppy-disk drives in addition to your hard-disk drives, turn them on now. Insert the floppy disks you are going to be using.

If you have a floppy-disk based system (i.e., your system software is on a floppy disk):

1. Holding down the RESET button, turn on the power to your computer. (On most systems, the RESET button is on the computer's front panel.)

2. Turn on the floppy-disk drive. Insert your System Disk into drive zero (the System Drive) and close the drive door. (The System Disk is the disk that contains the programs that comprise AMOS.) If you have more than one drive in your disk unit, you can go ahead and insert the floppy disks that you are going to use. (Refer to Section 2.3, Changing Disk Cartridges and Floppy Disks.)

2.2 GETTING THE OPERATING SYSTEM'S ATTENTION

As soon as you have performed the procedure above, the system should be up and running. To let you know that it is ready and eager to respond to your instructions, AMOS displays its prompt symbol, a dot, at the left side of your terminal display. If you do not see such a symbol on your terminal, type a Control-C; that is, hold down the CONTROL key on your keyboard (sometimes labeled CTRL) while you type a C. You should see this on your terminal:

```
^C
.
```

(The ^C is the system's way of repeating back to you the Control-C that you just typed.) A Control-C is the system interrupt command; in this case, it serves to get AMOS' attention.

If you still do not see the AMOS prompt, check to see that your terminal is on, and that the cables running between the computer and the terminal are firmly in place. If you still see no prompt after typing a Control-C, check with the System Operator.

If all is well (i.e., you see the AMOS prompt), try typing a few characters on the keyboard. As you type, the system displays the characters on your terminal display. (We will talk more about using the terminal keyboard in Chapter 3, Communicating with AMOS.) From this point you can begin to communicate with the computer. (If you see each character displayed twice on the screen as you type, check your terminal for a switch labled FULL/HALF or FULL DUPLEX/HALF DUPLEX; turn the switch to FULL DUPLEX.)

2.3 CHANGING DISK CARTRIDGES AND FLOPPY DISKS

To change disk cartridges or floppy disks, follow the instructions that accompanied your disk drives. When changing disks, you do not need to reset the computer. Also, DO NOT turn off the drives or the computer.

After changing a disk cartridge or floppy disk, you must always "mount" that new disk by using the MOUNT command. (AMOS always mounts the System Disk for you when you first turn on or reset the system.)

2.3.1 The MOUNT Command

Type MOUNT and the name of the device which contains the new disk. (Remember to include the colon.) Then hit the RETURN key on your terminal keyboard. For example:

```
.MOUNT AMS1: ↵
```

the command above mounts the floppy disk you have placed in drive AMS1:.

Using a MOUNT command is the only way you have of letting the system know when you change a disk. You MUST use MOUNT every time you change a disk; if you do not, it is quite probable that you will lose files on that new disk, since the system may write over data on the disk (being confused about which areas on the disk are free, and which are already being used by files).

If you incorrectly enter the name of the device you want to mount, you see the NONEXISTENT DEVICE message:

```
.MOUNT ASM1: ↵
NONEXISTENT DEVICE
```

Check your spelling and retype the command.

2.4 TURNING OFF THE SYSTEM

Before you turn off your computer or disk drives, make sure that you are at the AMOS command level (that is, that you see the AMOS prompt symbol on your terminal display, which indicates that you are talking to the operating system), and that AMOS is not in the process of transferring data between the computer and the disk. Turn off your terminals and printer.

If you have a hard-disk based system:

1. If your disk drive has a removable cartridge and you want to remove and store the cartridge, do so now. (Follow the instructions supplied by the manufacturer of your drive.)

2. Power down the hard-disk drive. Usually this procedure consists of more than just turning the device off. For example, on the Control Data Hawk drive, you must first release the START/STOP button; when the light behind that button goes off, you may then turn off the power to the drive. This takes 60 seconds. Refer to the instructions shipped with your drive for the exact power-down procedure for your disk drive.

3. Holding down the RESET button, turn off the power to your computer.

If you have a floppy-disk based system:

1. Remove and store the floppy disks that are in the drives.

2. Turn off the power to your floppy-disk drives.

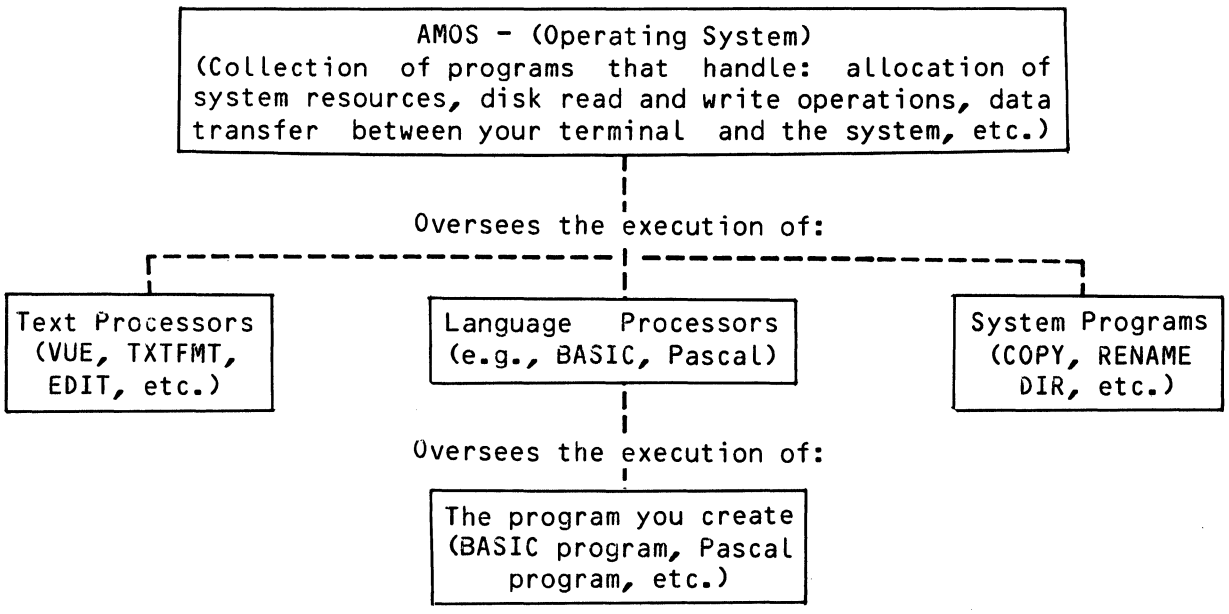
3. Turn off the power to your computer.

CHAPTER 3

COMMUNICATING WITH AMOS

The Alpha Micro Operating System (AMOS) is a collection of programs that serve as the system monitor. AMOS is the interface between you and the computer; it handles and schedules the many requests for system resources (CPU time, data transfer to and from the disks, etc.) that occur on a multiuser, multiprogramming system. Even when you are running programs on the system that are independent of AMOS (e.g., a BASIC program that you have created or the text formatting program, TXTFMT), AMOS supervises the execution of those programs.

To give you some idea of the hierarchy of the software components of your system, take a look at the rough diagram below. (If you want more information on the structure of AMOS, refer to the "System Operator's Information" section of the AM-100 documentation packet and the manual titled "Introduction to AMOS.")



3.1 PROMPT AND CURSOR SYMBOLS

When you see the AMOS prompt (a dot) on the left-hand side of your terminal display, you know that the system is up and running; you may now begin to enter instructions to AMOS. At this point we say that you are "at AMOS command level"; that is, that you are communicating directly with the operating system. At other times you may see other prompt symbols that indicate that you are communicating with a program that AMOS is executing. For example, when you use the text editing program, EDIT, you see the EDIT prompt symbol: *. At that point you must enter commands that EDIT understands. When you exit that program, you are again at AMOS command level (you see the AMOS prompt again). The various prompt symbols serve to remind you of the program with whom you are communicating. Remember: to enter AMOS commands, you must be at AMOS command level.

If you are using a CRT terminal, alongside of the prompt you also see a symbol called the cursor. The cursor may be a small white rectangle, a triangle, a blinking line, etc., depending on the type of terminal you have. The cursor always marks your current screen position. Any characters that you type appear at that position on the screen. As you type, the cursor moves as the new characters appear on the screen.

3.2 THE KEYBOARD

The first step in communicating with AMOS is to be able to type your instructions on the terminal keyboard. The keyboard is very similar to that of a standard typewriter, but you will find a few extra keys that have special functions. Take a moment to look at the keyboard of your terminal so that you can easily locate these keys later:

1. RETURN (sometimes labeled RET) - RETURN is the carriage return key. Just as you type a carriage return on a typewriter to begin a new line on the page, a RETURN tells the system that you are ending a line of input and that you want to begin a new line. AMOS will not process an instruction from you until you type a RETURN to let it know that you are finished with that line.

2. RUB (sometimes labeled DEL, DELETE, or RUBOUT) - RUB is the deletion key; it backspaces AND deletes. If you make a mistake while typing an instruction to AMOS, you can erase errors by using the RUB key. Hit the RUB key; if you are using a CRT terminal, you see the cursor move to the left, erasing the character in its new position. Keep typing RUB until the incorrect characters are gone; now type the correct characters. (If you are using a hard-copy terminal, rubouts usually appear as a backslash (\) followed by the characters you've deleted and another backslash. For example, deleting the word THE appears as: \EHT\.)

3. SHIFT - The shift key acts much as the shift key on a typewriter. By holding down the shift key you can type upper case letters and the symbols that occur on the upper half of the keys that bear two symbols.

4. ALPHA (sometimes labeled CAPS LOCK) - While the ALPHA key is locked in place, all letters appear as upper case; however, the keys that bear two symbols are not affected by the ALPHA key. That is, to type a %, you must hold down the shift key while you type a 5, even if the ALPHA key is down. (NOTE: Not all terminals have ALPHA keys.)

5. ESC (sometimes labeled ESCAPE or ALT MODE) - The Escape key is used with several programs on the system to signal the end of input or to switch between command modes. However, you do not use Escape at the AMOS command level.

6. CONTROL (sometimes labeled CTRL) - The Control key gives you the opportunity to enter a different kind of character to the system--a Control-character. (See below for a list of the important Control-characters that AMOS recognizes.) To type a Control-Character, hold down the CONTROL key while you type the appropriate letter. For example, to type a Control-C, hold down the CONTROL key while you type a C.

3.3 CONTROL-CHARACTERS

Control-C

The Control-C is the system interrupt command. Use it to interrupt whatever program is in progress, and return to the AMOS command level. After typing a Control-C to interrupt a program, you cannot resume execution of that program; you must start it over from the beginning.

You may usually interrupt programs even when they are displaying data. For example, if you ask the DIR command for a list of all the files in your account, you may interrupt it by typing a Control-C while it is listing your account directory.

Some programs, such as VUE and BASIC, do not recognize a Control-C as an exit command; instead you must use the exit command for that particular program if you want to return to AMOS command level. Other programs do recognize Control-Cs, but if an exit command exists for a program, it is usually a good idea to use that command to leave the program instead of typing a Control-C. Several programs perform various cleanup functions when you

use their exit commands; if you circumvent their procedures for an orderly exit by using a Control-C, the programs do not have a chance to perform those steps.

Control-U — While at AMOS command level, you may erase the characters from the left of the cursor to the left edge of your terminal display by typing a Control-U. (On CRT Terminals, the erased line of characters simply disappears, and the cursor moves to the left edge of the display; on a hard-copy terminal, you see a ^U, and the terminal readies itself for a new line of input.)

Control-I — A Control-I is a tab character. (Many terminals have a TAB key that you can hit instead of typing Control-I.) A tab moves the cursor to the next tab stop on your terminal display.

Control-S — If you have a CRT Terminal, it often happens that a program or command displays more data on your terminal than will fit on one screen-page. To stop a screen display, type a Control-S. You may then read the data on the screen at your leisure. Not only does the display freeze, but AMOS actually stops sending data to your terminal until you type a Control-Q (see below); at that point AMOS resumes sending information where it left off. While a Control-S is in effect, AMOS stores (but does not act upon) anything that you type except for a Control-Q.

Control-Q — When you type a Control-S to freeze the screen display (see above), you must type a Control-Q to resume screen display. If you have typed anything while the Control-S was in effect, a Control-Q tells AMOS that it can now go ahead and act upon that input. Try this out by typing a Control-S, a couple of commands (ending each with a RETURN); now type a Control-Q, and you will see all of the commands appear on the screen in quick succession.

3.4 TYPING COMMANDS

When you type an instruction to AMOS, you are typing an AMOS command. All valid AMOS commands either tell AMOS to perform some function for you (e.g., show you what files are in your account, erase certain files from the disk, etc.), or they tell AMOS to bring in and execute a program that is independent from AMOS (e.g., the command BASIC tells AMOS to bring in and execute the language processor BASIC). It is important to recognize those programs that are independent from AMOS, so that you remember with whom you

are communicating. For example, the BASIC command PRINT 3+3 is valid while you are communicating with BASIC (that is, while you are "inside" BASIC); but it is not a command that you can use when communicating directly with AMOS (i.e., while at AMOS command level). When in doubt, remember that the AMOS prompt indicates that you are at AMOS command level.

In later sections we'll talk in great detail about the form of specific commands; for now, though, here are some basics:

1. ENTERING COMMANDS - Although other programs on the system make a distinction between upper and lower case (e.g., BASIC), AMOS itself does not. Therefore, you may enter all commands in either upper or lower case, or a combination of both.

   For example, you may enter the directory command as:

   .DIR ↵

   or, as:

   .diR ↵

   You may also enter the names of files as upper or lower case or a combination of both. (NOTE: After you type them, AMOS converts all filenames and command names to upper case.) Remember that other entities on the system (for example, BASIC) do not accept lower case filenames.

2. COMMAND LENGTH - All AMOS commands and filenames are six characters or fewer.

3. COMMAND LINE - Some commands require that you specify files, arguments or other parameters. This group of characters that consists of the command itself and the additional information you supply with it, is called the command line. For example, the entire line below is a RENAME command line:

   .RENAME VAL.BAS=EVAL.BAS ↵

4. ENDING A COMMAND LINE - When you are at AMOS command level, you must ALWAYS hit RETURN after typing a complete command line; this lets AMOS know that you are finished with the line. AMOS ignores any command line that does not end with a RETURN. (NOTE: Although you will usually use the RETURN key, you may also end a command line by typing a Line Feed.)

5. COMMENTS - Any line at AMOS command level that begins with a semicolon (;) is a comment line, and is ignored by AMOS. You may also append a comment to the end of a command line:

```
.DIR [100,*]  ; Display directories for Project #100 ↵
```

This feature will be of more use to you when you begin to build command files.

6. TYPING MISTAKES - If you make a mistake while typing a command line, you may always correct any mistakes in that line if you have not yet typed a RETURN. Use the RUB key to erase single characters, or type a Control-U to erase the entire current line. (See the paragraphs above for an explanation of the RUB key and Control-U.) A Control-C tells AMOS to ignore the current line.

If you hit RETURN before correcting your mistake, and the command you entered was not a valid AMOS command, AMOS tells you that it did not recognize the command:

```
.PRIMT ↵
?PRIMT ?
.
```

You meant to say PRINT. After letting you know that it does not understand PRIMT, AMOS displays its prompt symbol. You are now free to try again.

3.5 THE HELP COMMAND

The purpose of the HELP command is to give information about the system and the system commands to the user new to AMOS. You do not have to be logged into the system to use HELP. To find out what topics the HELP command knows about, type HELP followed by a RETURN:

```
.HELP ↵
```

The screen clears and you see something like this:

```
Help is available for:

APPEND  BAUD    COPY    DEL     ERASE   LOAD
```

To see information about one of the topics listed, type HELP followed by the name of the topic:

```
.HELP COPY ↵
```

The screen clears and the terminal displays information on that topic. If you ask for information on a topic that HELP doesn't know about, you see:

```
I'm sorry I can't help you

Help is available for:

APPEND  BAUD    COPY    DEL     ERASE   LOAD
```

You may create your own HELP topics by using one of the system editors to create text files with .HLP extensions. (See Section 6.1.3, Extension, for a discussion of file extensions.) If you omit the account specification when you ask for a topic, the HELP command searches for the appropriate .HLP file in these accounts (in this order): 1. the System Help File Library (DSK0:[7,1]); 2. the library account for the project you are currently logged into (see Section 5.1, Project-programmer Numbers, for a discussion of library accounts); and, 3. the account you are currently logged into.

The HELP command automatically includes the .HLP files that you create in its list of topics, and displays your HELP files on command.

CHAPTER 4

A QUICK DEMO OF SYSTEM USE

Even before you learn all of the details about the AMOS system commands, we're going to take you through a brief, quick demonstration of the use of your machine. By no means is this chapter meant to be a comprehensive demonstration of all of the things you can do with the system; instead it is meant to introduce you to some of the major system commands. As we mention each command in this demonstration, we also tell you what manual to turn to if you want more information on using that portion of the system.

This chapter walks you through some simple procedures, but does not really tell you all of the details about how things work. Remember that the commands and programs we talk about in this section are all capable of performing a much greater variety of tasks than we attempt here. Turn to the appropriate documentation to find out what your system really can do.

This chapter will make much more sense (and will be of greater benefit) if you are able to sit down in front of an AMOS system and try the examples. If you have the opportunity, then, ask the System Operator for a few minutes of machine time and an account number that you can use.

Do NOT attempt to use the system if the initial system installation procedures have not been performed by the System Operator.

4.1 LOGGING IN

We assume that the system is up and running, and that you are sitting down in front of a terminal (probably a terminal with a video display). If you see a dot at the left hand of the screen, you are ready to go:

```
.
```

If not, type a Control-C (hold down the CONTROL key on the keyboard, and type a C). You should now see the AMOS prompt on the screen; if not, consult the System Operator for advice and go no further.

Once you see the AMOS prompt, you are ready to log into the system.

Type LOG, the number of the account you are going to use, and hit the RETURN key (the square brackets are optional):

```
.LOG [200,5] ↵
```

or

```
.LOG 200,5 ↵
```

NOTE: We assume that you are using an account on the System Disk (DSK0:). If this is not the case, type MOUNT, the name of the device you are using and a RETURN. For example:

```
.MOUNT DSK1: ↵
```

This makes sure that the device is mounted and ready for use. (If that device has already been mounted, using the MOUNT command again does no harm.) After you have mounted the disk, log into the account on that device by using the LOG command with the name of the device preceding the account number:

```
.LOG DSK1:[200,5] ↵
```

If the system requires a password for your account, you see:

```
.LOG [200,5] ↵
Password:    ↵
```

Enter the password that you received from the System Operator. Hit RETURN. You'll notice that the system does not show your password on the screen as you type it; this helps you to keep the password secret. Now you should see:

```
.LOG [200,5] ↵
Password:    ↵
Logged into DSK0:[200,5]
```

which tells you that you are logged into the system under account [200,5] on device DSK0:.

You can now begin to use the system.

** FOR MORE INFORMATION .... **

See this manual, Section 5.3, Logging into the System, and Section 2.3.1, The MOUNT Command.

4.2 CREATING A TEXT FILE WITH VUE

Now we will create a text file using the text editor, VUE. (NOTE: VUE is a screen-oriented text editor; that means that to use it you MUST use a CRT terminal. Check with the VUE manual to see the kinds of CRT terminals with which you can use VUE.) The file we'll create will be named MYFILE.TXT. Type:

```
.VUE MYFILE.TXT↵
```

VUE looks for a file named MYFILE.TXT. If there is such a file, VUE says:

```
AlphaVue Version 2.0
Loading MYFILE.TXT
```

and transfers a copy of the file from the disk into memory so that you can work on that text file. Since there is not as yet a file named MYFILE.TXT, VUE instead asks you:

```
AlphaVue Version 2.0
MYFILE.TXT DOES NOT EXIST - CREATE IT?
```

Answer yes by typing a Y and a RETURN. Now the screen clears and you are in VUE editing mode. You are no longer talking directly to the operating system; instead you are communicating with the text editing program, VUE. The commands you give to VUE are not the same kinds of commands that you can give to AMOS.

You see a screenful of asterisks, which indicates an empty text file. To place text into the file, just start typing. When you reach the end of the screen line, or when you want to start a new line, hit the RETURN key. Type a few lines of text; anything at all-- comments on the weather, an office memo, whatever you like.

4.2.1 Erasing Characters

To back up in the file and change things, you must move the cursor back to the point in your text that you want to change. Pick a word several lines back; we're going to erase it.

If you have a terminal with cursor control keys (four keys labeled with arrows), you may be able to use them to move the cursor around on the screen. (However, some terminals with cursor control keys transmit the wrong information to VUE. Experiment and see.)

Try moving the cursor by hitting the key labeled with an up-arrow.

(If your terminal does not have cursor control keys, or they are not active, type a Control-H to move backward, a Control-L to move forward, a Control-K to move up, and a Control-J to move down.)

When you reach the line containing the word you want to erase, move the cursor forward or backward until it is on the left-most character of the word you are going to erase. Type a Control-D. The first letter of the word disappears. You can type more Control-Ds to delete one character at a time, you can type a Control-V to remove all characters to the right of the cursor up to the next space, or you can type a Control-Y to remove all characters to the right of the cursor up to the end of the line.

4.2.2 Inserting Characters

You can now add a new word by just typing; however, this writes over the words already on the line. You can insert a word by typing a group of Control-Fs (which inserts spaces), and then typing the word over the spaces. Or, you can insert a word by typing a Control-Q (which tells VUE NOT to overwrite characters already on the line), and then typing the word you want to insert. When you finish typing the word, type a Control-Q again; otherwise, you will remain in this mode and will not be able to overwrite characters. (A Control-Q places you into character-insert mode if you are not already in it, or removes you from that mode if you are.)

4.2.3 Leaving VUE

When you are done experimenting, type an Escape to enter command mode. (You can re-enter editing mode by typing another Escape.) You see a line of text at the top of the screen that tells you what file you are editing, and various other items of information that will become important to you when you begin to use VUE regularly.

Below this header line is the VUE prompt symbol: >. The cursor waits at that position for you to type a command (or an Escape to return to editing mode).

When you become practiced at using VUE, you will learn about the commands you can use in command mode that allow you to do such things as replace every occurrence in your file of a particular word or phrase, format blocks of text, center lines, etc. For now, type an F and a RETURN; this tells VUE to F(inish). VUE now writes your text file to the disk. After a brief moment you are back at AMOS command level.

** FOR MORE INFORMATION .... **

> See the manual titled "VUE: Screen-Oriented Text Editor - Version 2.0."

4.3 FINDING OUT WHAT FILES ARE IN YOUR ACCOUNT

Now that you are again communicating with AMOS, you can see that the file you created is now in your account. To see a list of the files in your account (that is, to see your account directory), type DIR followed by a RETURN. (If there are a large number of files in the account, you will not be able to see them all listed on the screen at once. Try typing DIR/W followed by a RETURN; this formats the display into four columns on the screen.)

```
.DIR ↵
```

You see something like this:

```
MYFILE TXT  2            DSKÜ:[200,5]
MYFILE BAK  1
UNBILL BAS  6
Total of 3 files in 9 blocks
```

The information above tells us that we have three files in account DSKÜ:[200,5]: MYFILE.TXT, MYFILE.BAK and UNBILL.BAS. These files are respectively 2, 1 and 6 blocks in length. (We divide up the disk into units of 512 bytes called blocks.)

The display above is called a directory listing. The directory listing that you see when you use the DIR command will probably have more files in it than the example above, and the size of MYFILE will depend upon how much text you typed in when you created the file using VUE.

** FOR MORE INFORMATION .... **

> See this manual, Section 9.2, Finding Out What Files Are on the Disk (DIR).

4.4 DISPLAYING A FILE

To display the file that you've just created, type:

```
.TYPE MYFILE.TXT ↵
```

You see your file displayed on the screen. Your file may be so long that it does not fit all on one screen-page. If so, you can freeze the text display so you can see it by typing a Control-S; to release the display, type a Control-Q. To interrupt the display, type a Control-C.

** FOR MORE INFORMATION .... **

> See this manual, Section 10.1, Displaying the Contents of a File (TYPE)

4.5 COPYING A FILE

You can make a copy of your text file by using the COPY command. Pick a name from one to six characters in length. We'll use the name of NEW.TXT. Type:

```
.COPY NEW.TXT=MYFILE.TXT ↵
MYFILE.TXT to NEW.TXT
Total of 1 file transferred
```

If you now look at your file directory (using the DIR command), you see that you have a copy of your original file:

```
.DIR ↵
MYFILE TXT  2          DSK0:[200,5]
MYFILE BAK  1
UNBILL BAS  6
NEW    TXT  2
Total of 4 files in 11 blocks
```

You can use the TYPE command to assure yourself that you do indeed have an exact copy.

** FOR MORE INFORMATION .... **

See this manual, Section 9.5, Copying Files (COPY).

4.6 RENAMING A FILE

You can use the RENAME command to change the name of your file:

```
.RENAME DEMO.FX1=NEW.TXT ↵
NEW.TXT to DEMO.FX1
Total of 1 file renamed
```

If you use the DIR command again, you see that NEW.TXT is now named DEMO.FX1.

** FOR MORE INFORMATION .... **

See this manual, Section 9.3, Renaming Files (RENAME).

4.7 GETTING INTO BASIC

Now, leave AMOS command level by invoking the BASIC language processor. Type BASIC and a RETURN:

```
.BASIC ↵

READY
```

You have told AMOS to run the program that understands the computer language that we call BASIC. The BASIC language consists of simple, almost-English phrases that you can combine to form a BASIC program.

When you see the word "READY," you are communicating with the program that understands the computer language BASIC. We say that you are "in BASIC." You cannot use the AMOS commands here; instead you must use commands phrased in the BASIC language. To get used to communicating with the BASIC language processor, we'll enter a small BASIC program. Type the example below, entering all spaces and punctuation EXACTLY as shown. If you make a mistake while you are still on a line, you can back up and erase mistakes by using the RUB key. If you make a mistake after you have typed a RETURN, you can correct a line by simply retyping it; the LATEST version of a line will be the one that winds up in the program.

```
10 FOR I=1 TO 10 ↵
20 PRINT TAB(5*I),"WELCOME TO AMOS" ↵
30 NEXT ↵
```

To run your program, type RUN and a RETURN. You see this on the screen:

```
READY
10 FOR I=1 TO 10 ↵
20 PRINT TAB(5*I),"WELCOME TO AMOS" ↵
30 NEXT ↵
RUN ↵
COMPILE
COMPILE TIME WAS 0.10 SECONDS
               WELCOME TO AMOS
               WELCOME TO AMOS
                         WELCOME TO AMOS
                         WELCOME TO AMOS
                         WELCOME TO AMOS
                                   WELCOME TO AMOS
                                   WELCOME TO AMOS
                                   WELCOME TO AMOS
                                             WELCOME TO AMOS
                                             WELCOME TO AMOS
RUNTIME WAS 0.78 SECONDS

READY
```

4.7.1 Saving a BASIC Program

To save your program, type SAVE WELCOM.BAS:

```
SAVE WELCOM.BAS ↵

READY
```

Now type BYE to leave BASIC and return to AMOS command level.

```
BYE ↵
```

The next thing you see is the AMOS prompt (.) telling you that you are now back at AMOS command level. If you use the DIR command, you see that the BASIC program you saved is a file in your account:

```
.DIR ↵
MYFILE TXT  2          DSKU:[200,5]
MYFILE BAK  1
UNBILL BAS  6
DEMO   FX1  2
WELCOM BAS  1
Total of 5 files in 13 blocks
```

** FOR MORE INFORMATION .... **

See the manual titled "AlphaBASIC User's Manual."

4.8 ERASING A FILE FROM YOUR ACCOUNT

To erase the files that you have created, type ERASE, the name of the file (with extension) and a RETURN:

```
.ERASE DEMO.FX1 ↵
DEMO.FX1
Total of 1 file deleted, 2 disk blocks freed
```

Many of the AMOS commands recognize the wildcard symbol: *. (A wildcard is a symbol that matches any characters.) Let's use a wildcard to erase all of the BASIC programs in the account:

```
.ERASE *.BAS ↵
UNBILL.BAS
WELCOM.BAS
Total of 2 files deleted, 8 disk blocks freed
```

The command above tells the ERASE command to erase from your account all files that have a .BAS extension, regardless of their names.

** FOR MORE INFORMATION .... **

See this manual, Section 9.4, Erasing Files (ERASE).

4.9 LOGGING OFF THE SYSTEM

Now that you are done experimenting with the system, sign off the system by using the LOGOFF command:

```
.LOGOFF
User 200,5 logged off
```

You are now off the system. If you try to perform the commands you used above, you see:

```
[LOGIN PLEASE]
```

To get back on the system, use the LOG command again.

CHAPTER 5

IDENTIFYING YOURSELF TO AMOS

Once the system is turned on, and you have familiarized yourself with your equipment, you can begin to use the system. First, however, you must identify yourself to AMOS as a user of the system; this process of identification is called "logging into the system."

Every user of the system has one or more accounts. When you create files on a disk, the system marks those files as belonging to your account. No one (aside from the System Operator) can create files in your account except yourself and others within your own project (see Section 5.1, Project-programmer Numbers, below). Since AMOS maintains a list on each disk of the users who can create files on that disk, you must have an account on each disk, floppy disk or disk cartridge that you are going to use for storing files.

Each account has a directory (a kind of catalog) that lists all of the files in that account. You can use the DIR (Directory) command to see a display of your account directory.

5.1 PROJECT-PROGRAMMER NUMBERS

The System Operator assigns all user accounts. When you are assigned an account, you receive a project-programmer number (often called a PPN) and (optionally) a password. (NOTE: If you are using more than one disk, you will have an account on each disk. If you have only one account on each disk, they can all have the same PPN.)

Your PPN is a unique, two-part number that distinguishes you from all other users on the system. The first number is called the project number. If several users' PPNs have the same project number, those users are said to be in the same project. Users in the same project have certain privileges when it comes to transferring files between each others' accounts.

The second number is called the programmer number, and is separated from the project number by a comma. You will usually see PPNs enclosed within square

brackets. Here are a number of examples of project-programmer numbers with the brackets:

[110,5] [334,7] [250,12] [200,1] [100,100]

Both the project and the programmer number may range from 0 to 377, octal. (For a discussion of the octal numbering system, see the manual Introduction to AMOS; for now, the fact that the numbers are octal just means that no digit may be greater than 7.) A programmer number of zero usually indicates a library account for that project (e.g., the account [311,0] contains files of interest to all of the users in project 311).

Project numbers 1-77 are reserved by Alpha Micro for system software and the AlphaAccounting business package. Some of the special accounts set aside by Alpha Micro on your System Disk are:

[1,2]	Reserved for System Operator
[1,4]	System Program Library
[1,6]	Device Driver Library
[2,2]	Command File Library
[7,0]	Miscellaneous System Library
[7,1]	Help File Library
[7,2]	Mailbox Data File Library
[7,4]	LISP Language Library
[7,5]	Pascal Language Library
[7,6]	BASIC Language Library
[7,7]	MACRO Language Library

5.2 PASSWORDS

The System Operator may assign you an account password. The password is for your protection; if you maintain its secrecy, other users cannot access the system by using your project-programmer number to log into your account. If a password for an account exists, you don't need to enter it if you are transferring into that account from another account within the same project.

5.3 LOGGING INTO THE SYSTEM

There are very few commands that you can perform without being logged into the system; if you try to use the system without being logged in, you see the following message:

```
?Login please
```

To log into the system, use the LOG command followed by your PPN (the square brackets are optional):

```
.LOG [120,34]
```

If you have only one account with that PPN, LOG will search the available disks on the system (starting with the System Disk, DSK0:), and log you into the account on the proper device. If you have more than one account on different disks with that same PPN number, you must specify the device that contains the specific account you wish to work under:

```
.LOG DSK1:[350,6] ↵
```

The command above tells AMOS that you want to access account [350,6] on the logical unit DSK1:. (See Section 6.1.1, Device Name, for an explanation of how to specify devices.)

If you have a password associated with your account, LOG asks for it:

```
.LOG AMS1:[100,3] ↵
Password:  ↵
```

As you type your password, the system does not display it on your terminal. Remember that the purpose of the password is to keep unauthorized users from gaining access to the computer through your account; keep your password a secret.

If you have typed your PPN and password correctly, you will be logged into the system:

```
.LOG [321,10] ↵
Password:  ↵
Logged into DSK0:[321,10]
```

If you have not typed your PPN or password correctly, you see one of the messages:

```
?Account number invalid

?Bad password

?Command format error
```

and you will have to begin the login procedure all over again. If you have been unsuccessful in logging in (and have tried several times), check with the System Operator to make sure that your PPN and password are correct.

If you log into an account where another user is already logged in, you see:

```
.LOG [211,56] ↵
Logged into DSK0:[211,56]
Caution - other jobs same PPN
```

It is generally a bad idea for two users to be sharing the same account at the same time. For example, if one user erases some files while another user is accessing one of those files, or if both users try to access the same file at the same time, you will have serious problems.

Besides logging you into the system, the LOG command can perform several other functions which we discuss below.

5.3.1 Finding Out What Account You Are Logged into

If you cannot remember which account you are logged in under, type LOG and a RETURN:

```
.LOG ↵
Current login is DSK0:[321,10]
```

If you are not logged in, you see the message:

```
Not logged in
```

5.3.2 Transferring to Another Account

Once you are logged into the system, you can transfer to another account by using the LOG command with the PPN of the account to which you wish to tranfer:

```
.LOG AMS1:[100,3] ↵
Password:
Transferred from DSK0:[321,10] to AMS1:[100,3]
```

5.3.3 LOG and the Ersatz Devices

Aside from the usual kinds of devices (e.g., DSK1:, AMS1:, etc.), the LOG command also recognizes another type of device called an ersatz device. The ersatz devices provide a shorthand way of specifying accounts that you use frequently. The ersatz devices that LOG understands are:

SYS:	specifies	DSK0:[1,4]	System Program Library
DVR:	"	DSK0:[1,6]	Device Driver Library
CMD:	"	DSK0:[2,2]	Command File Library
HLP:	"	DSK0:[7,1]	Help File Library
BOX:	"	DSK0:[7,2]	Mailbox Data File Library
LSP:	"	DSK0:[7,4]	LISP Language Library
PAS:	"	DSK0:[7,5]	Pascal Language Library
BAS:	"	DSK0:[7,6]	BASIC Language Library
MAC:	"	DSK0:[7,7]	MACRO Language Library

To log into one of the accounts above, enter LOG and the appropriate ersatz device:

```
.LOG SYS: ↵
User logged into DSK0:[1,4]
```

LOG and several special commands (see Chapter 9, The Wildcard File Commands) are the only commands that recognize the ersatz devices.

5.3.4 The START Command File

Whenever you log into an account, LOG looks for a file in that account named START.CMD. If such a file is found, the system assumes that it is a command file, and begins executing it as such. This allows you to perform certain procedures automatically every time you log into an account (e.g., erase all backup files, create a directory listing for that account, etc.). If it is to be executed when you log into the account, the command file MUST be named START.CMD. We discuss command files in great detail-- what they are, how to build them and what kinds of system commands may be included in them-- in Chapter 8, Command Files and DO Files.

5.3.5 System Mail

When you first log into the system (but NOT when you use LOG to transfer between accounts), LOG checks the file in DSK0:[7,2] named MAIL.JNK. If such a file exists, the system prints the first line of the file on your terminal display. This is one way that information of interest to all users of the system can be made available to each user as he or she logs into the system:

```
.LOG [200,56] ↵
Check with Mr. Smith for info on changes in state tax tables.
```

You can use one of the system text editors to create and change the MAIL.JNK file.

5.4 LOGGING OFF THE SYSTEM

When you are through using the system, tell AMOS so by using the LOGOFF command:

```
.LOGOFF ↵
User 140,3 logged off
```

Whenever you leave your terminal for an extended period, it is a wise idea to log off; this prevents unauthorized users from sitting down at your terminal and using the files that your account can access.

CHAPTER 6

IDENTIFYING FILES TO AMOS

One of the things that you will be doing most often on the AMOS system is dealing with disk files: copying them, erasing them, creating them, etc. (For a discussion of files see the manual "Introduction to AMOS.") To be able to perform these procedures, you must be able to identify to AMOS those files with which you want to work; we call such an identification a "file specification," or "filespec."

6.1 FILE SPECIFICATIONS

A full filespec consists of several elements: 1. a device name (identifying where the file is to be found-- usually a disk of some type); 2. a filename (the name of the file); 3. an extension (a zero to three letter code that identifies the file type); and 4. a PPN (identifying the account in which the file is to be found).

A typical filespec might look something like this:

```
AMS0:INFO.TXT[234,12]
```

This tells us that the file named INFO is to be found in account [234,12] on unit number zero of a floppy-disk drive that uses AMS-format floppies. We also know that the file is a text file because it has the .TXT extension.

6.1.1 Device Name

Usually when we talk about a device, we're talking about a data storage device on which the system maintains your files. In almost all cases, such a device will be a disk; although you can use special units (such as your terminal, memory, etc.) as devices-- see below, Section 6.1.1.1, Special Devices. A device name tells AMOS where to find your file.

The device name identifies the logical unit on which a file may be found. The reason we talk about logical units instead of just the disk drive

itself, is that one physical device may contain several logical units (e.g., a single Control Data Hawk drive contains two logical units that you access separately-- a five-megabyte fixed disk and a five-megabyte removable disk cartridge. One Calcomp Trident 300-megabyte drive can contain 19 logical units-- DSK0:-DSK18:!).

A device name consists of three letters that identify the type of physical device being used (e.g., STD identifies a floppy-disk drive that handles IBM-format disks), and a number followed by a colon that identifies which logical unit is being used. So, for example, if you have two Control Data Hawk drives, the first physical device might contain logical units DSK0: and DSK1:, and the second physical device might contain logical units DSK2: and DSK3:. (Regardless of its type, the physical device that holds the programs that make up AMOS itself is always named DSK; and the logical unit that holds those programs-- called the System Disk-- is always named DSK0:.) Ask the System Operator for a list of the devices available on your system. Some common physical device types are:

HWK The Control Data Hawk hard-disk drive (Model 9427H), used with the AM-500 Hard-Disk Controller board.

AMS Alpha Micro-format floppy-disk drive, used with the AM-200 Floppy-Disk Controller board.

STD IBM-format floppy-disk drive, used with the AM-200.

TRI Calcomp Trident hard-disk system.

IMG 128-byte sector, non-AMOS structured disk.

6.1.1.1 Special Devices - Several special devices may be defined on your system that allow you to treat memory and your terminal as just another physical device. These special devices are:

MEM: Your memory partition (that is, the area of memory that you are using in which to run your job.) If MEM: has been defined as a device, you can use COPY to copy files to memory from the disk and vice versa; you can use DIR with MEM: to find out what modules are in your memory area.

TRM: Your terminal. You can use COPY to write files to that terminal as if it were a disk. This performs the same function as the TYPE command.

TRM:XX Terminal named XX; that is, you can refer to a specific terminal by supplying the name of that terminal. (Type TRMDEF and a RETURN to see the names of the terminals defined on the system.)

6.1.2 Filename

Every file on the disk has a name associated with it. Filenames may be from one to six characters in length. Although you may enter a filename as upper or lower case, or both, AMOS converts all filenames that you enter at AMOS command level to upper case. That is:

```
TRIDNT
```

is the same as:

```
Tridnt
```

or:

```
tridnt
```

(Programs independent from AMOS-- such as BASIC-- may distinguish between upper and lower case in filenames; check the manuals belonging to those programs for information on how they treat filenames.)

Filenames can only contain letters and numbers (e.g., FILE-2 is NOT a valid filename).

6.1.3 Extension

Following the filename (and separated from it by a dot) is the zero- to three-character extension. The purpose of the extension is to identify the type of file with which you are dealing. For example, a file with a .BAS extension is a BASIC program. The extension may be any letters or numbers that you want to assign to a file, but usually the extension is one of several extensions that AMOS recognizes:

AVRAGE.BAS	A BASIC source program produced by saving a program created inside BASIC, or by using one of the system text editors, VUE or EDIT.
AVRAGE.RUN	A compiled BASIC program; created by using the system command COMPIL on the source program (e.g., COMPIL AVRAGE.BAS), or by saving the program while within BASIC (e.g., SAVE AVRAGE.RUN).
CHANGE.SBR	A BASIC subroutine; an assembly language program that can be called by a BASIC program to perform commonly needed functions.
CUSTNM.DAT	A data file; that is, a text file created by a BASIC program that contains data that other BASIC programs can access (e.g., a list of customer names).

WORDS.TXT	A text file created by using the system text editors, EDIT or VUE.
WORDS.BAK	A backup text file created when you use EDIT or VUE, so that you always have on hand an earlier version of the text file you are currently working on (in case of problems during your current editing session).
WORDS.LST	A text file created by using TXTFMT to format a .TXT file, by using DIR to create a file containing a directory listing, or by using MACRO to generate a listing of an assembly language program.
SYSTEM.HLP	A text file that contains information about the system (i.e., a HELP file). Type HELP and a RETURN to see the list of files that you can ask for.
ERASBK.CMD	A command file is a text file that contains system commands. You can invoke all of the commands in the command file by simply typing the name of the file.
DIRFIX.DO	A DO file is a special kind of command file that allows you to specify text items to be substituted for parameter symbols in the DO file.
DEBUG.MAC	An assembly language source program created by using one of the system text editors.
DEBUG.OBJ	An assembly language object program created by using the system Macro-assembler, MACRO, on a .MAC program.
DEBUG.PRG	An assembled program produced by using the linkage editor, LINK, on an .OBJ file to create a machine language program.
DEBUG.SYM	A file that contains the symbol table for a .PRG file; created by using the program SYMBOL on a .OBJ file.
OPTIMZ.LSP	A program written in LISP.
TOPDWN.PAS	A program written in Pascal.

Remember: you can assign any extension to a file; even a null extension (e.g., FILEA.); the extensions above are often assigned to files by various programs on the system, and are the extensions that many programs use as defaults. For example, unless you specify otherwise, BASIC saves a program file with the extension .BAS; if you do not specify the extension when loading a program file into BASIC, BASIC assumes the default extension of .BAS.

6.1.4 Project-programmer Number

We have already discussed PPNs in an earlier chapter (Chapter 3, Communicating with AMOS). The PPN in a filespec identifies the account in which the file is to be found. If you are referring to a file in the account you are currently logged into, you may usually omit the PPN in the filespec. (But not always-- see Chapter 9, The Wildcard File Commands.)

6.2 WILDCARD SYMBOLS

The usual filespec selects one file (e.g., DSK1:POOL.LSP[344,1] refers to only one file-- POOL.LSP in account DSK1:[344,1]). Usually that's fine, but what if, for example, you want to erase ALL files in your account that have a .BAS extension? You could specify each file individually:

```
.ERASE SEED.BAS,RNDM.BAS,TRMCRV.BAS,SOS.BAS↵
```

It would be much easier if you could specify all of those files with one filespec. To allow you to do so, various AMOS commands recognize special symbols that we call wildcards. A wildcard can match any other symbol or group of symbols. For example, instead of typing the entire command line above, you can use one filespec to specify all files in your account that have a .BAS extension:

```
.ERASE *.BAS↵
```

The asterisk (*) can match any group of symbols. In the example above, files of ANY name with a .BAS extension match the filespec *.BAS. That means, of course, that the command above erases ALL .BAS files in your account.

NOTE: Not all AMOS commands recognize wildcards. You must refer to the documentation for a specific command to see if you can use wildcards in the filespecs you supply to that command.

The two most common wildcards are ? and *. All AMOS commands that recognize wildcards recognize these two symbols.

* Matches any symbol or group of symbols in a filename or extension. BOTANY.* selects all files in your account named BOTANY of ANY extension (e.g., BOTANY.TXT, BOTANY.LST, BOTANY.BAS).

  You may precede the * with one or more symbols (e.g., F1*.MA* selects the files F1TST.MAC, F1NEW.MAX, and F1OLD.MAX), but you may not follow the * with any characters (e.g., *DSK.TXT).

? Matches any one symbol in a filename or extension. ???DSK.MAC selects PACDSK.MAC, DIRDSK.MAC, and AR1DSK.MAC.

You may place characters before or after ?s (e.g., M???1.TXT selects MARY1.TXT and MTST1.TXT). If ?s appear at the end of a filename or extension (e.g., FR???.MAC or ACCNT.T??), that many or FEWER characters may match. For example, P????.RUN may match the files PINVT.RUN, PTST.RUN, and P1.RUN); otherwise, the number of matching characters exactly matches the number of ?s.

(NOTE: Several of the commands that you can use when handling files have even more advanced wildcarding abilities-- see Chapter 9, The Wildcard File Commands.)

6.3 FILE SPECIFICATION DEFAULTS

Under certain conditions, various portions of the system are smart enough to be able to fill in information that you have left out. For example, if you omit the device name and the PPN from a filespec, many of the AMOS commands will assume that you are referring to a file that appears in the account and on the device that you are currently logged into. Those commands fill in the "default" account and device specifications.

For example, if you are logged into account [100,5] on device DSK1: Most AMOS commands that read this filespec:

BIOPAK.BAS

search for the file on DSK1: in account [100,5]. In the case above, then, the default device and PPN specifications are the account and device you are currently logged into.

The defaults assumed by the system depend on what command you are using, but we can make some generalizations:

1. The default account specification is the PPN you are logged in under.

2. The default device name is the name of the device you are logged into.

3. The default device unit is zero (e.g., the system assumes that AMS: refers to AMS0:).

(NOTE: These generalizations do not apply to ALL commands on the system; the commands known as the wildcard file commands maintain their own set of defaults. For example, if you do not supply a device unit number to a wildcard file command, that command performs a wildcard search for the

specified file on ALL units of the device specified, not just on logical unit zero. Refer to Chapter 9, The Wildcard File Commands, for more information on the defaults used by these commands.)

Many programs on the system use specific default extensions. For example, the text formatting program, TXTFMT, uses the default extension of .TXT. That is, if you omit a file extension from a filespec that you give to TXTFMT, TXTFMT assumes that the file is a .TXT file. The point to remember here is that you may omit extensions from a filespec that you give to a command, as long as that extension is the default extension being used by the command. In other words, since the TXTFMT default extension is .TXT, you may omit the extension from a filespec for a .TXT file. If you want to use TXTFMT on a file that is NOT a .TXT file, you must include that file's extension in the filespec.

Remember that the defaults used by the system depend upon the command to which you are specifying the file, so look at the documentation for a specific command to see what defaults that command uses.

PART II

THE SYSTEM COMMANDS

We have introduced you to some of the system commands in Part I, Getting Started. Now we're going to discuss the major system commands in comprehensive detail. If at any time you are confused by a term or concept, refer to the Index for other references to the topic or to the manual "Introduction to AMOS."

The following chapters group the commands by function and type. If you are already familiar with the AMOS commands, you will probably want to just skim these chapters, and turn directly to the "AMOS System Commands Reference Sheets" for concise, alphabetically-ordered summaries of command operation.

Not every system command is covered by this manual. Those commands that have manuals of their own (e.g., BASIC, VUE, TXTFMT), those commands of interest primarily to the assembly language programmer and System Operator and those commands not of common interest to the general user of the system are discussed in the "AMOS System Commands Reference Sheets" and in other appropriate manuals.

CHAPTER 7

INTRODUCTION TO AMOS COMMANDS

Before we get into specifics on the AMOS system commands, we would like to provide a general overview on what an AMOS command is, what form it takes and how AMOS responds when you enter a command.

Until now we have talked about the commands that you enter at the AMOS command level as if they were grouped into two categories: 1. instructions that tell AMOS to run a program independent of itself (e.g., the BASIC command tells AMOS to run the language processor program, BASIC); and 2. instructions that tell AMOS to perform system functions (e.g., the DIR command tells AMOS to display a list of the files in your account).

It was convenient to make this distinction in our earlier discussions, but now is the time to clarify just what an AMOS command is-- actually, every command that you enter at AMOS command level is just the specification of a disk file or a disk file that has been loaded into memory (a memory module). AMOS responds to the command by trying to locate the memory module and executing it, or by finding the file on the disk, loading a copy of it into memory and executing it. Every AMOS level command specifies an assembly language program (a .PRG file) or a command file.

In other words, when you instruct AMOS to perform a system function (e.g., DIR), you are asking it to load into memory the DIR program and execute it. What this means is that since programs that perform system functions are not actually part of the operating system itself but are simply files on the disk, you can add to the commands that AMOS recognizes by just writing your own .PRG files or command files.

When you enter a command at AMOS command level, AMOS goes through several search procedures looking for the program or command file specified by the command. For example (unless you specify an extension and an account number), AMOS first looks for the file as a .PRG file in the System Program Library Account, DSK0:[1,4].

You will not usually have to worry about this process, but you may want to refer to Appendix B, AMOS Command Processing, if you find yourself in a situation where you have more than one .PRG, .CMD or .DO file of the same name in different accounts, and you want to know which file AMOS is going to find first if you enter just the filename.

If AMOS cannot find the program or command file specified by a command, you see your input repeated back to you enclosed in question marks:

```
?CREATE?

?VUEW  ?
```

Enter the command again, checking to make sure that your spelling is correct.

7.1 COMMAND SYNTAX

The syntax of a command is its proper form. At the front of each command description in the chapters that follow, you will see a line that says COMMAND SYNTAX. Beneath this is a line that illustrates the syntax to follow when you enter a command line containing that command. For example:

COMMAND SYNTAX:

```
.DIR {listfilespec=}{filespec1{....,filespecN}}{/switch{/switch}}
```

Below is a discussion of some of the symbols that you may see in such an example:

1. At the start of every command line you see the AMOS prompt (.); this indicates that you must enter every AMOS command while at AMOS command level.

2. The phrase "filespec" indicates a file specification (e.g., DSKU:CHNG.BAS[100,2]). "Filespec1" indicates the first file specification; "filespecN" means the "Nth" file specification. For example:

   ```
   .COMMAND filespec1,.....filespecN
   ```

   tells you that the command expects a list of filespecs, 1 through N.

3. The {} symbols indicate the optional elements of the command line. For example:

   ```
   .COMMAND {filespec}
   ```

   means that you do not have to specify a file when using the command.

You may sometimes see embedded optional elements. For example:

.COMMAND {/switch{/switch}} ↵

tells you that you may optionally specify a switch (a code that selects a command option) which may in turn optionally be followed by another switch.

4. The curly arrow symbol (↵) indicates a RETURN at the end of the command line.

The exact syntax that a command follows depends on the particular command; for specific information, refer to the documentation for that command. However, all of the commands that you can use at AMOS command level have certain things in common:

1. Commands are six characters or fewer in length.

2. You may enter commands in upper or lower case, or a combination of both.

3. All command lines must end with a RETURN or a line-feed, and most commands require that the command line fit on one screen line.

4. All file specifications that you enter on a command line follow the conventions we discussed in Chapter 6, Identifying a File to AMOS.

7.1.1 Command Defaults

Every command description in the following chapters includes a section titled COMMAND DEFAULTS which lists all of the file specification defaults used by that command. The set of defaults used by a command depends upon the command. For example, DIR assumes a default extension of .*, but ERASE assumes an empty extension as the default (e.g., INVCON.).

Refer to the documentation for a specific command for information on its defaults.

7.2 COMMAND SWITCHES

Many of the AMOS commands allow you to select among several command options by including one or more "switches" on the command line. The switch is either a one character code or a short word (e.g., /R or /QUERY) that tells the command which command options to put into effect. For example: the ERASE command usually does not ask you to confirm deletions; if you would like it to do so, you may include the QUERY switch on the ERASE command line:

```
.ERASE/QUERY *.BAK ↵
```

(Separate the switch from the rest of the command line by preceding it with a slash-- /.) The command line above tells ERASE to ask for confirmation before deleting each .BAK file from your account.

Command options available vary depending upon the specific command, as does the form that the command switches take. For example, some commands assume that each new switch starts with a slash (e.g., .COPY OLD.*=NEW.*/QUERY/NODELETE), and that switches may appear anywhere on a command line (e.g., .ERASE *.BAK/QUERY,*.TXT); other commands assume that EVERY single character after a slash to the end of the command line is a separate switch (e.g., .MAP/FSR). Check the documentation in this and other manuals to see how a specific command handles switches.

CHAPTER 8

COMMAND FILES AND DO FILES

There are many times when you find yourself entering the same sequence of commands over and over. For example, let's say that every time you log into your account you erase all backup files; then you look at the directory of your account:

```
.LOG [100,4] ↵
Logged into DSK0:[100,4]
.ERASE *.BAK ↵
MEMDOC.BAK
COMF.BAK
Total of 2 files deleted, 3 disk blocks freed
.DIR ↵
MEMDOC  TXT     12              DSK0:[100,4]
COMF    TXT     5
VAR     BAS     10
VAR     RUN     7
Total of 4 files in 34 blocks
```

The example above is a simple one; sometimes a frequently used sequence of commands may be quite long and tedious to type. An extremely powerful tool for dealing with this problem is the command file. A command file is a text file that contains the same kinds of input that you might enter from the keyboard. You can tell AMOS to read its instructions from a command file instead of having to enter those commands and data yourself.

Let's say that you create the command file LOGIN.CMD to perform the functions in the example above. The file contains the following lines of text:

```
:T
; Do Login clean-up functions
LOG [100,4]
ERASE *.BAK     ; Get rid of the backup files.
DIR
```

(If you do not include the :T at the front of your command files, you will not see the lines of text on your terminal display as AMOS processes them. See Section 8.1.1, Special Symbols in Command Files, for an explanation of this and other special symbols.)

At AMOS command level, enter the name of your command file:

.LOGIN ↵

AMOS now reads its instructions from the file LOGIN.CMD and performs the functions asked for.

If the extension of a command file is .CMD or .DO (denoting a special kind of command file called a DO file), you do not have to include the extension when entering the name of the command file; otherwise, you must specify the file extension. (That is, if your command file is DOIT.TXT, you must include the .TXT when specifying the file.)

To create a command file, use one of the text editor programs to make a text file (usually with the .CMD extension). Fill the file with the commands you would ordinarily enter from the keyboard. You may include comments within your file by preceding them with a semicolon (;). AMOS does not process the comments. It does display comments on the terminal display as it processes the command file if a :T symbol is at the front of the file. It is a good idea to liberally comment your command files so that you remember exactly what functions the file performs.

* IMPORTANT NOTE:

> Although you may type commands in either upper or lower case when entering them directly from the keyboard, the commands in your command file MUST be in upper case. Comments or messages may be in either upper or lower case.

[Of special interest to the System Operator is a unique command file called the system initialization command file (SYSTEM.INI), that the system uses every time you turn on or reset the computer. This command file has special properties and commands that help AMOS to tailor the system software for your particular hardware system. The System Operator can find information on the SYSTEM.INI in the section titled "System Operator's Information" in the AM-100 documentation packet.]

8.1 THE CONTENTS OF A COMMAND FILE

A command file can contain any commands or data that you might enter from the keyboard; the file can even contain the name of another command file.

AMOS continues to read lines of text from the command file until the end of the file. You may run and exit programs, supply data to programs or perform system functions, all under the control of a single command file.

As an example, let's say that you have a BASIC program that you want to edit, compile and then test. Suppose that you also want to delete the backup file created by the editor. You might want to create a small command file to perform these functions:

```
:T
VUE AVRAGE.BAS
COMPIL AVRAGE
RUN AVRAGE
3
12
7
ERASE *.BAK
```

The first line of the command file (:T) ensures that you see the lines of text in the command file as AMOS processes them. The next line (VUE AVRAGE.BAS) invokes the text editor, VUE, and tells VUE that you want to edit the BASIC source program named AVRAGE. After you make some changes to the program, you leave VUE by using the VUE exit command. Now AMOS again begins to read its instructions from your command file.

You are again at AMOS command level, and the next line of the command file tells AMOS to bring in the BASIC compiler and compile the BASIC program. The compiler produces a new file named AVRAGE.RUN (the compiled version of your original source program). Line 4 of the command file tells AMOS to execute AVRAGE.RUN.

The next three lines provide test data for the program, which asks for three numbers. When the BASIC program finishes, you are back at AMOS command level.

The final command in the command file tells AMOS to erase all backup files in your account.

8.1.1 Special Symbols in Command Files

Any command or data that you can enter from the keyboard is a legal element of a command file. In addition to these elements, there are several special symbols that appear only in command files (e.g., the :T symbol mentioned above), and that are never seen or processed by programs other than AMOS. If you use these symbols, they MUST appear at the very beginning of a command file line.

:T TRACE - To see the command file lines on your terminal screen as AMOS processes them, you must have a :T at the top of your command file. (Sometimes it's useful to NOT be able to see the commands and data in a command file as the file is processed; in that case, leave the :T out of your file.) A :T takes precedence over any :S symbols that follow it in the file.

:S SILENCE - Use a :S symbol to suppress the display of command file lines as AMOS processes them. (However, a :S will have no effect if a :T appears before it in the command file.) Any program output generated while the command file is in control also does not appear on your terminal.

:R REVIVE - Use a :R symbol to counteract a :S symbol. You can use multiple :S and :R symbols within one command file to allow the user of your command file to see some program output and command file lines, but not others.

:<...> MESSAGE - These symbols allow you to include messages and comments within your command files. AMOS displays all characters within the <> symbols on your terminal display when it reaches that point in the command file. A command file message is not acted upon by AMOS or any other program; AMOS displays command file messages regardless of whether :S or :R symbols appear in the command file. A message may be more than one line in length; the end of the message is indicated by the > symbol and not a RETURN. (NOTE: AMOS ignores any characters on the line that follow the end-of-message symbol; that is, AMOS skips over any characters between a > and a RETURN.)

:K KEYBOARD INPUT - The :K symbol allows the user of your command file to enter one line of data or commands to either AMOS or the program currently being executed. When it finds a :K symbol in your command file, AMOS halts processing of the file until the user of your file enters a line of characters that end with a RETURN.

; COMMENT CHARACTER - the semicolon symbol marks a comment line which is not processed, but is displayed with the rest of the command file if a :T is present at the top of the file.

Below is a small command file that uses some of these special symbols. The command file BACKUP.CMD transfers copies of the .MAC files in account DSK0:[300,1] to account DSK1:[300,5], changes the extensions of the files in DSK0:[300,5] to .OLD (to indicate inactive, archive files), and then erases all backup files in DSK0:[300,1]:

```
:<THIS FILE TRANSFERS ACTIVE .MAC FILES FROM DSK1:[300,1] TO DSK0:[300,5]
>
COPY DSK1:[300,5]=DSK0:*.MAC[300,1]
RENAME *.OLD=DSK1:*.MAC[300,5]
:R
:<
ANSWER Y OR N TO CONFIRM OR ABORT EACH ERASURE OF A .BAK FILE
>
ERASE DSK0:*.BAK[300,1]/QUERY
:<
IF YOU WISH TO SEE THE DIRECTORY FOR ACCOUNT DSK1:[300,5], TYPE A 'RETURN';
OTHERWISE, TYPE A CONTROL-C.
>
:K
DIR DSK1:[300,5]
```

Unlike our previous examples, the command file above does NOT start with a :T; that means that the user of the file sees only the command file messages (those characters enclosed with :<>) until the point in the command file where the :R appears. (The :R allows the user to see the output of the ERASE and DIR programs.) The :K near the end of the file allows the user of your command file to abort the use of the command file by typing a Control-C.

8.2 DO FILES

A very special type of command file that allows you to pass arguments to that file is called a DO file (and has a .DO extension).

A DO file contains exactly the same type of elements as a regular command file (including the special symbols mentioned above), but also includes some additional symbols that allow you to specify items of text to be substituted into the command file at the time that you invoke the file. (These additional parameter symbols are: $0,$1,$2,.....$9.) Because you can pass text items to a DO file, you can use DO files in many different situations in which a regular command file would be too specific.

For example, suppose you are working on a set of BASIC programs. You may enter the same commands every time you test a new program. Still, you can't use a standard command file because you do not know beforehand what the name of your BASIC program is going to be. Create a command file with the .DO extension, and place one of the parameter symbols ($0) in the file where you want to be able to substitute in the BASIC program name:

```
:T
:< DO FILE TO COMPILE AND RUN BASIC PROGRAMS.
>
;
; Erase whatever's in my memory partition.
DEL *.*

;
; Load in BASIC subroutines we'll need.
LOAD DSK0:XLOCK.SBR[7,6]
LOAD DSK0:BASORT.SBR[7,6]
LOAD DSK0:XMOUNT.SBR[7,6]
;
; Compile and run the BASIC program
COMPIL $0.BAS
RUN $0.RUN
;
; Get rid of any backup files in the account.
ERASE *.BAK
;
; Clean up memory partition again.
DEL *.*
```

If the DO file above is named TEST.DO, then we can invoke it by entering the name of the file at AMOS command level along with the item of text we want to substitute into the file for the $0 symbol:

```
.TEST CANCEL↵
```

As AMOS processes the file above, you see the DO file on the screen. The lines in the file that contain the parameter symbol $0, now contain the argument you specified when invoking the DO file:

```
; Compile and run the BASIC program
COMPIL CANCEL.BAS
RUN CANCEL.RUN
```

8.2.1 Building and Invoking DO Files

Create a DO file by using one of the text editors to build a text file with the .DO extension. The file may contain up to ten different user-defined parameters (indicated by the parameter symbols $0-$9).

When you enter the name of the DO file at AMOS command level, you also include an argument list, the items of which will be substituted for the parameter symbols in the DO file. The items in the argument list are separated by blanks. (To include a blank within an argument, enclose the argument within <> symbols-- e.g., <argument #2>.)

Each parameter symbol becomes associated with one of the argument list items (the first item with parameter $0, the second item with parameter $1, and so on). For example:

```
:T
TXTMFT $0
PRINT $1
DIR $2
```

If we invoke this DO file in the following way:

```
.DOC PSTINV.TXT PSTINV.LST PSTINV.* ↵
```

the $0 becomes associated with PSTINV.TXT, the $1 becomes associated with PSTINV.LST and $2 becomes associated with PSTINV.*. the DO file above becomes transformed into:

```
:T
TXTFMT PSTINV.TXT
PRINT PSTINV.LST
DIR PSTINV.*
```

You can use a parameter symbol to represent an entire filespec, a portion of a filespec, a command or any other piece of text inside a DO command file. For example:

```
:T
;$0
TXTFMT AMS$1:$2.TXT
PRINT AMS$1:$2.LST
DIR AMS$1:$2.*
```

when invoked with:

```
.FORM <FILE IS PSTINV> 2 PSTINV ↵
```

is transformed into:

```
:T
;FILE IS PSTINV
TXTFMT AMS2:PSTINV.TXT
PRINT AMS2:PSTINV.LST
DIR AMS2:PSTINV.*
```

If you have more items in your argument list than there are parameter symbols in the DO file, the extra items are ignored. If you have fewer items in your argument list than there are parameter symbols, the extra parameters are ignored. Argument list items are associated with parameter symbols NOT in the order that the parameter symbols appear in the file, but in the order in which the parameter symbols are numbered. (That is, the first item in the argument list is associated with parameter $0, even if parameter $2 appears before $0 in the DO file.)

8.2.2 Special Parameter Symbols

In addition to the usual DO file parameter symbols, four special parameter symbols allow you to use DO files in a more flexible way and for a greater range of applications.

$: CURRENT DEVICE SYMBOL - Represents the device that the user of the DO file is currently logged into. For example, if the user of your DO file is logged into an account on DSKÜ: at the time that he or she uses your DO file, the line in the DO file:

```
LOG $:[1,4]
```

is transformed into:

```
LOG DSKÜ:[1,4]
```

You can use this symbol in combination with the Current PPN Symbol (below) to keep track of the current account and device of the user of your DO file; the DO file can log the user into another account to perform special functions, and then return him to his own account and device.

$P CURRENT PPN SYMBOL - Represents the account that the user of the DO file is currently logged into. For example, if the user of your DO file is logged into account [230,5], the DO file statement:

```
DIR DSKÜ:[$P]
```

is transformed into:

```
DIR DSKÜ:[230,5]
```

$ NULL PARAMETER SYMBOL - A single $ indicates a null parameter in a default parameter list or a null argument in an argument list. This symbol allows you to designate which parameter will be associated with which argument. Take a look at the example below for an idea of how to use the $ symbol.

$D DEFAULT PARAMETER LIST - If you specify fewer items in the argument list than there are parameter symbols in the DO file, AMOS usually ignores the extra parameter symbols. You can, however, supply a default argument list that AMOS will use if you omit an argument list on the DO file command line, or if you do not supply a complete argument list. For example, the ERASE command's default extension is an empty extension. You can create your own ERASE command (in this case named REMOVE) in which the default extension is *.

```
$D $ .*
:T
ERASE $0$1
```

So, if you invoke the DO file with an argument list containing just a filename, you see something like this:

```
.REMOVE PSTINV↵

ERASE PSTINV.*
```

NOTE: If a $D line appears in your DO file, it MUST be the first line of the file (even before a :T symbol).

8.2.3 SAMPLE DO FILES

Below are some examples of the kinds of DO files you can create to help you perform frequently used sequences of commands. If you want all users on the system to be able to share your command files and DO files, have the System Operator copy them over to the System Command File Library (DSK0:[2,2]).

8.2.3.1 TFORM.DO - You can use the text formatting program, TXTFMT, to format a group of text files so that they form one document. If you always begin the filespec list with the same file or files, a DO file can be convenient:

```
:T
TXTFMT HEADER,PRPNOT,PRFACE,$0
```

For example, the DO file above tells TXTFMT to format a header file (HEADER) that contains standard formatting information (page size, line size, etc.), a file that contains an official proprietary notice (PRPNOT), and a file containing a standard preface (PRFACE). Let's say that these three files are always formatted in front of a document. You can call TFORM.DO with a filespec that specifies the main body of the document:

```
.TFORM YEARLY↵
TXTFMT HEADER,PRPNOT,PRFACE,YEARLY
```

8.2.3.2 PRINTE.DO - When you are printing a long list of files (e.g., BASIC programs), it is often convenient to separate those files by sending a form-feed character to the printer after each file that you print; then each file begins at the top of a page.

When you call PRINTE.DO, give it the filespec of the file you want to print. PRINTE prints the file, and then sends a form-feed character (FF.TXT) to the printer.

```
PRINT $0
; FF.TXT is in System Command File Library
PRINT DSK0:FF.TXT[2,2]
```

FF.TXT is a text file that contains only one symbol-- a form-feed character. To create FF.TXT, follow these steps:

1. Type MAKE FF.TXT followed by a RETURN:

   .MAKE FF.TXT ↵

2. Type EDIT FF.TXT followed by a RETURN:

   .EDIT FF.TXT ↵

3. Now you see the EDIT prompt: *. Type 12I followed by two Escapes. (The two Escapes show up on the screen as two dollar signs):

   *12I$$

4. Type an E followed by two Escapes (the EDIT exit command):

   *E$$

5. You are at AMOS command level again and you have just created a text file that contains a form-feed character. (The EDIT command "12I" told EDIT to I(nsert) the numerical data 12 into your file-- this happens to be the ASCII code for a form-feed.)

8.2.3.3 BACKUP.DO - This DO file transfers backup copies of the specified files in account [200,1] from one disk cartridge (or floppy disk) to the same account on another disk cartridge (or floppy disk):

```
$D *.*
:T
SET VERIFY      ; Check write operations and report disk errors
SET DSKERR
:<Backup Account [200,1] of Working Cartridge onto Archives Cartridge
>
;
LOG DSK1:[200,1]
COPY DSK0:=$0   ; Copy specified files in DSK1:[200,1] onto DSK0:
;
:<Change cartridge to Archives Cartridge; when ready, MOUNT--
>
:K
MOUNT DSK1:
COPY =DSK0:$0   ; Copy specified files in DSK0:[200,1] to DSK1:[200,1]
;
:<Change cartridge back to Working Cartridge; when ready, MOUNT--
>
:K
MOUNT DSK1:
:<Clear DSK0:
>
ERASE DSK0:*.*[200,1]
LOG $:[$P]      ; Return to previous account.
```

8.2.3.4 WRITE.DO - Uses the COPY command to send a file to any printer without using the PRINT command (i.e., without going through the line printer spooler program). WRITE.DO allows you to send a file to any terminal (including a printer) as long as that device has been defined on the system as a terminal. The format with which you call WRITE.DO is:

```
.WRITE Filespec TO terminal-name ↵
```

Argument #0 = Filespec, argument #1 = "TO" and argument #2 = terminal-name.

```
$D $ $ TRM6     ; The default terminal is TRM6
; Send file to designated terminal (or printer)
:T
COPY TRM:$2=$0
```

NOTE: Because we do not specify a parameter $1 in the DO file, the second argument ("TO") is ignored. We have defined a default argument for parameter $2-- a terminal named TRM6:

```
.WRITE DSTRIB.BAS TO QUME↵

.COPY TRM:QUME=DSTRIB.BAS↵
DSTRIB.BAS to TRM:DSTRIB.BAS
Total of 1 file transferred

.WRITE CMPTX.RUN↵

.COPY TRM:TRM6=COMTX.RUN↵
COMTX.RUN to TRM:COMTX.RUN
Total of 1 file transferred
```

8.2.3.5 ASSMBL.DO - Assemble, link and test a .PRG file whose source is in three .MAC files:

```
:T
; Assemble it.
;
MACRO $0/T
MACRO $1/T
MACRO $2/T
;
; Link the three .OBJ files that result
;
LINK $0,$1,$2
;
; Create symbol table file for the three .OBJ files
;
SYMBOL $0,$1,$2
;
; Run the assembled and linked .PRG file
;
$0.PRG
```

To call the DO file, supply an argument list containing the names of three .MAC files:

```
.ASSMBL DCOPY1 DCOPY2 DCOPY3↵
```

CHAPTER 9

THE WILDCARD FILE COMMANDS

This chapter and the next describe the commands that you will use most frequently in handling files. There are other file commands that we do not discuss in this manual; you will find a list of ALL of the commands you can use on files in the "AMOS System Commands Reference Sheets." For now, just remember that there are several programs not discussed in this manual that help you to create and access files; for an introduction to those programs, see the manual titled "Introduction to AMOS."

9.1 INTRODUCTION TO WILDCARD FILE COMMANDS

This chapter describes five of the commands that you can use on files: COPY, DIR, ERASE, PRINT and RENAME. We have grouped these commands apart from the rest of the file commands because of the special abilities they share; even though these five commands perform very different functions, they have in common an advanced ability to recognize wildcard file specifications that allows them to process file specifications differently than do the rest of the commands on the system. In honor of their special talent for using wildcards, we have named these commands "wildcard file commands."

In addition to their own functions, several of these commands also allow you to perform the same functions as other system commands (with the added benefit of sophisticated file specification wildcarding). Refer to the sections in the documentation for each command that discuss the use of that command and ersatz and special devices for more information on these special functions.

We have already discussed the wildcard symbols * and ? (see Chapter 6, Identifying a File to AMOS). A specification that contains a wildcard symbol can represent more than one file (e.g., *.TXT may represent the files JAN.TXT, FEB.TXT and MARCH.TXT). Wildcard file commands process the wildcard symbols * and ? somewhat differently than do other commands, and they have other wildcarding abilities that the other commands lack. As the system evolves in the future, you will probably begin to see more commands processing file specifications in the same way as the current wildcard file commands.

Before discussing each of the commands in detail, we'd like to present a few of the basic rules that all wildcard file commands follow when processing a command line.

9.1.1 Wildcard Symbols

The wildcard file commands recognize the following wildcard symbols in file specifications (see HINTS AND RESTRICTIONS, below):

*	Matches any symbol or symbols in a filename, extension or PPN (e.g., GL*.TXT matches the files GLDGR.TXT, GLZ.TXT, and GLOW1.TXT, because it selects all .TXT files whose names begin with GL).
?	Matches any one symbol in a filename, extension or PPN (e.g., PAY?LL.FNX matches PAYRLL.FNX, PAY2LL.FNX, and PAYTLL.FNX, because it selects all files whose names begin with the three letters "PAY," and that have the fifth and sixth letters "LL").
ALL:	Matches any file-structured, mounted device (e.g., ALL:PRESR.BAS matches the files DSKÜ:PRESR.BAS, AMS1:PRESR.BAS, and STD1:PRESR.BAS).
dev:	Matches any unit of a file-structured, mounted device. If you omit the unit number from a device specification, wildcard file commands look for the specified files on all logical units of that device.
[]	Matches any PPN (e.g., WRK1.BAS[] matches the files WRK1.BAS[23,4], WRK1.BAS[110,4] and WRK1.BAS[100,3]). Equivalent to [*,*].

Throughout this chapter we'll be showing examples of the use of these wildcard symbols in the discussions of the specific wildcard file commands.

HINTS AND RESTRICTIONS:

In addition to the restrictions mentioned in Section 6.2, Wildcard Symbols, there are some minor restrictions on the use of the above wildcard symbols:

1. When you use the * symbol in a PPN, you may either use the symbol to represent the entire project or programmer number (e.g., [*,21], [100,*]), or you may follow the symbol with one or more numbers (e.g., [*20,34],[150,*5]). You may NOT place numbers before the symbol (e.g., [34*,5]). Examples of wildcard PPNs:

WATCH.LSP[*,21] matches WATCH.LSP[100,21], WATCH.LSP[230,21]

WLDCRD.TXT[220,*] matches WLDCRD.TXT[220,4], WLDCRD.TXT[220,57]

SYSTM1.FD[*1,5] matches SYSTM1.FD[1,5], SYSTM1.FD[301,5]

2. If ?s appear at the beginning of a PPN project or programmer number, that many or FEWER numbers will match. (For example, PROJCT.CMD[??1,??] matches any files named PROJCT.CMD in accounts whose project numbers end with 1 and are one two or three digits long, and whose programmer numbers are one or two digits long.) If ?s appear at the end or in the middle of a PPN project or programmer number (e.g., [1?0,2??]), EXACTLY that many numbers may match the ? symbols.

9.1.2 Input File Specifications

Input filespecs (or infilespecs) are file specifications that select the files on which you want a command to take action. For example, let's say that you want to erase several files:

```
.ERASE DSK0:MCSAM.RUN,DSK1:MCELI.RUN
```

The filespecs DSK0:MCSAM.RUN and DSK1:MCELI.RUN are the infilespecs in the command line above. When you use wildcard symbols in an infilespec, you are asking one filespec to select a group of files. Instead of the example above, we might have said:

```
.ERASE DSK:MC*.RUN
```

The command above selects all files whose names begin with MC, that have an extension of .RUN and which exist in the account you are currently logged into on ALL units of device DSK:.

INFILESPEC DEVICE AND ACCOUNT DEFAULTS:

If you completely omit account and device specifications from a group of infilespecs, wildcard file commands use as their initial default the account and device you are currently logged into. Unlike other AMOS commands, these commands also allow you to set those defaults. That is, you can set the account and device defaults for a specific group of infilespecs. If you omit those specifications, the command will use the defaults that you have set.

1. Whenever you include a device specification in an infilespec, you set the default device for the rest of that command line. For example, let's assume that you are logged into an account on DSK0:.

```
.DIR RESRCH.DAT,AMS1:CURVES.RUN,MAXWLL.RUN
```

The example above looks for RESRCH.DAT on DSKU: (the initial device default), and then looks for CURVES.RUN on device AMS1:. The default device now becomes AMS1:, so DIR looks for MAXWLL.RUN on AMS1: as well. To return the device default to DSKU:, you must explicitly tell DIR to look for a file on that device (i.e., in the example above, to reset the default device to the initial default, you must change MAXWLL.RUN to DSKU:MAXWLL.RUN).

2. To set the default account, place the PPN at the front of an infilespec. For example:

```
.DIR LAWYER.LSP,[120,34]OFFICE.BAS,LAWDOC.TXT,OFFICE.RUN[230,1]
```

The initial default is the account you are logged into. We reset the default account to [120,34] by placing that PPN in front of a filespec ([120,34]OFFICE.BAS). Now DIR searches in that account for any of the files on the rest of the command line for which we have omitted a PPN (in this case, LAWDOC.TXT). You can reset the account default to ALL accounts by using the wildcard PPN symbol, [].

3. If an infilespec does not specify a PPN project or programmer number (e.g., [,12], [300,]), the wildcard file commands use the current default project or programmer number. This example:

```
.ERASE [12,34]MTST.MAC,MTST.OBJ[,35],[110,2]TNK.BAS,TNK.R2[,5]
```

evaluates to:

```
.ERASE MTST.MAC[12,34],MTST.OBJ[12,35],TNK.BAS[110,2],TNK.R2[110,5]
```

That is, since we have omitted the project number for MTST.OBJ and TNK.R2, the ERASE command substitutes in the current default project number at those places in the command line.

4. The wildcard file commands all have one or more switches that you can use to select command options. For example, when you use the /QUERY switch with the ERASE command (e.g., .ERASE/QUERY *.TXT), that command asks you to confirm its action before it erases each file. See Section 9.1.4, Command Switches, for information on switches and on setting the default switch for a command line.

9.1.3 Output File Specifications

Several of the wildcard file commands require that you supply an output file specification (an outfilespec) as well as one or more infilespecs. While an infilespec specifies a file on which a command is to act, an outfilespec gives information to the command on HOW to act. For example, the RENAME command below renames the file ORDERS.DAT (the infilespec) to ARCVBL.DAT (the outfilespec):

```
.RENAME ARCVBL.DAT=ORDERS.DAT ↵
```

The infilespec selects the file on which to act (ORDERS.DAT), and the outfilespec tells RENAME the new name to give to the file. Outfilespecs modify selected portions of infilespecs. In the case of the RENAME command above, we replace the filename and extension of the infilespec with the filename and extension of the outfilespec.

When an outfilespec contains wildcard symbols, it only partially modifies the infilespec; those portions of the infilespec that correspond to the wildcard portions of the outfilespec are left alone. For example:

```
.RENAME *.OLD=MACROS.NEW,WRKFIL.NEW ↵
MACROS.NEW TO MACROS.OLD
WRKFIL.NEW TO WRKFIL.OLD
Total of 2 files renamed
```

In the example above, the filename portion of the outfilespec (*.OLD) is a wildcard symbol. The filenames of the infilespecs are left unchanged, and only the extension is modified. We might have achieved the same effect by using a wildcard infilespec to select the two input files MACROS.NEW and WRKFIL.NEW:

```
.RENAME *.OLD=*.NEW ↵
MACROS.NEW TO MACROS.OLD
WRKFIL.NEW TO WRKFIL.OLD
Total of 2 files renamed
```

Another example of wildcard use in outfilespecs:

```
.RENAME GL*=*.AR ↵
AR1FIL.AR TO GL1FIL.AR
AR2DAT.AR TO GL2DAT.AR
UNBILL.AR TO GLBILL.AR
Total of 3 files renamed
```

The example above takes the files selected by the infilespec *.AR, and replaces the first two letters of the filename with GL.

9.1.4 Command Switches

The wildcard file commands recognize a switch by the fact that it begins with a slash (e.g., /Q); switches may appear anywhere on the command line. A typical command switch might be /QUERY or /Q, which tells the command to ask for confirmation before it acts on each input file. You may include more than one switch on a command line as long as you precede each switch with a slash:

```
.RENAME/QUERY/DELETE *.BAS=*.BS1,*.BS2 ↵
```

You may usually abbreviate switch names to just the letters that uniquely

identify that switch. For example, you may enter the /QUERY switch by typing either /QUERY or /Q. For each switch there often exists another switch of opposite action, identified by the prefix "NO": e.g., /DELETE and /NODELETE, /QUERY and /NOQUERY.

Wildcard file commands use two different kinds of switches: OPERATION SWITCHES and FILE SWITCHES. An OPERATION SWITCH has the same effect no matter where it appears on the command line, and it affects all filespecs on that command line:

```
.DIR/DATA *.TXT[,75],SYNCH[,37] ↵
```

has the same effect as:

```
.DIR *.TXT[,75]/DATA,SYNCH[,37] ↵
```

A FILE SWITCH can apply to only specific filespecs, depending on where it appears on the command line. If a FILE SWITCH appears directly after the command (e.g., .RENAME/Q *.MAC=*.TXT,*.MC1) then the switch applies to all filespecs. For example, the command below asks for confirmation before it erases each of the files selected by the filespecs:

```
.ERASE/Q TYPSET.MAC,TYPSET.OBJ,*.SYM ↵
```

If a FILE SWITCH appears directly after an infilespec, then it applies only to the files selected by that specification:

```
.ERASE WLDCRD.TXT,*.LST/Q,DATA.MAC ↵
```

only asks for confirmation before erasing files that match the specification *.LST.

Refer to the documentation for the specific command you are using to see if a switch is an OPERATION or a FILE SWITCH.

To set the default switch for a command line, place the new default switch in front of a filespec; that switch will become the default for the command line until you reset the default by placing another switch in front of a filespec. (Placing a switch at the end of a filespec DOES NOT reset the default switch.) For example, the ERASE command does not initially ask for confirmation of erasures; that is, the default switch is /NOQUERY. The command below begins with a default switch of /NOQUERY, resets the default switch to /QUERY, and then sets it back again to /NOQUERY:

```
.ERASE REDWNG.DAT,BLKBRD.*,/QUERY BLUBRD.*,ROBIN.*,/NOQUERY CANARY.RT,GNAT.* ↵
```

9.1.5 Ersatz Devices

Wildcard file commands recognize both the standard system storage devices (e.g., DSK0: and AMS1:) and the system special devices TRM: and MEM:. (Check with your System Operator if you are unable to use TRM: or MEM:; he or she will check to make sure that the programs TRM.DVR and MEM.DVR are in account [1,6] of the System Disk, and that TRM: and MEM: are defined system devices.)

Aside from the special devices (which many components of the system recognize), there exists another class of device that only the wildcard file commands and the LOG command recognize-- the ersatz devices. The purpose of the ersatz devices is to make it easier for you to specify the accounts you use most frequently. These devices are:

RES:	specifies	System Memory	
SYS:	specifies	DSK0:[1,4]	System Program Library
DVR:	"	DSK0:[1,6]	Device Driver Library
CMD:	"	DSK0:[2,2]	Command File Library
HLP:	"	DSK0:[7,1]	Help File Library
BOX:	"	DSK0:[7,2]	Mailbox Data File Library
LSP:	"	DSK0:[7,4]	LISP Language Library
PAS:	"	DSK0:[7,5]	Pascal Language Library
BAS:	"	DSK0:[7,6]	BASIC Language Library
MAC:	"	DSK0:[7,7]	MACRO Language Library

For example, .DIR SYS: is equivalent to .DIR DSK0:[1,4]. That is, by typing .DIR SYS: you can see a directory display for account DSK0:[1,4]. .DIR RES: gives a directory listing of the memory modules in system memory.

9.2 FINDING OUT WHAT FILES ARE ON THE DISK (DIR)

Every account has a directory which lists all of the files in that account. You will probably use the DIR (Directory) command most often to find out what files are in your own account; you can also use the DIR command in several different ways to gain information about accounts other than your own, and to locate specific files by account.

COMMAND SYNTAX:

.DIR {listfilespec=}{filespec1{,...filespecN}}{/switch{/switch}} ↵

COMMAND DEFAULTS:

The default listfilespec is a filename and extension of DIRECT.LST, and the account and device you are logged into.

The default filespec is a filename and extension of *.*, and the device and account you are logged into.

The default switches are /WIDE:1/NOBASE.

SPECIAL FUNCTIONS:

You can use DIR to perform the same functions as the MAP and SYSTEM commands (see Section 9.2.7, DIR and Special and Ersatz Devices).

9.2.1 Finding Out What Files Are in Your Account

(For the purposes of this example, let's say that you are logged into account [100,1] on DSK1:.) Type DIR followed by a RETURN:

```
.DIR ↵
SCHED   BAK 14     DSK1:[100,1]
SCHED   TXT 15
PARTNO  BAS 8
PARTNO  RUN 6
TEST    MAC 12
Total of 5 files in 55 blocks
```

(Remember that you can use Control-S to freeze the screen display if you have so many files in your account that DIR cannot list them all on one screen-page. Use a Control-Q to release the screen display.)

The directory you see is the directory of the account into which you are currently logged. In this case, you see a list of all of the files in your account.

The right hand of the display tells you which account directory you are seeing (DSK1:[100,1]). The left hand of the display lists all of the files in the account by name, extension and size in blocks. (Files are written to the disk in units called blocks. One block is 512 bytes.) The last line of the display tells you how many files are shown in the directory display, and how many blocks on the disk are used by those files.

9.2.2 Finding Out What Files Are in an Account Other Than Your Own

To find out what files are in an account other than the one you are currently logged into, you must tell DIR which account directory to display. Type DIR followed by the specification of the account or accounts whose directory you want to see:

```
.DIR DSK0:[200,5] ↵
TABLE   MAC  6        DSK0:[200,5]
TREE    LSP  10
VECTR   MAC  8
Total of 3 files in 24 blocks
```

If you specify just a PPN (that is, if you just type DIR [200,5]), DIR looks for that account on the device you are currently logged into. If you

specify just the device (e.g., .DIR AMS1:), DIR searches that device for the account you are currently logged into.

You may see directory listings for more than one account by following the DIR command with several account specifications, separated by commas:

```
.DIR AMS1:[56,1],[120,5]↵
FORMS  TXT  23          AMS1:[56,1]
CLASS  BAS  10
Total of 2 files in 33 blocks

CHPTR1 TXT  47          AMS1:[120,5]

Grand total of 3 files in 80 disk blocks
```

If more than one file is listed for each account, a message tells you how many files are in the directory listing for that account, and how many disk blocks are used by those files. The last line of the display tells you the number of files listed in the entire display, and how many blocks are used by all of the files.

Since we left off filenames and extensions in the example above, DIR assumed that we wanted directory displays for ALL files on disk AMS1: that belong to the accounts [56,1] and [120,5].

* Be careful here. Remember that DIR is a wildcard file command. By specifying device AMS1:, we have set the default device for the rest of the command line. That means that even if you are logged into DSK0:, the DIR command above looks for account [56,1] AND account [120,5] on AMS1:. If what you meant to say was, "Show me all files in account [56,1] on AMS1:, and all files in account [120,5] on the device I am currently logged into," you must say:

  ```
  .DIR AMS1:[56,1],DSK0:[120,5]↵
  ```

  where you explicitly specify the device on which to look for account [120,5]; or:

  ```
  .DIR [120,5],AMS1:[56,1]↵
  ```

  where the default is set to AMS1: after DIR looks for account [120,5] on the device you are logged into.

9.2.3 DIR and Wildcard Symbols

You can use the various wildcard device and account specification symbols to tell DIR to display directories for a group of accounts. For example, to see displays of all accounts on a device, use the wildcard PPN symbol []:

```
.DIR DSK1:[]↵
DO      HLP  12            DSK1:[100,2]
VUE     HLP  23
LOG     HLP  3
Total of 3 files in 38 blocks

PAGE    PRG  14            DSK1:[200,5]
PAGE    MAC  20
HEADR   MAC  7
Total of 3 files in 41 blocks

STAT    TXT  57            DSK1:[200,10]
INIT    TXT  2
Total of 2 files in 59 blocks

Grand total of 8 files in 138 disk blocks
```

You can also use wildcard symbols in the PPN specification (e.g., DIR [*,10] or DIR [100,*]) to select all accounts with the same project or programmer number. To see directory listings for a specific account located on all mounted disks, use the wildcard device symbol ALL: (e.g., DIR ALL:[300,1]), or use a wildcarded unit specification (e.g., DIR DSK:[101,2] to select DSK0:[101,2], DSK1:[101,2] and DSK2:[101,2]).

9.2.3.1 Using DIR to Find Specific Files - You will find that DIR is very useful for locating files (that is, finding out what account a specific file belongs to), and for displaying a directory listing of only certain files in an account. Until now we've given DIR only the portions of file specifications that identify accounts, and have thus asked DIR for a directory listing of ALL of the files in the accounts we have specified. By including filenames and extensions, you can instruct DIR to list ONLY certain files in the directory display. For example, if the BASIC program APSBR.BAS exists in your account, you may ask for a directory display that lists only that file:

```
.DIR APSBR.BAS↵
APSBR   BAS  23            AMS1:[156,3]
Total of 1 file in 23 blocks
```

This may not seem very useful in itself, until you consider the use of the wildcard symbols in file specifications. By using wildcards, you can ask DIR for a listing of all files that have a specific extension or name. For example:

```
.DIR *.BAS↵
APSBR   BAS  23            AMS1:[156,3]
ARSBR   BAS  14
GLSBR   BAS  12
DATAEN  BAS  7
Total of 4 files in 66 blocks
```

In the example above, you have asked for a directory listing of all BASIC files in your account. You could just as easily have asked for a display of all files that have a five-character name that ends with SBR (e.g., .DIR ??SBR), or all files that have names that begin with an A (e.g, .DIR A*). (NOTE: The default filespec for DIR is *.* and the device and account you are currently logged into. Therefore, omitting a file extension causes DIR to use the default extension, *. In other words-- specifying .DIR A* is equivalent to specifying .DIR A*.*.)

You can use the device wildcard symbols when you want DIR to locate files for you. For example, suppose you know that several accounts on DSK2: contain some special text files; use the wildcard PPN symbol to find the accounts where the files appear:

```
.DIR []DSK2:OPSER*.TXT,OPDOC*.TXT↵
OPSER1 TXT 12              DSK2:[110,4]
OPSER3 TXT 23
Total of 2 files in 35 blocks

OPSER2 TXT 8               DSK2:[110,7]

OPDOC1 TXT 25              DSK2:[100,35]
OPDOC4 TXT 17
Total of 2 files in 42 blocks

Grand total of 5 files in 85 disk blocks
```

In the example above, we set the default account to ALL accounts on DSK2: by using the wildcard PPN symbol [].

9.2.4 Creating a File That Contains a Directory Listing

Type DIR, the name of the file you want to create, an equal sign and the specifications of the accounts for which you want a directory listing:

```
.DIR MYFILS.LST=DSK1:[255,12]↵
```

The example above creates a file named MYFILS.LST in the account you are logged into; this file contains a listing of the directory for DSK1:[255,12]. If you leave the command line after the equal sign blank (e.g., .DIR MYFILS.LST=), DIR assumes you want a listing for the directory of the account you are currently logged into. If you leave the area between the DIR command and the equal sign blank, DIR creates a listfile named DIRECT.LST for you in your account. (By implication, then, a command of "DIR = " creates a file named DIRECT.LST that contains the directory listing for the account and device you are currently logged into.)

Experiment with using DIR to create a directory file, and you will find that you can create a file in any account within your own project (i.e., the PPN of the account in which you are creating the file and the PPN of the account you are logged into have the same project number). The file you create may contain a directory listing for ANY account, even those outside of your own project.

9.2.5 Printing a Directory Listing

To print a copy of a directory listing you can either: 1. create a file that contains the directory listing (see Section 9.2.4, above), and then print that file using the PRINT command (see Section 9.6, Printing a File--PRINT); or, 2. use the DIR command to directly send a directory listing to a printer.

To send a directory listing to a printer (or to any other kind of terminal), enter:

```
.DIR TRM:printer-name=(filespec)
```

where "printer-name" is the name of the terminal to which you wish to send the listing. (If you specify a printer, the printer must be defined on the system as a terminal). To see the names of the terminals defined on the system, type TRMDEF followed by a RETURN. You will probably need to ask the System Operator which of these terminals are printers.

Let's say that the printer you want to use has been defined on the system as terminal TERM6. To send to that printer a directory listing of all of the BASIC files in account [300,5], type:

```
.DIR TRM:TERM6=*.BAS[300,5]
```

9.2.6 Selecting DIR Options

By including one or more switches on the DIR command line, you may select several of the DIR options. The switches are:

/DATA	or /D	Just lists complete filespecs, one per line.
/KILL	or /K	Deletes and replaces existing listfile if it has same specifications as your listfilespec.
/WIDE	or /W	Arranges directory listing in four columns.
/WIDE:n	or /W:n	Arranges directory listing in n columns.

/HASH or /H	Displays a hashmark for each file. (A hashmark is a computed value based on characteristics of the file.) Serves to help you distinguish between different versions of the same file.
/BASE or /B	Displays the starting disk address (in octal) of the file.
/NOBASE or /NOB	Turns off the /BASE option.
/CONTIGUOUS or /C	Displays a C next to the extension of a contiguous file. Allows you to distinguish between random (that is, contiguous) files and sequential files.
/FULL or /F	Gives you full range of information options. Same as specifying /H/B/C.

You may place more than one switch on a command line as long as you precede each switch with a slash. For example:

```
.DIR/H/W:2 [110,2]↵
```

DATA OPTION: The /D switch tells DIR to list the complete file specification for each file in the directory listing. For example:

```
.DIR/D [130,1],[130,5]↵
AMS1:COM.BAS[130,1]
AMS1:FF.CMD[130,1]
AMS1:INTRO.TXT[130,1]
AMS1:AUGMNT.LSP[130,1]
AMS1:START.BAS[130,5]
AMS1:COM.BAS[130,5]
AMS1:COM.RUN[130,5]
```

One fact that makes the /D switch of considerable use, is that a directory listfile created while the /D switch is in effect, contains file specifications in a form that is readable by BASIC. BASIC programs can therefore read the listfile to gain a list of files to open for data transfer.

The /D switch is an operation switch; it acts the same no matter where you place it on the DIR command line.

```
.DIR/D [100,2],[110,6]↵
```

produces the same directory listing as:

```
.DIR [100,2]/D,[110,6]↵
```

KILL OPTION: If you use DIR to create a file containing a directory listing,

DIR will not create the new file if a file with the same specifications already exists on the disk. Use the /K switch to tell DIR to destroy any existing file whose specifications match those of your new file.

```
.DIR/K ADIRCT.LST=[110,4]↵
```

The /K switch is an operation switch.

WIDE OPTION: As you increase the number of files in your account, you will often find that a directory display overflows your screen. So that you can inspect the listing at your leisure, you may either use Control-S and Control-Q, or you can use the /W option. The /W:n option tells DIR to arrange your directory listing in several columns. (The number of columns = the number that follows the colon.) For example:

```
.DIR [100,1]/W:2↵
AMSO:[100,1]
MENU    BAS  15     ALOAD   BAS  20
MDOC    TXT  43     SACCNT  BAS  14
FF      DO   1
Total of 5 files in 93 blocks
```

produces a two-column directory display. If you omit the colon and number after the /W, DIR assumes that you want a directory listing of four columns. (Four columns is the default because it makes for a nice screen display.)

```
.DIR/W *.TXT↵
DSK1:[110,3]
HEADER  TXT  13     MINE  TXT  12     INTRO  TXT  8     OUTLIN  TXT  23
START   TXT  4      TEST  TXT  2      TURN1  TXT  6     AFILE   TXT  5
Total of 8 files in 73 blocks
```

The /W switch is an operation switch.

HASH OPTION: It often happens that as you develop a program or a document, you have several different versions of the file in various accounts. If they are all named the same, how do you find a specific version (e.g., the latest) of the file? One way is to keep track of the files' hashmarks so that you can tell them apart. A file hashmark is a value that is computed based on characteristics of that file. No two files have the same hashmarks unless those two files are identical; even a difference of one character results in different hashmarks.

The /H option tells DIR to compute and display hashmarks for the files you specify. /H is a file switch, which means that where you place the switch on the command line determines which files are affected by it.

To affect all of the files specified, place the /H switch directly after the DIR command itself on the command line. For example, the command line below tells DIR to compute and display hashmarks for all of the files specified:

```
.DIR/H SCAN.PRG[100,3],SCAN.PRG[110,5],AMS0:SCAN.PRG[120,5]
SCAN   PRG  12  604-027-223-636    AMS1:[100,3]

SCAN   PRG  13  650-353-035-656    AMS1:[110,5]

SCAN   PRG  21  355-642-671-471    AMS0:[120,5]

Grand total of 3 files in 46 disk blocks
```

To get hashmarks for specific files, place the /H switch after the specifications of the files for which you want hashmarks. For example, this command tells DIR to compute and display hashmarks for only the second of the files specified:

```
.DIR SCAN.PRG[100,3],SCAN.PRG[110,5]/H,AMS0:SCAN.PRG[120,5]
SCAN   PRG  12    AMS1:[100,3]

SCAN   PRG  13  650-353-035-656    AMS1:[110,5]

SCAN   PRG  21    AMS0:[120,5]

Grand total of 3 files in 46 disk blocks
```

BASE OPTION: The /B switch tells DIR to display the base (starting) disk address of the selected files. If you use DIR to see information about modules in memory (see Section 9.2.7, DIR and Special and Ersatz Devices), the /B switch tells DIR to display the base memory addresses of the modules.

The /B switch is a file switch: its placement on the command line affects which files it acts upon:

```
.DIR *.CMD,*.PRG/B
C      CMD  1    DSK0:[102,1]
P1     CMD  1
Total of 2 files in 2 blocks

LISP   PRG  33   4405    DSK0:[102,1]

Grand total of 3 files in 35 disk blocks
```

In the example above, the /B switch affects only those files selected by *.PRG.

NOBASE OPTION: The /NOB switch allows you to turn off the /B switch. For example, suppose that you want base disk addresses for every file specified on the command line but the last; follow the last filespec with a /NOB switch (e.g., .DIR/B *.PRG,*.CMD,FREE.MAC/NOB).

CONTIGUOUS OPTION: The /C switch allows you to determine which files listed in your directory are contiguous (that is, random files), and which are sequential files. (For information on sequential versus random files, see the manual "Introduction to AMOS.") When the /C switch is in effect, DIR places a "C" next to the extension of each contiguous file in the directory display:

```
.DIR/C [255,1]
NORTH   BAS    7                    DSK1:[255,1]
PAGE    MAC    21
BDIR    DAT  C 12
Total of 3 files in 40 blocks
```

In the example above, the file BDIR.DAT is marked as a contiguous file. The /C switch is an operation switch.

FULL OPTION: The /FULL switch gives you the full range of the DIR information options. /F is equivalent to specifying the switches: /HASH/BASE/CONTIGUOUS. /FULL is an operation switch:

```
.DIR/FULL
MILAGE  BAS    7     250-240-447-745      2320    DSK3:[200,4]
MILAGE  RUN    3     015-410-514-370      2330
DATA    WRK  C 13     533-501-341-776     13624
Total of 3 files in 33 blocks
```

9.2.7 DIR and Special and Ersatz Devices

DIR recognizes the ersatz devices (see Section 9.1.5, Ersatz Devices). So you can, for example, find out what files are in the BASIC Language Library account (DSK0:[7,6]) by typing:

```
.DIR BAS:
```

DIR also recognizes the special devices MEM: (user memory partition) and RES: (system memory). That means that you can use DIR to discover information about programs that are in your memory partition or in system memory. By using the /H switch (see above, HASH SWITCH), you can ask DIR to compute and display hashmark totals for the programs in memory:

```
.DIR MEM:LOG/H
LOG   PRG  1026    512-123-435-601      MEM:
Total of 1 file, 1026 bytes
```

NOTE: You can use DIR with the MEM: and RES: devices to perform many of the same functions as the MAP and SYSTEM commands, with the added benefit that you are able to take advantage of the advanced wildcarding abilities of DIR. The command:

```
.DIR MEM:CRM*.RUN
```

gives a listing of all of the compiled BASIC programs whose name begins with CRM that are currently in your memory partition. The command:

```
.DIR RES:*.DVR
```

lists all Driver programs currently in system memory. Use the /H switch to compute and display hashmark totals for the modules.

You may specify both special and regular devices within the same command:

```
.DIR/BASE [200,1],MEM:
F2DSK   PRG 4   2156        DSK0:[200,1]
PREDOC  TXT 20  4405
Total of 2 files in 24 blocks

HASH    PRG 1016    36640       MEM:
QDT     PRG 330     40642
INDEX   PRG 566     41366
NULL    PRG 1276    42466
Total of 4 files in 3188 bytes

Grand total of 6 files in 24 blocks and in 3188 bytes
```

The example above gives a directory display for the account DSK0:[200,1], and for the modules in your memory partition (MEM:).

9.2.8 DIR Error Messages

Below is a list of some of the error messages that you can see when using DIR. If you encounter an error message that is not mentioned here, refer to Appendix A, AMOS System Error Messages.

1. %No such files
   DIR couldn't find any files that matched your filespecs. Are you sure that you are searching the correct account? Check your spelling.

2. %Account does not exist - [p,pn]
   The indicated account does not exist. Are you sure that you've specified the correct device? Remember that an explicit device specification sets the default for the rest of the command line.

3. ?Specification error ^
   The command line was not in proper format. The ^ symbol points to the location in the command line that DIR does not understand:

```
.DIR AMS0::
?          ^ Specification error
```

4. ?More than one output specification

   You may only specify one listfile; that is, only one file specification may appear on the left of the equal sign when you use DIR to create a file that contains your directory listing:

```
.DIR LSTFIL.LST,DIRLST.LST=[110,23] ↵
?More than one output specification
```

5. %No file-oriented device corresponding to dev: is mounted

   You specified a device but did not include a unit number (e.g., AMS:). DIR tried to match all possible device units with your specification, but was unable to find any that were mounted.

```
.DIR AMS: ↵
%No file-oriented device corresponding to AMS: is mounted
```

   Check your spelling. If that's OK, try mounting the disk. If that doesn't work, check with your System Operator to make sure that the device has been defined in the system initialization command file, that the device has a driver program in area [1,6] of the System Disk, and that the device is file-structured.

6. ?Cannot find DSK0:SCNWLD.SYS[1,4] or MEM:SCNWLD.SYS

   All wildcard file commands need the SCNWLD program to process wildcard symbols in filespecs. If you see this message, check with your System Operator who will make sure that a valid copy of SCNWLD is installed on your system in the System Program Library Account (DSK0:[1,4]).

7. [CANNOT OPEN MEM:filespec - DEVICE DOES NOT EXIST]
   [CANNOT OPEN RES:filespec - DEVICE DOES NOT EXIST]

   You told DIR to compute hashmarks on modules in your memory partition or in system memory (e.g., .DIR/H RES: or .DIR/H MEM:), but the special devices MEM: and RES: have not been defined in the device table of the system initialization command file. Check with your System Operator.

```
.DIR/H RES:TRM.DVR ↵
TRM   DVR 262 [CANNOT OPEN MEM:TRM.DVR - DEVICE DOES NOT EXIST]
```

9.3 RENAMING FILES (RENAME)

You can change the names and extensions of disk files by using RENAME-- what is more interesting, is that you can change the names and extensions of whole groups of files at one time by using RENAME. Because RENAME is so powerful, be cautious while you are getting used to it; make sure that it will do what you think it is going to do.

COMMAND SYNTAX:

.RENAME {outfilespec}={infilespec1{...,infilespecN}}{/switch{/switch}}↵

COMMAND DEFAULTS:

The initial default infilespec is the account and device into which you are logged and a filename and extension of *.*.

The initial default outfilespec is ALL:[] and a filename and extension of *.*.

Initial default switch settings are /NODELETE and /NOQUERY.

SPECIAL FUNCTIONS:

You can use RENAME to rename memory modules by specifying the MEM: device (see Section 9.3.5, RENAME and Special and Ersatz Devices).

9.3.1 Renaming a File in Your Account

Let's say that you want to change the name of a file in your account from WRKFIL.TXT to SUBRTN.BAS. Type RENAME. Now specify the new name of the file and type an equal sign; then specify the file you want to rename. Hit RETURN:

```
.RENAME SUBRTN.BAS=WRKFIL.TXT↵
WRKFIL.TXT TO SUBRTN.BAS
Total of 1 file renamed
```

9.3.2 Renaming a File in an Account Other Than Your Own

You are allowed to rename files in accounts other than the one you are logged into:

```
.RENAME DOC1.TXT=FRED.TXT[110,4]↵
FRED.TXT[110,4] TO DOC1.TXT
Total of 1 file renamed
```

The example above renames file FRED.TXT (in account [110,4]) to DOC1.TXT. Note that the new file, DOC1.TXT, is still in account [110,4]. The files that you rename must be either in your own account or in an account that shares the project number of the account you are logged into. If you try to rename a file in an account that is not in your own project, you see:

```
.RENAME DCLARE=WRKPRG[100,1]↵
WRKPRG.LSP[100,1] to DCLARE.LSP
[CANNOT RENAME WRKPRG.LSP[100,1] - PROTECTION VIOLATION]
%No files renamed
```

The exception to this rule is that the System Operator may (when logged into the System Operator's Account, DSK0:[1,2]) rename files in any account regardless of its project number.

Since you cannot change the location of a file by using the RENAME command, you are not allowed to include an account specification in the RENAME outfilespec:

```
.RENAME ME[200,3]=TITLE[100,5]↵
RENAME TITLE.TXT[100,5] to ME.TXT[200,3]
?Device or [P,Pn] specifications on output are illegal
```

9.3.3 RENAME and Wildcard Symbols

Because RENAME is a wildcard file command, you can use it to perform some very sophisticated renaming functions. Wildcard symbols in infilespecs allow you to specify a group of files with one filespec (e.g., *.BAS specifies all BASIC programs in an account). A wildcarded infilespec, then, tells RENAME to rename ALL files selected by the filespec. Wildcard symbols in the outfilespec are treated a little differently.

As we saw in Section 9.1.3, Output File Specifications, the outfilespec modifies the selected infilespecs:

```
.RENAME *.TXF=JIM.TXT,GUIDE.TXT↵
JIM.TXT TO JIM.TXF
GUIDE.TXT TO GUIDE.TXF
Total of 2 files renamed
```

The outfilespec in the command above leaves the filename unfixed, and so RENAME leaves the names of the infilespecs unchanged.

If we were to use wildcard symbols in both the outfilespec and infilespec, we would be saying: "For all the files selected by the infilespecs, replace those portions of the infilespec that are fixed by the outfilespec." For example:

```
.RENAME OVTPAY.*=WRKPRG.*,WRKFIL.*↵
WRKPRG.BAS TO OVTPAY.BAS
WRKPRG.RUN TO OVTPAY.RUN
WRKFIL.TXT TO OVTPAY.TXT
WRKFIL.DAT TO OVTPAY.DAT
Total of 4 files renamed
```

In the example above, our filespecs included extensions of .* just so that

it would be easier to understand what was happening. However, remember that if you omit an extension, or both a filename and an extension from a filespec, RENAME uses the default of *. Therefore, the command:

```
.RENAME OVTPAY=WRKPRG,WRKFIL ↵
```

is exactly equivalent to the command in the example above.

9.3.4 Selecting RENAME Options

By including one or more of the following switches on the command line, you can select several RENAME options:

/QUERY	or	/Q	Requests confirmation to rename.
/NOQUERY	or	/NOQ	Renames without asking for confirmation.
/DELETE	or	/D	If file with new name already exists, deletes it before performing rename.
/NODELETE	or	/NOD	If file with new name already exists, does not perform rename.

You may place more than one switch on a command line if you precede each switch with a slash. For example:

```
.RENAME/Q/D *.OLD=*.NEW ↵
```

QUERY OPTION: The /Q switch tells RENAME to ask for confirmation before renaming a file. You can see where this might be convenient, if you're not completely sure of which files your RENAME command is going to affect.

The /Q switch is a file switch. To tell RENAME to ask for confirmation of all renamings, place the /Q switch directly after the command; to ask for confirmation of the renaming of specific files, follow those filespecs with the /Q.

When RENAME requests confirmation of a renaming, it tells you what it intends to do and displays a question mark. To tell RENAME to go ahead with the renaming, reply with an upper or lower case Y; to tell it not to perform that particular renaming, reply with an upper or lower case N. DO NOT hit a RETURN after your answer. After RENAME responds to each answer, it goes on to the next renaming operation:

```
.RENAME *.OLD=BOOK.TXT,*.NEW/Q ↵
BOOK.TXT TO BOOK.OLD
SICPAY.NEW TO SICPAY.OLD?Y ↵
VACPAY.NEW TO VACPAY.OLD?N ↵
OVTPAY.NEW TO OVTPAY.OLD?Y ↵
Total of 3 files renamed
```

(The example above asks for confirmation of the renaming of the files that match the *.NEW specification, since /Q follows that specification. If you should want to interrupt the RENAME command and return to AMOS command level, type a Control-C:

```
.RENAME LNDST*=MAP*/Q↵
MAP1.TXT TO LNDST1.TXT?Y↵
MAP1.BAK TO LNDST1.BAK?N↵
MAP2.TXT TO LNDST2.TXT?Y↵
MAP2.BAK TO LNDST2.BAK?N↵
MAP3.TXT TO LNDST3.TXT?^C
.
```

NOQUERY OPTION: The /NOQ switch turns off the /Q switch. For example, suppose you want RENAME to confirm every renaming except the last-- place a /NOQ at the end of the last file specification:

```
.RENAME/Q MUSIC=MNOTE,MDOC/NOQ↵
MNOTE.MAC TO MUSIC.MAC?Y↵
MNOTE.OBJ TO MUSIC.OBJ?Y↵
MNOTE.TXT TO MUSIC.TXT?N↵
MDOC.TXT TO MUSIC.TXT
MDOC.LST TO MUSIC.LST
Total of 4 files renamed
```

DELETE OPTION: RENAME cannot rename a file if a file of that name already exists in the account; to instruct RENAME to delete any existing, conflicting file before renaming a file, use the /D switch.

The /D switch is a file switch; it can affect either ALL infilespecs or only specific infilespecs, depending on where you place it on the command line.

NODELETE OPTION: The /NODELETE switch turns off the /DELETE switch. /NOD is a file switch.

9.3.5 RENAME and Special and Ersatz Devices

You may rename memory modules (copies of disk files that have been loaded into your area of memory) by specifying the MEM: device:

```
.RENAME UNBILL=MEM:CUSID↵
MEM:CUSID.PRG TO UNBILL.PRG
Total 1 file renamed
```

The example above changes the name of the module CUSID.PRG in your memory area to UNBILL.PRG.

RENAME also understands the ersatz devices.

```
.RENAME INQUIR=BAS:WRKFIL
DSKU:WRKFIL.TXT[7,6] TO INQUIR.TXT
Total of 1 file renamed
```

The example above tells RENAME to rename the file WRKFIL.* in the BASIC Program Library (DSKU:[7,6]). (In the example above, you must be logged into an account in Project 7 to avoid a Protection Violation error.)

9.3.6 RENAME Error Messages

Below are some common error messages you are likely to see when you use RENAME. Refer to Appendix A, AMOS System Error Messages, if you encounter an error not discussed here.

1. %No files renamed
   Because of one reason or another, RENAME was not able to rename any files. Perhaps it wasn't able to find any files that matched your wildcard filespecs. Make sure that you entered the filespecs on the command line in the proper order (i.e., outfilespec to the left of the equal sign; infilespec to the right).

2. %No file-oriented device corresponding to dev: is mounted
   You specified a device, but did not include a unit number (e.g., AMS:). RENAME tried to match all possible units with your specification, but did not find any that were mounted. Check your spelling; if that looks OK, try mounting the disk (use the MOUNT command). If you still have no success, check with System Operator, who will make sure that the device referred to by the error message is a valid system device and that a device driver program for that device exists.

3. %Account does not exist - [p,pn]
   The indicated account does not exist. Check your typing; if that's OK, try using the DIR command to see a directory for that account. If you still have no luck, check with the System Operator.

4. ?Specification error ^
   RENAME does not understand the format of your command line; the ^ symbol points to the confusing element of the line. Retype the command.

5. ?More than one output specification
   You cannot rename a file to two names; use only one outfilespec.

6. ?Device or [p,pn] specifications on output are illegal
   You cannot change the location of a file by using RENAME; do not include an account or device specification in a RENAME outfilespec.

7. ?Cannot find DSK0:SCNWLD.SYS[1,4] or MEM:SCNWLD.SYS
   All wildcard file commands need the SCNWLD.SYS program to process wildcard symbols in filespecs. Check with your System Operator, who will make sure that a valid copy of SCNWLD.SYS is installed on your system in account DSK0:[1,4].

8. ?Missing output specification
   You must give an output specification so that RENAME knows what to use as the new names for your input files.

9. [CANNOT RENAME infilespec - FILE ALREADY EXISTS]
   No two files within an account can share the same name and extension; if you want RENAME to delete existing files when it sees a renaming conflict, use the /D switch.

9.4 ERASING FILES (ERASE)

You can erase files on the disk by using the ERASE command. Because ERASE is a wildcard file command, you can erase entire groups of files at a time with one wildcard file specification. You may erase files from your own account or from accounts within your project.

COMMAND SYNTAX:

.ERASE {outfilespec=}infilespec1{,....infilespecN}{/switch{/switch}}

COMMAND DEFAULTS:

The initial outfilespec default is *.* and the account and device you are currently logged into.

The initial infilespec default is the account and device you are currently logged into and a filename of *. The default extension is an empty extension.

The default switch is /NOQUERY.

SPECIAL FUNCTIONS:

You can use the ERASE command to delete memory modules (see Section 9.4.6, ERASE and Special and Ersatz Devices).

9.4.1 Erasing Files From Your Account

Type ERASE followed by the specification for the file you want to erase; then hit RETURN:

```
.ERASE SEED.MAC↵
SEED.MAC
Total of 1 file deleted, 6 disk blocks freed
```

You may specify more than one file to be erased by including more than one infilespec on the command line separated by commas:

```
.ERASE CONTRL.LSP,REMOTE.PRG↵
CONTRL.LSP
REMOTE.PRG
Total of 2 files deleted, 23 disk blocks freed
```

9.4.2 Erasing Files From Accounts Other Than Your Own

You can erase files from your own account or from other accounts within your own project. To erase files from other accounts, include the device and account specification for those accounts as part of your infilespecs:

```
.ERASE DSK0:VUE.HLP[230,0],BASIC.HLP[230,5],SWITCH.HLP[230,10]↵
DSK0:VUE.HLP[230,0]
DSK0:BASIC.HLP[230,5]
DSK0:SWITCH.HLP[230,10]
Total of 3 files deleted, 134 disk blocks freed
```

(NOTE: Remember that ERASE is a wildcard file command; the first infilespec in the command line above sets the default device to DSK0: for the entire list of infilespecs.)

If you try to erase a file from an account that does not share the project number of your own PPN, you see:

```
[CANNOT DELETE filespec - PROTECTION VIOLATION]
```

where filespec is the input file you tried to erase. The exception to this rule is that the System Operator may (if logged into the System Operator Account, DSK0:[1,2]) erase files from all accounts regardless of project numbers.

9.4.3 ERASE and Wildcard Symbols

By using wildcard symbols you can select a group of files to be erased:

```
.ERASE *.BAK↵
OEITEM.BAK
WINGS.BAK
BRYCE.BAK
CANCEL.BAK
Total of 4 files deleted, 23 disk blocks freed
```

Remember that you can use wildcard PPN and device symbols:

```
.ERASE DSK:[200,*]A*.TXT
AUINTR.TXT[200,3]
APRIL.TXT
DSK1:APPA.TXT[200,6]
DSK1:APPB.TXT[200,6]
DSK2:AUINTR.TXT[200,6]
Total of 5 files deleted, 33 disk blocks freed
```

The example above erases every .TXT file whose name begins with A which exists in an account in Project 200 on all units of device DSK:.

9.4.4 Using an Outfilespec

ERASE offers a unique feature that allows you to further select among the input files specified by the infilespecs. When you use an outfilespec, you tell ERASE: "For all the files selected by the infilespecs, if a corresponding file selected by the outfilespec ALSO exists, erase the output file." For example:

```
.ERASE *.BAK=*.MAC
EIGHT.BAK
HEADER.BAK
RELEAS.BAK
Total of 3 files deleted, 123 disk blocks freed
```

tells ERASE "Erase all .BAK files in my account if a .MAC file of the same name also exists in the account."

```
.ERASE =*.TXT[110,6]
MPROD.TXT
UNIV.TXT
SOL.TXT
Total of 3 files deleted, 54 disk blocks freed
```

Because the outfilespec default is the device and account you are currently logged into (and a filename and extension of *.*), the command above tells ERASE "Erase all .TXT files in the account I am currently logged into if those files ALSO exist in account [110,6]." (If an equal sign appears in an ERASE command line, ERASE realizes that you are specifying an outfilespec. If nothing appears to the left of the equal sign but the ERASE command, ERASE uses the default outfilespec.)

9.4.5 Selecting ERASE Options

You may select among the following ERASE options by including the proper switch on the ERASE command line (both switches are file switches):

/QUERY or /Q	Asks for confirmation before erasing each file.
/NOQUERY or /NOQ	Performs erasures without asking for confirmation.

The initial default switch is /NOQUERY. You may combine /Q and /NOQ switches on a command line. For example:

```
.ERASE/QUERY *.BAK, *.TXT,/NOQUERY TITLE.*,TECH.DOC,TETRA.BAS
IMAGE.BAK?Y
COMF.BAK?N
STAT.TXT?N
INIT.TXT?Y
TITLE.BAS
TITLE.RUN
TECH.DOC
TETRA.BAS
Total of 6 files deleted, 120 disk blocks freed
```

9.4.6 ERASE and Special and Ersatz Devices

You may use ERASE to delete memory modules by specifying the MEM: device:

```
.ERASE MEM:LOG.*,QDT.*
LOG.PRG
QDT.PRG
Total of 2 files deleted, 1406 bytes of memory freed
```

ERASE also understands the ersatz devices. You can, for example, erase files in the BASIC Language Library, DSK0:[7,6], by specifying the BAS: device:

```
.ERASE BAS:DRILL.*
DSK0:[7,6]DRILL.SBR
DSK0:[7,6]DRILL.BAS
DSK0:[7,6]DRILL.RUN
Total of 3 files deleted, 123 disk blocks freed
```

(NOTE: In the example above, you must be logged into an account in Project 7 to avoid a Protection Violation error.)

9.4.7 ERASE Error Messages

Below are some of the more common error messages you might see when using ERASE. For a list of other system error messages, see Appendix A, AMOS System Error Messages.

1. %No files deleted
   ERASE was not able to find the files you specified. Make sure that you included account and device specifications if they are needed.

2. %Account does not exist - [p,pn]
   The account indicated does not exist. Remember to include a device specification if needed. Try using the DIR command for that account, using the ALL: device specification to locate it. Check with your System Operator if you still have no success.

3. %No file-oriented device corresponding to dev: is mounted
   You specified a device, but did not include a unit specification (e.g., STD:). ERASE was not able to find a mounted unit matching that specification. Use the MOUNT command to make sure that the device is mounted. Check with your System Operator to see if the device you specified is a valid system device.

4. ?Specification error ^
   ERASE did not understand the format of your command line. The ^ symbol points to the confusing element of the line. Retype the command line.

5. ?More than one output specification
   You may not specify more than one output file. For example, ERASE *.BAK,*.TXT=*.MAC, is an illegal command.

6. ?Cannot find DSK0:SCNWLD.SYS[1,4] or MEM:SCNWLD.SYS
   All wildcard file commands need SCNWLD.SYS to be able to process wildcard filespecs; check with your System Operator who will make sure that your system has a valid copy of SCNWLD.SYS in account DSK0:[1,4].

9.5 COPYING FILES (COPY)

To transfer files from one account to another, use the COPY command to make a copy of the original file in the new account. You may also make a copy of a file within its own account if you assign a unique name to that copy. No files within an account can have the same name.

Unless you are creating a file in your own account, you may only transfer files between accounts whose PPNs have the same project number (e.g., you may copy files between accounts [300,1] and [300,2], but not between [120,5] and [100,4]). You may copy files into the account you are currently logged into from any account on the system.

COMMAND SYNTAX:

.COPY {outfilespec}={infilespec1{,....infilespecN}}{/switch{/switch}} ↵

COMMAND DEFAULTS:

The initial default infilespec is a filename and extension of *.*, and the current default account and device.

The initial default outfilespec is a filename and extension of *.* and the current default account and device. (If the System Operator uses COPY from the System Operator's Account, DSKÜ:[1,2], the default outfilespec is a filename and extension of *.*, the device you are logged into and the wildcard account, [].)

The initial default switches are: /DELETE/NOQUERY/NOPACK.

SPECIAL FUNCTIONS:

You may use COPY to send files to a terminal or printer by specifying the TRM:device. You may use COPY to load files into memory, and to save memory modules as disk files. (See Section 9.5.7, COPY and Special and Ersatz Devices.)

The System Operator (if logged into the System Operator's Account, DSKÜ:[1,2]) may use COPY to backup all accounts (or selected accounts) from one device onto another.

NOTE: Because of the default infilespecs and outfilespecs, almost every combination of filespecs that you give COPY causes something to happen. For example:

.COPY/PACK = ↵

tells COPY to make copies of all files in the account you are logged into, and to place those copies in the account. This example replaces all files in your account with copies of themselves-- not usually a very useful thing to do and VERY dangerous if other users are running on the system at the time of the command. So, be careful when you use COPY.

9.5.1 Copying a File in Your Own Account

To make a copy of a file within your own account, type COPY, the name you want to assign to the new copy, an equal sign and the name of the file you want to copy:

```
.COPY ARCHVE.BAS=WRKFIL.BAS ↵
WRKFIL.BAS to ARCHVE.BAS
Total of 1 file transferred
```

Your account now contains the original file and a copy of that file.

9.5.2 Copying a File into Your Own Account from Another Account

To copy a file into your own account from another account, type COPY, the name you want to assign to the copy, an equal sign and the specification of the file you want to copy (including the specification of the account in which the file is to be found); then type a RETURN:

```
.COPY NEWLSP.LSP=DSK1:RESET.LSP[200,3] ↵
DSK1:RESET.LSP[200,3] to NEWLSP.LSP
Total of 1 file transferred
```

(NOTE: In both examples above, we created a copy of a file in our own account, and so did not include an account specification in the outfilespec.)

9.5.3 Copying a File into An Account Other Than Your Own

To create a file in an account that is not your own, the account you are logged into and the account you are transferring to MUST be within the same project (that is, their PPNs must share the same project number):

```
.COPY DSK1:ALPHA.MAC[300,5]=DSK0:SCRTCH.MAC[300,7] ↵
DSK0:SCRTCH.MAC[300,7] to ALPHA.MAC[300,5]
Total of 1 file transferred
```

9.5.4 COPY and Wildcard Symbols

Because COPY is a wildcard file command, you can use it to transfer copies of entire sets of files at one time. For example:

```
.COPY DSK1:[103,4]=*.PRG ↵
READ.PRG to DSK1:READ.PRG[103,4]
SYNTX.PRG to DSK1:SYNTX.PRG[103,4]
DSKWRT.PRG to DSK1:DSKWRT.PRG[103,4]
Total of 3 files transferred
```

The example above tells COPY to transfer copies of all .PRG files in the account and device you are currently logged into over to account DSK1:[103,4]. NOTE: Because the outfilespec does not contain a filename or extension, COPY assigns the names selected by the infilespec (*.PRG) to the new files.

You can modify the names selected by the infilespec by including a filename and/or extension in the outfilespec.

```
.COPY *.TXF=*.TXT↵
CHPTR1.TXT to CHPTR1.TXF
PRFACE.TXT to PRFACE.TXF
INDEX.TXT to INDEX.TXF
Total of 3 files transferred
```

Because no two files in an account may have the same name and extension, you may not usually supply more than one infilespec. However, if your outfilespec is wildcarded, you CAN specify more than one input file:

```
.COPY *.BAS=*.BS1,CREATE.BS2,*.BS3↵
WDANGL.BS1 to WDANGL.BAS
SCANNR.BS1 to SCANNR.BAS
CREATE.BS2 to CREATE.BAS
FACTOR.BS3 to FACTOR.BAS
Total of 4 files transferred
```

By including wildcard symbols in device and account specifications, you can transfer copies of files from more than one account:

```
.COPY DSK1:[]=DSK0:*.OBJ[100,*]↵
DSK0:QUAD.OBJ[100,1] to DSK1:QUAD.OBJ[100,1]
DSK0:FILE1.OBJ[100,1] to DSK1:FILE1.OBJ[100,1]
DSK0:GEOM.OBJ[100,5] to DSK1:GEOM.OBJ[100,5]
DSK0:EUCLID.OBJ[100,10] to DSK1:EUCLID.OBJ[100,10]
Total of 4 files transferred
```

NOTE: Be careful to include the wildcard PPN symbol [], when copying files from a set of accounts into the corresponding accounts on another device. If we had not included the [] symbol in the outfilespec of the example above, COPY would have transferred the copies of all the files in the accounts in Project 100 over to the single account we are currently logged into.

9.5.5 COPY and the System Operator

The System Operator (when logged into the System Operator's Account, DSK0:[1,2]) has special privileges when using the COPY command:

1. The System Operator may copy files from one account to another regardless of project number.

2. The outfilespec default for a COPY command used from DSK0:[1,2] is a filename and extension of *.*, the device the System Operator is currently logged into and an account specification of []. This allows the System Operator to backup groups of accounts from one device to another without having to worry about all the files being copied into DSK0:[1,2]:

```
.COPY =DSK1:[]↵
```

The command above tells COPY to copy files from all accounts on DSK1: over to the corresponding accounts on the device the System Operator is logged into.

3. If the System Operator tries to copy files over to a nonexistent account, COPY creates the account and then copies the files over. The new account has the same password (if any) that belonged to the original, source account (regardless of whether the new account has the same PPN as the source account).

9.5.6 Selecting COPY Options

By including one or more of the following switches on the command line, you can select several COPY options:

/QUERY or /Q	Requests confirmation to transfer.
/NOQUERY or /NOQ	Does not ask for confirmation for transfers.
/DELETE or /D	Copies to existing files.
/NODELETE or /NOD	Does not copy to existing files.
/PACK or /P	Allows user to copy file over to itself.
/NOPACK or /NOP	Prevents user from copying file over to itself.

You may place more than one switch on a command line if you precede each switch with a slash:

```
.COPY/NODELETE/QUERY =*.BAS[100,2]↵
```

QUERY OPTION: The /Q switch tells COPY to ask for confirmation before transferring each file copy. Until you get used to using COPY, it might be a good idea to always include the /Q switch after the COPY command on the COPY command line:

```
.COPY/Q CRIDX[200,1]=CUSTMR
CUSTMR.BAK to CRIDX.BAK[200,1]?N
CUSTMR.BAS to CRIDX.BAS[200,1]?Y
CUSTMR.RUN to CRIDX.RUN[200,1]?Y
CUSTMR.TXT to CRIDX.TXT[200,1]?Y
Total of 3 files transferred
```

Respond to each question mark with a Y (for Yes) or an N (for No), and do NOT type a RETURN after your answer.

The /Q switch is a file switch; if you place it before a filespec (e.g., COPY *.TXT=CALNDR.TXF,/Q *.LST,*.TNX) it becomes the default switch for the rest of the command line. If you place the switch directly after a filespec (e.g., COPY *.BAS=*.BAS1/Q,*.BAS2), it affects only the files selected by that filespec.

NOQUERY OPTION: The /NOQ switch turns off the /Q switch. For example:

```
.COPY/Q *.MAC=*.MC1,*.MC2,/NOQ *.MC3,*.MCR
HYDRA.MC1 to HYDRA.MAC?Y
DUAL.MC1 to DUAL.MAC?Y
FILE2.MC2 to FILE2.MAC?N
STRESS.MC3 to STRESS.MAC
STRS2.MC3 to STRS2.MAC
BRIDGE.MCR to BRIDGE.MAC
Total of 5 files transferred
```

NODELETE OPTION: If you ask COPY to create a new file with the name that belongs to an existing file in the account, COPY usually simply replaces that old file with the new copy. If you do not want COPY to delete existing files, use the /NOD OPTION:

```
.COPY/NODELETE =AMS1:*.RUN[110,3]
AMS1:ICE.RUN[110,3] to ICE.RUN
AMS1:CHEM.RUN[110,3] to CHEM.RUN
%Not copied - Destination file already exists.
%No files transferred
```

The /NOD switch is a file switch.

DELETE OPTION: The /D switch turns off the /NODELETE switch. For example:

```
.COPY *.TXT=/NOD *.LST,*.TXF,/D *.TNF
```

PACK OPTION: The /P switch allows you to copy a file over onto itself. No two files with the same name and extension may exist within the same account. If you ask COPY to create a copy of a file within an account and you want the copy to have the same name as the original, COPY can delete the original file and replace it with the copy. This is a VERY dangerous procedure if another job is running concurrently on the system; your file can be lost or scrambled if another user accesses the disk at the same time. COPY does not allow you to do this kind of thing unless you include the /PACK switch on the command line:

```
.COPY/PACK =*.TXT↵
VALLEY.TXT to VALLEY.TXT
DEVICE.TXT to DEVICE.TXT
RANDOM.TXT to RANDOM.TXT
Total of 3 files transferred
```

Now, why would you want to copy a file over onto itself? Copying files over onto themselves compresses those files on the disk (that is, it "packs" the area of the disk taken up by those files); this uses the space on the disk more efficiently, and allows you to have more contiguous records free on the disk. (The System Operator can pack the disk more thoroughly by logging into the System Operator's Account and entering: .COPY/PACK =[]. The System Operator may also use the DSKPAK command to pack contiguous files.)

NOPACK OPTION: The /NOP switch turns off the /P switch. /NOP is a file switch.

9.5.7 COPY and Special and Ersatz Devices

Using the ersatz and special devices, you can perform a wide variety of system functions with COPY. For example:

```
.COPY TRM:={infilespec{,...infilespecN}}↵
```

performs much the same function as the TYPE command, but it allows you to specify more than one infilespec and to use wildcard symbols. It sends a copy of the files selected to your terminal display.

```
.COPY TRM:=*.BAS↵
```

sends a copy of all BASIC files in your account to your terminal display. (To display the contents of files on a particular terminal or printer other than your own terminal, specify the name of the terminal after the TRM: specification: .COPY TRM:QUME=FILEA.)

```
.COPY MEM:={infilespec{,...infilespecN}}↵
```

performs the same function as the LOAD command; it loads a copy of the files selected into your memory partition area.

```
.COPY =MEM:{infilespec{,...infilespecN}}↵
```

performs the same function as the SAVE command; it saves a copy of the memory modules selected as disk files in your account. (See Section 11.4, Saving Memory Modules as Files-- SAVE, for information on memory modules.)

You can also use the ersatz devices as a shorthand way to identify disk accounts:

```
.COPY =BOX:CATLOG↵
CATLOG.BAK[7,1] to CATLOG.BAK
CATLOG.HLP[7,1] to CATLOG.HLP
Total of 2 files transferred
```

(In the example above, we must be logged into an account in Project 7 to avoid a Protection Violation error.)

9.5.8 COPY Error Messages

Below are many of the error messages you may see when using COPY. If you encounter an error not discussed here, refer to Appendix A, AMOS System Error Messages, for a more complete listing of system error messages.

1. %No files transferred
   COPY could not find any files that matched your infilespecs, or some other problem prevented COPY from making file copies.

2. %Attempt to copy file to self--
   % Make sure no other jobs are running
   % Include the switch /PACK in your copy command

   You tried to create a copy of a file with the same name as the original; if COPY were to allow you to do so, it would delete the original file and replace it with the copy-- an extremely dangerous procedure if another job is running on the system at the time of the command. The purpose of this message is to prevent you from accidentally copying a file to itself; if you want to do so, use the /PACK switch, and make sure that no other job is running on the system.

3. %No file-oriented device corresponding to dev: is mounted
   You specified a device but did not include a unit number. After searching for all possible units of that device, COPY failed to find a mounted unit matching your specification:

   ```
   .COPY =AMS:SCAN.PRG↵
   %No file-oriented device corresponding to AMS: is mounted
   ```

   After you've checked your spelling, try mounting the disk you want to use. If you still have no success, check with the System Operator for help; he or she will make sure that the device is a legal, defined system device.

4. %Not copied - Destination file already exists
   You tried to create a new file with the name belonging to an existing file while the /NODELETE switch was in effect. If you want to replace existing files with new copies, do not use the /NODELETE switch.

5. %Account does not exist - [p,pn]
   You tried to access an account that does not exist:

```
.COPY =*.BAS[230,1]↵
%Account does not exist - [230,1]
```

   Check your typing; if that looks OK, try using DIR to see if the account does indeed exist.

6. ?Specification error ^
   Your command line is not in proper format. The ^ symbol points to the confusing item on the command line. Check the command syntax and retype the line.

7. ?More than one output specification
   You may only specify one outfilespec. (You cannot copy an input file to more than one output file.)

8. ?Missing output specification
   You forgot to type the equal sign on the command line.

9. [CANNOT OPEN filespec - PROTECTION VIOLATION]
   You tried to create a file in an account that does not share the same project number as the file you are currently logged into.

10. ?Device full
    There is no more room on the disk. Try copying to another device.

11. ?You are not logged in under [1,2], can't create [p,pn]
    Only if you are logged into the System Operator's Account, [1,2], can you use COPY to create a nonexistent account.

12. ?Output MFD is full
    You tried to create a new account by using the COPY command while logged into [1,2]. The Master File Directory on the device was full, so COPY wasn't able to create the account. (Each device has a limit of 64 accounts.) Create the account on another device.

13. %Random files can not be transferred to MEM:
    You may only copy sequential files into your memory partition.

14. ?Files may not be transferred to RES:
    You may not add programs to system memory after the system is up and running. (The System Operator may add programs to system memory by modifying the system initialization procedure.)

15. ?Cannot find DSK0:SCNWLD.SYS[1,4] or MEM:SCNWLD.SYS
    The COPY command needs the SCNWLD program to process wildcard filespecs. Check with your System Operator for help; he or she will make sure that a valid copy of SCNWLD.SYS exists in account DSK0:[1,4].

9.6 PRINTING A FILE (PRINT)

We have already mentioned the use of the COPY command for sending a copy of a file to a printer or other terminal (see Section 9.5.7, Copy and Special and Ersatz Devices). However, a more sophisticated mechanism exists for performing the same kind of function: the PRINT command. Before beginning to discuss the actual syntax and use of the PRINT command, we'd like to introduce you to spoolers and queues.

A printer is not a sharable output device; that is, only one user may access the device at a time. (Obviously, chaos would result if the system were to allow two users to print their files at the same time on the same printer.) A special kind of program (called a spooler) allows you to put in a request for the use of an occupied system resource so that when the resource (in this case, a printer) again becomes free, it will attend to your task. The PRINT command is a line printer spooler; it stacks up (or "spools") requests for the use of the system printers until such time as a printer becomes available for use.

You may use the PRINT command to enter your request for the use of a printer into the waiting line (called a queue); you may either select the queue of a specific printer, or you may allow the system to choose a printer for you (in which case it will place your request into the queue of the printer that has the fewest number of disk blocks waiting to be printed). When a printer becomes available, the request at the head of the queue is acted upon, and the spooler prints the file specified by that request. The queue operates on a "first-come-first-served" basis; that is, files are printed in the same order as their corresponding requests in the queue.

COMMAND SYNTAX:

.PRINT {printerspec=}{infilespec1{,...infilespecN}{/switch{/switch}}}

COMMAND DEFAULTS:

The default infilespec is the account and device you are logged into, an extension of .LST and a null filename.

The default printspec is that printer with the least number of blocks currently in the printer queue.

The default switches are set by the System Operator when he or she initially sets up the line printer spooler to run on your particular system; check with the System Operator for a list of the switch defaults.

9.6.1 Sending a File to a Printer

To send a file to a printer, type PRINT, the specification of the file you want to print, and a RETURN:

```
.PRINT ARCTGT.TXT↵
```

If only one printer has been set up for spooler use, the print request will enter that printer's queue; if you have more than one printer defined on the system, your request will enter the queue of the printer that has the fewest number of file blocks waiting to be printed.

If you want to send a file to a particular printer, type PRINT, the name of the printer, an equal sign and the name of the file:

```
.PRINT QUME=TEACHR.BAS↵
TEACHR.BAS
Total of 1 file (23 blocks) in printer request
```

(See Section 9.6.3, below, to see how to find out the names of the printers that have been defined on the system.)

9.6.2 PRINT and Wildcard Symbols

You may use the standard wildcard file command filespec wildcards when specifying input files to the PRINT command. For example:

```
.PRINT *.BAS↵
RENWRT.BAS
RANDM.BAS
NUMGEN.BAS
Total of 3 files (127 blocks) in printer request
```

The command above prints all BASIC files in your account.

9.6.3 Finding Out Information About the Printer Queues

You may use the PRINT command to see what printers have been defined for use with the printer spooler, and to find out information about the printer requests waiting in the queues. Type PRINT followed by a RETURN:

```
.PRINT↵
```

You see something like this:

```
HYTYPE          Form: NORMAL    Total of 11 blocks

LEASE.LST       Copies: 2       Form: NORMAL    *       2 blocks remaining
RENTER.TXT      Copies: 1       Form: NORMAL            9 blocks

QUME    The queue is empty
```

The display above gives us information on the two printers (named HYTYPE and QUME) that have been set up by the System Operator for use with the spooler. First we see information about the queue of the printer HYTYPE: 1. HYTYPE (the name of the printer); 2. Form: NORMAL (the type of form mounted on the printer); and, 3. Total of 11 blocks (the number of disk blocks in the queue waiting to be printed).

The next two lines of the display tell us what files are waiting to be printed, how many copies of each will be printed and the kinds of forms on which they are supposed to be printed. You also see the number of disk blocks per file to be printed. An asterisk marks that file which is currently being printed.

The last line of the display above begins the information about the queue of the printer QUME. In this case, the queue has no files waiting to be printed:

```
The queue is empty
```

If the queue did contain any print requests, you would see a display similar to the one above for the HYTYPE queue.

9.6.4 Setting Printer Forms

Every printer used by the spooler has a type of form that is associated with it. (This assignment is done by the System Operator at the time of system initialization.) For example, if you usually print checks on one printer, the type of form that should be mounted on that printer might be named CHECKS. When you type PRINT and a RETURN (as in the section above), you see the kind of form associated with each printer.

You may specify the kind of form on which you want your listing printed (see the FORMS OPTION, below). In fact, if the form mounted on the printer is NOT named NORMAL, you MUST include a form specification in your print request.

So that you don't accidentally print a report on an invoice form or vice versa, PRINT checks the form-type you specify against the type of form that is supposed to be mounted on the printer. If the two do not match, PRINT informs the spooler Operator job (who is selected by the System Operator at the time of system initialization) that the forms need to be changed on the specified printer. For example, suppose that a user on the system specifies that a list of customer names be printed on two-part forms on printer PRNTR6, but only regular paper is mounted on that printer. The spooler

Operator sees this message on his or her terminal display:

```
; LPTSPL - Please mount 2PART on PRNTR6
```

The message specifies both the printer on which the forms must be changed and the type of forms that must be mounted. This message appears once a minute on the spooler Operator's terminal until the Operator changes the forms; then the print requests waiting in the queue can be attended to.

When you change the forms on a printer, tell the PRINT command that you have done so by using the SET command:

```
.SET FORMS printer-name form-name↵
```

where you specify the printer on which you have changed the forms and the name of the form you have mounted. For example:

```
.SET FORMS PRNTR6 2PART↵
```

The form-name may be any name you choose from one to six characters. To avoid confusion, however, everyone on the system should use a standard set of form names (e.g., CHECKS, NORMAL, 2PART, INVCE, etc.)

9.6.5 Selecting PRINT Options

You may choose one or more of the options below by including the appropriate switches on the PRINT command line. The default switch settings are chosen by the System Operator at the time of system initialization; check with the System Operator for those defaults.

/COPIES:n	or /C:n	Prints the number of copies requested.
/KILL	or /K	Removes the specified files from either the queue of a specific printer or from the queue of any printer.
/DELETE	or /D	Deletes the specified files after printing them.
/NODELETE	or /NOD	Does not delete the specified files after printing them; the initial default setting.
/BANNER	or /B	Prints a banner page preceding each listing.
/NOBANNER	or /NOB	Does not print a banner page preceding each listing.
/FORMS:	or /FO:	Checks to make sure that the specified forms are mounted on the printer. (If you don't use /FO:, PRINT assumes /FORMS:NORMAL.)

/HEADER	or /H	Print a page header at the top of every page of the listing.
/NOHEADER	or /NOH	Don't print page headers.
/FORMFEED	or /FF	Print a final form feed at the end of each listing.
/NOFORMFEED	or /NOF	Don't print a final form feed at the end of each listing.
/LPP:n		Maximum number of lines to print on each page.
/WIDTH:n	or /WI:n	The page width (in characters). Used for headers only.
/WAIT	or /WA	Tells PRINT to reinsert print requests into the queue as PRINT finishes processing earlier requests, rather than to discard them if the queue is full.

You may specify more than one switch on a command line, as long as you precede each switch with a slash:

```
.PRINT TETRA.*/COPIES:2/FORMS:NORMAL/BANNER↵
```

COPIES OPTION: To tell PRINT to make more than one copy of your listing, include the /C:n switch on your command line. /C:n is a file switch; it can affect all files specified on the command line or only specific files, depending on where you place it on the command line:

```
.PRINT STEPPR.MAC/C:2,*.BAS↵
```

tells PRINT to make two copies of STEPPR.MAC, but only one copy each of the BASIC files in your account. The command line:

```
.PRINT/C:2 STEPPR.MAC,*.BAS↵
```

tells PRINT to make two copies of STEPPR.MAC and all of the BASIC files in your account.

KILL OPTION: Use the /K switch to remove a print request from a printer queue. You may remove the request from the queue of a specific printer by specifying the name of that printer.

```
.PRINT TERM6=NEWFIL.TXT/K↵
NEWFIL.TXT
Total of 1 file killed
```

or by not specifying a printer, you may tell PRINT to search the queues of all printers, and to remove the specified file from the queue in which it is found.

```
.PRINT NEWFIL.TXT,CRESR.LST ↵
NEWFIL.TXT
CRESR.LST
Total of 2 files killed
```

The Filespecs that you specify when using the /K switch may NOT contain wildcards. /KILL is an operation switch; it kills ALL files selected by the Filespecs on the command line. You must enter one /K switch for each print request in the queue, even if two requests are for the same file.

DELETE OPTION: You may ask PRINT to erase a file after it prints it by including the /D switch on your command line. /DELETE is a file switch.

NODELETE OPTION: The /NODELETE switch turns off the /DELETE switch. /NOD is the default switch setting and is a file switch.

BANNER OPTION: By including the /BANNER switch on your command line, you may ask PRINT to print a special identification page preceding each listing. The banner page gives the following information:

1. The banner page identifies the listing by displaying in large, block letters the name and extension of the file printed.

2. At the bottom of the banner page, PRINT tells you the version number of the line printer spooler program and the name of the printer on which the listing was made:

   ```
   LPTSPL Version 3.0 running on LPT0
   ```

3. The next line of data gives the full specification of the file that was printed and the date on which it was printed. (Use the DATE command to set the system date.)

   ```
   File DSK0:SYSLPT.INI[1,4] printed on 4/1/79
   ```

4. The third line tells you the name of the form on which the listing was printed:

   ```
   Forms: NORMAL
   ```

5. If the /DELETE option is in effect, the banner page also displays this message:

   ```
   File will be deleted after printing
   ```

   and if you are printing more than one copy of a file, you see a message that tells you which copy this listing is:

   ```
   Copy 1 of 3
   ```

The /BANNER switch is a file switch.

NOBANNER OPTION: The /NOBANNER switch turns off the /BANNER switch. Check with the System Operator to see which switch is the initial default. /NOB is a file switch.

FORMS OPTION: The /FORMS:n switch allows you to specify the type of form on which you want your listing printed. The default form is NORMAL (the name usually assigned to the most regularly mounted form on a printer). PRINT checks to make sure that the form you specify is the form associated with the printer you want to use; if it is not, the spooler Operator sees a message on his or her terminal requesting that the form you specified be mounted on the printer.

Since the default form is NORMAL, not using the /FO:n switch is the same as specifying /FORMS:NORMAL. If the printer you are using does not have the form NORMAL mounted, you MUST use the /FO:n switch to specify the type of form that is mounted on that particular printer.

HEADER OPTION: The /HEADER switch allows you to print a header at the top of every page of the listing. Page headers give the name of the file being printed, the date, and the current page number. /HEADER is a file switch.

NOHEADER OPTION: The /NOHEADER switch turns off the /HEADER switch. Check with the System Operator to see which option is the initial default. /NOHEADER is a file switch.

FORMFEED OPTION: The /FORMFEED switch prints a final form feed at the bottom of each listing. It is a file switch.

NOFORMFEED OPTION: THE /NOFORMFEED switch does not print a final form feed at the end if each listing. It is a file switch.

LPP OPTION: The /LPP switch indicates the number of lines to print on each page (Lines Per Page). If /HEADER is set, PRINT prints a form feed when it outputs a full page and then it prints a page header. /LPP:n is a file switch.

WIDTH OPTION: The /WIDTH switch determines the page width (in characters). PRINT only uses this value in printing page headers. The /WIDTH value must be between 80 and 132, inclusive. /WIDTH:n is a file switch.

WAIT OPTION: The /WAIT switch determines the operation of the PRINT command when you try to enter more print requests into a printer queue than PRINT can handle. Normally, PRINT discards the extra print requests. /WAIT tells PRINT not to discard the extra print requests, but to reinsert them into the queue as PRINT finishes processing earlier requests. This option ties up your terminal while it waits for room to be made in the queue. /WAIT is an operation switch.

9.6.6 PRINT and Ersatz Devices

You may include the ersatz device specifications in your PRINT command line. For example:

```
.PRINT BAS:*.TXT↵
DSK0:SBRDOC.TXT[7,6]
DSK0:BASHLP.TXT[7,6]
Total of 2 files (34 blocks) in printer request
```

The example above prints all text files in the BASIC Language Library account, DSK0:[7,6].

9.6.7 PRINT Error Messages

Below are some of the common error messages you may see when using PRINT. Refer to Appendix A, AMOS System Error Messages, if you encounter a message not discussed below.

1. %No files in printer request
   PRINT was not able to find the files you specified. Check your spelling again, and make sure that you included the proper account and device specifications.

2. ?Specification error ^
   PRINT did not understand the format of your command line. The ^ symbol points to the item on the line that confused PRINT.

3. ?More than one printer specified
   You may not include more than one printerspec. Make sure that the equal sign in your command line was in the proper place:

   ```
   .PRINT LPT1,OLDFIL.TXT=OLDFIL.TXT↵
   ?More than one printer specified
   ```

4. ?Device or [P,PN] specifications on output are illegal
   PRINT thought that you were trying to include file specification elements in your printer specification. Make sure that you did not accidentally enter an equal sign as part of a file specification.

5. ?Output printer not found
   PRINT was not able to find the printer you specified. Check with the System Operator for a list of the names of the printers set up for use with the spooler, or use the PRINT command followed by a RETURN to see information about the printers whose queues contain print requests.

6. ?Invalid argument for COPIES
   You tried to give a non-numeric argument to the /COPIES:n switch; retype the command line.

7. ?Invalid argument for FORMS
   PRINT did not understand the symbols you entered after the /FORMS: switch. For example, you will see this message if you forget to enter the colon after the /FORMS: switch.

8. ;Jobname - Please mount form form-name on Printer-name
   The Spooler Operator job sees this message if you specify a form-type that is not defined as being mounted on the printer.

9. ?Cannot find DSK0:SCNWLD.SYS[1,4] or MEM:SCNWLD.SYS
   All wildcard file commands require the SCNWLD program to be in the System Program Library account (DSK0:[1,4]) or in memory. Check with the System Operator, who will make sure that a valid copy of SCNWLD is installed in DSK0:[1,4].

10. ?Invalid argument for LPP
    You gave a non-numeric argument to the lines per page switch or the format of your command was incorrect. Check your typing.

11. ?Invalid argument for WIDTH
    You gave a non-numeric argument to the page width switch or the format of your command was incorrect. Again, check your typing first.

CHAPTER 10

MORE FILE COMMANDS

In addition to the wildcard file commands, there are three other major commands you can use when working on files. None of these commands understand wildcard symbols. For information on the other file commands available, see the "AMOS System Commands Reference Sheets."

10.1 DISPLAYING THE CONTENTS OF A FILE (TYPE)

To display a copy of a text file on your terminal display, use the TYPE command. If you want to display specific disk records, or display the contents of a file as both ASCII characters and as octal or hexadecimal data, refer to the documentation on the DUMP command in the "AMOS System Commands Reference Sheets." We assume that the file you want to see is an ASCII text file. NOTE: ASCII is the name of the numerical code which the computer uses to represent text symbols. All files on the system are ASCII files except for .RUN, .OBJ, .PRG and .SBR files. (For information on ASCII code and the form in which the computer represents and stores data, see the manual "Introduction to AMOS.")

COMMAND SYNTAX:

```
.TYPE filespec )
```

COMMAND DEFAULTS:

The TYPE command assumes an extension of .LST and the account and device you are currently logged into.

To use TYPE, enter the word TYPE followed by the filespec of the file you want to see; then hit RETURN:

```
.TYPE PSTCRM.TXT )
```

If the extension of your file is .LST, you do not need to include the extension in your filespec. TYPE does not understand the wildcard symbols ? and *.

10.1.1 Hints and Restrictions

If the file display covers more than one screen-page, use Control-S to freeze the screen display and a Control-Q to release it.

TYPE works only on sequential files. (See the manual "Introduction to AMOS" for an explanation of sequential and random files.) Most files that you will be using on the AMOS system are sequential files (e.g., all text files that you create with EDIT and VUE). If you try to use TYPE on a random file, you see:

[CANNOT OPEN filespec - FILE TYPE MISMATCH]

where filespec is the filespec you gave to TYPE.

You can use TYPE on any kind of sequential file; however, if you use TYPE on a non-ASCII file, the system will still try to display the data in the file as if it were ASCII. Therefore, although you can use TYPE on a file containing machine language (a .PRG or .OBJ file), you will see the data interpreted as if it were text-- a confusing and meaningless display.

10.1.2 TYPE Error Messages

Below are some common error messages you can see when using TYPE. If you encounter an error not mentioned below, refer to Appendix A, AMOS System Error Messages.

1. [CANNOT OPEN filespec - FILE NOT FOUND]
   TYPE wasn't able to find the file you specified. Check your spelling; if that's OK, try using the DIR command (e.g., DIR ALL:[]filespec) to see if you can locate the account in which the file appears.

2. [CANNOT INIT filespec - DEVICE DOES NOT EXIST]
   You tried to specify a file on a device that TYPE cannot find. Check your spelling; if that's OK, check with the System Operator to see if the device is a valid system device:

   ```
   .TYPE DKS1:CRLF.MAC
   [CANNOT INIT DKS1: - DEVICE DOES NOT EXIST]
   ```

3. [CANNOT OPEN filespec - DISK NOT MOUNTED]
   The system can't access the device specified by your filespec. Use the MOUNT command to mount the disk.

4. [CANNOT OPEN filespec - FILE TYPE MISMATCH]
   You tried to display the contents of a random file:

   ```
   .TYPE REGINS.DAT
   [CANNOT OPEN REGINS.DAT - FILE TYPE MISMATCH]
   ```

You will have to use another method to look at the data in the file.

5. [FILE SPECIFICATION ERROR]
   TYPE doesn't understand the form of your command line. Check your spelling, and retype.

```
.TYPE ↵
[FILE SPECIFICATION ERROR]
```

6. [CANNOT OPEN filespec - ILLEGAL USER CODE]
   You've specified an account that does not exist. Check your typing and try again. If you still get this message, make sure that you are accessing the correct device.

10.2 APPENDING FILES (APPEND)

It's often convenient to keep a large project divided into sub-projects while you are working on the task. For example, this manual was written as a large number of small files. To combine several files together into one file, use the APPEND command.

COMMAND SYNTAX:

```
.APPEND outfilespec=infilespec1{,...infilespecN} ↵
```

COMMAND DEFAULTS:

The default extension for outfilespec and infilespecs is an empty extension.

The default account and device is the account and device you are currently logged into.

The APPEND command creates a new file (specified by the outfilespec) that contains the contents of all of the files specified by the infilespecs. You MUST specify an outfilespec:

```
.APPEND ASMBLR.MAC=PARSE.MAC,SCAN.MAC,TABLE.MAC,CNVRT.MAC ↵
```

The example above creates a new file named ASMBLR.MAC that contains the contents of the files PARSE.MAC, SCAN.MAC, TABLE.MAC and CNVRT.MAC (in that order). APPEND does not change the contents of any of the input files, and does not delete them from the disk.

If you want the new file to have the same name as one of the input files, your outfilespec may have the same name as one of the infilespecs (and APPEND will delete that infilespec when the file concatenation is finished).

```
.APPEND CTALOG.BAS=CTALOG.BAS,INVOIC.BAS ↵
```

The example above adds the contents of INVOIC.BAS onto the end of the file CTALOG.BAS.

10.2.1 APPEND Error Messages

Below are some common error messages that you may see when using APPEND. If you encounter an error not discussed below, refer to Appendix A, AMOS System Error Messages.

1. [CANNOT OPEN filespec - FILE NOT FOUND]
   You've specified a file that the system cannot find. Check your spelling and retype. If you still have no luck, use the DIR command (.DIR ALL:[] filespec) to try and find the account and device in which the file resides.

2. [CANNOT INIT devn: - DEVICE DOES NOT EXIST]
   Check the spelling of your device specifications; the system doesn't recognize the device you've specified. If your spelling looks OK, check with the System Operator for help.

   ```
   .APPEND DKS3:TOTAL.TXT=DSK3:PART1.TXT,DSK3:PART2.TXT↵
   [CANNOT INIT DKS3: - DEVICE DOES NOT EXIST]
   ```

3. [CANNOT OPEN filespec - DEVICE NOT MOUNTED]
   The device you've specified is a legal system device, but it is not mounted. Use the MOUNT command. Now try appending again.

4. [CANNOT OPEN filespec - FILE TYPE MISMATCH]
   You can only use APPEND on sequential files. You will see this message if you try to use APPEND on random files.

5. [FILE SPECIFICATION ERROR]
   APPEND didn't understand the form of your command line:

   ```
   .APPEND↵
   [FILE SPECIFICATION ERROR]
   ```

   Retype the command line.

10.3 SORTING A FILE (SORT)

Use the SORT command to alphabetically and numerically sort the data in a text file. For example, suppose you have a file that contains a list of customer names, addresses, phone numbers and dates of last purchase. Each line in your file contains this information for one customer. (A line-- the group of characters between two carriage returns-- is called a logical record.)

You may want to rearrange the file so that the records are ordered by state, with the record containing information about Smith's Pharmaceuticals of Alabama appearing first in the file, and the record containing information about A&B Food Services of Rhode Island appearing last.

SORT allows you to choose the item (called the key) in the record on which to base the sort, and it allows you to sort on three different keys. For example, you can tell SORT to sort the records alphabetically by state; then for the customers within each state, sort alphabetically by customer name.

NOTE: You may only sort sequential files. You may sort files that are too large to fit into memory all at one time.

COMMAND SYNTAX:

.SORT filespec ↵

COMMAND DEFAULTS:

SORT assumes an extension of .DAT (for data file), and the account and device you are currently logged into.

To sort a sequential file, type SORT, the specification of the file you want to sort and a RETURN:

.SORT INVITM.DAT ↵

Now SORT begins to ask you questions about your data file and about how you want to sort the data in the file. SORT actually replaces your file with another file in which the data are arranged in the proper order. (See 10.3.3, Example, for a sample use of SORT.)

These are the questions that SORT asks:

1. Record size: - Although SORT knows that each logical record ends with a carriage return symbol, it needs to know the size of the largest logical record that it is going to be dealing with. Enter the maximum size (in bytes) of the logical data records in the file. For example, if every customer has a logical record in which his or her name, address and phone number appears, enter the size of the largest of these records. (Every character in the logical record is one byte of data, including spaces and punctuation. Exclude carriage return and line-feed symbol bytes.)

2. Key size: - The key is the field in the logical record on which you wish to sort (e.g., customer name, state or store name). Because you can choose up to three different keys, SORT asks this question once for each key that you define. If you want SORT to use less than three keys, answer this question with a RETURN after you have defined all of the keys that you want to use. Enter the size of each key in bytes.

3. Key position: - SORT asks this question for each of the keys that you define. Enter the column number in the data record where the first byte of the sort key occurs. (The first byte of a record is position #1.)

4. Key Order: - SORT asks this question once for each of the keys that you define. Enter an A if you want that key sorted in ascending order, or enter a D if you want the key sorted in descending order.

   NOTE: SORT sorts data based on the values of the ASCII codes for that data. Because each character has a different ASCII value, upper case letters are NOT the same as lower case letters. (That is, an "A" is not the same as an "a.") For that reason, all keys that begin with a capital letter will come before all keys that begin with a lower case letter (or vice versa, depending on whether you are sorting in ascending or descending order).

10.3.1 SORT Statistics

After SORT has sorted your file, it displays a number of statistics so that you can get an idea of how much your file was in need of being sorted:

1. Sorted n Records - This line tells you how many logical records SORT processed. If this number differs from the number of records you know to be in the input file, check your data file again and make sure that it is the correct one.

2. n Run - SORT tells you how many passes it made through the data to sort the file. If the entire file fits into memory, SORT performs the sort in memory, and you see "1 Run"; if your file was too large to fit into memory, SORT performs the sort on the disk (a modified poly-phase merge sort), and you see that SORT performed its sort in more than one Run.

3. n Key comparisions, m per record - This line tells you how many comparisons SORT made while doing the sort-- a fair indication of how out of order the file was to begin with. In general, the more out of order a file is, the larger the number of comparisons needed to sort it. (This is not always the case, since files that are already very close to being in order sometimes take more comparisons than files that are in more random order.)

4. hh:mm:ss Elapsed time, n ms per record - Note that elapsed time is given rather than compute time; this value will be affected by the number of other users on the system, and on the type of processing that they are doing.

10.3.2 Hints and Restrictions

SORT can sort files that are much larger than available memory, so you can sort files in a very small amount of memory (as little as 6K bytes in many cases). However, bear in mind that (in general) the more memory available, the faster the sort.

You may only sort sequential files.

SORT does not understand wildcard symbols in Filespecs.

If the file that you are sorting is too large to fit into memory all at one time (that is, if SORT is unable to perform the entire sort in memory), SORT performs the sort on the disk.

IF THE SORT IS PERFORMED ON THE DISK:

The condition of the disk on which you are performing the sort affects SORT's speed. A disk that has been used a great deal has much of its free space distributed randomly across the disk surface; this slows down the sort. If you use a freshly initialized disk, the temporary sort files tend to be allocated contiguously, which speeds up the sorting process somewhat. (The System Operator can initialize a disk for you. WARNING: Initializing a disk removes all data on that disk.)

If you are sorting files on floppy disks, you will find that sorts are faster if performed on AMS-format floppy disks rather than IBM standard-format disks.

You must make sure that each record is greater in length than the key sizes you specify; otherwise you will end up with a null file.

10.3.3 Example

Below is a small, unordered data file:

```
Hinchey, Edsel  7701 Wanda Dr    Santa Ana CA
Dion, F.        27819 Glen Anne  Anaheim   CA
Kinslayer, E.   5678 Calle Rio   Tucson    AZ
Clayton, Alfred 8523 Delta Rd    Orange    CA
Vine, Irv       243 Princeton    Lowell    MA
Swenson, John   120 Halstead     Chicago   IL
```

We'll assume that you want to sort the file first by state; then within state by customer name. The number of characters in each logical record (i.e., on each line) is 45 (including spaces and punctuation, but excluding the carriage return at the end of each record). So, the maximum record size is 45 bytes. Customer names begin in position #1, and states begin in position #44:

```
.SORT DATA↵

Record size: 45↵

Key size: 2↵
Key position: 44↵
Key order: A↵

Key size: 16↵
Key position: 1↵
Key order: A↵

Key size: ↵

Sorted 6 records
1 Run
20 Key comparisions,   3.33 per record

Elapsed time 00:00:05,          920 ms per record
```

If we look at the data file, it now looks like this:

```
Kinslayer, E.   5678 Calle Rio  Tucson     AZ
Clayton, Alfred 8523 Delta Rd   Orange     CA
Dion, F.        27819 Glen Anne Anaheim    CA
Hinchey, Edsel  7701 Wanda Dr   Santa Ana  CA
Swenson, John   120 Halstead    Chicago    IL
Vine, Irv       243 Princeton   Lowell     MA
```

10.3.4 SORT Error Messages

Below are some common error messages that you may see while using SORT. If you encounter an error not mentioned below, refer to Appendix A, AMOS System Error Messages.

1. ?Enter A or D
   SORT wants to know whether you want to sort the key in ascending (A) or descending (D) order; you must enter an A or a D.

2. ?Insufficient memory for sort
   SORT must be able to fit at least five of your data records into memory to perform a sort; if it cannot, you will see this message.

3. ?Record size must be >0
   ?Key size must be >0
   ?Key size must be less than record size

   SORT checks to make sure that you are giving it reasonable data. If you see any of these messages, it probably means that you have made a typing mistake. Key and record size must be at least one byte in length, and (since the key is an element of the record) the key must be smaller than the record.

4. ?Entire key must be within record

   If the [(start-position in the record of the key + the length of the key)-1] is beyond the end of the record, you see this message. (The minus one comes from the fact that the first position in the record is a one and not a zero.) SORT thinks that the record size you gave it was too small or that the key size you gave was too big; check your numbers.

5. ?Cannot DELETE Filespec - write protected

   You have tried to sort a file on a write-protected disk. SORT cannot replace your unsorted file with the new, sorted file, because it cannot write on the disk. Write-enable your disk and try again.

CHAPTER 11

MEMORY COMMANDS

To run a program, AMOS must find that program on the disk as a file, load a copy of the file into memory and execute it. The copy of a disk file that is loaded into memory is called a memory module.

Because many users may be on the system at the same time, all users have their own portion of memory in which to run their own programs; this portion of memory is called a user memory partition. When you enter a command at AMOS command level, AMOS loads a copy of the appropriate program or command file into your own memory partition and executes it. In the case of system programs (e.g., DIR) or command files, AMOS automatically deletes the program or command file from your memory partition after it finishes executing it. Check with your System Operator if you have questions about the amount of memory allocated to you. To find out what modules are in your memory partition, you can use the MAP command. (See Section 11.2, Finding Out What Modules Are in Memory-- MAP.)

When a particular program is often used by several users at the same time, the System Operator may arrange to place the program in system memory (the area of memory used by AMOS itself, and not by any individual user). When a program is in system memory, all users may use it at the same time, and that program does not have to be loaded into your own memory partition for you to use it (which gives you more free memory space). To find out what programs are in system memory, you can use the SYSTEM command (see Section 11.3, Finding Out What Modules Are in System Memory-- SYSTEM).

11.1 LOADING FILES INTO YOUR MEMORY PARTITION (LOAD)

Use the LOAD command to load a copy of a file into your memory partition. NOTE: You can load and execute assembly language programs (.PRG files) or command files (.CMD or .DO files) by simply entering the specification of the file at AMOS command level. There are times, however, when you might want to load a group of files into memory without executing them (e.g., a BASIC program may require that a group of subroutines be loaded into memory before you can execute that program).

COMMAND SYNTAX:

.LOAD Filespec ↵

COMMAND DEFAULTS:

LOAD assumes an extension of .PRG and the account and device you are currently logged into.

11.1.1 Hints and Restrictions

LOAD can be used to load the LOAD program itself (DSK0:LOAD PRG[1,4]).

If the file you are loading into memory is already in memory, LOAD deletes and replaces it with a fresh copy.

LOAD does not understand wildcard symbols.

LOAD does NOT tell you what files it has loaded into memory; you will know it was unsuccessful if it displays this message:

?Filespec NOT FOUND

11.2 FINDING OUT WHAT MODULES ARE IN MEMORY (MAP)

Use the MAP command to see a list of the modules in your memory partition or in system memory. Modules retain the same name and extension as their corresponding disk files (e.g., when a copy of DSK0:LOG.PRG[1,4] is loaded into memory, AMOS is able to locate it in memory by looking for a module named LOG.PRG).

COMMAND SYNTAX:

.MAP {Filespec}{/Switches}↵

COMMAND DEFAULTS:

The default Filespec is *.*.

The default Switches are: /FSBHMU

11.2.1 Finding Out What Modules Are in Your Memory Partition

Type MAP followed by a RETURN:

```
.MAP↵
```

You see a list of the memory modules currently in your memory partition with the following information on each line: 1. module name; 2. extension; 3. size in bytes (in decimal); 4. octal base address (the memory address where the module begins); and, 5. hashmark (see Section 9.2.6, Selecting DIR Options, for a discussion of hashmarks). The last line of the display tells you how many bytes are free in your memory partition, and gives the octal memory address of the first available memory location in the memory partition.

```
SYSTAT  PRG   566  033722  167-536-542-221
LMG     PRG  1016  035022  512-123-435-601
QDT     PRG   330  037024  435-713-462-175
FREE        48416  040736
```

We can tell from the example above that the SYSTAT module is a copy of an assembly language program (a .PRG file), is 566 bytes in length, and begins at the memory address 033722. The module's hashmark is 167-536-542-221. (The hashmark of a memory module is identical to the hashmark of the disk-file version of that module; that fact gives you the ability to compare modules and files, making sure that the proper version has been loaded into memory.) The last line of the display tells us that we have 48,416 bytes free in our memory partition, and that the first free memory location is 040736.

11.2.2 Displaying Information About Specific Memory Modules

If you want to discover the size of a particular memory module in your memory partition or its base memory address or hashmark, supply a Filespec to the MAP command:

```
.MAP LOG↵
LOG     PRE   1016  035022  512-123-435-601
FREE        48416  040736
```

11.2.3 Selecting MAP Options

You may choose among several MAP options by including one or more of the one-character MAP Switches at the end of the command line following a single slash (/). Each character after the slash up to the RETURN at the end of the command line is considered to be a separate Switch, so MAP Switches MUST appear at the end of the command line.

The MAP Switches allow you to limit the MAP display by requesting only

specific items of information. They also allow you to see the modules in system memory. (Also, see Section 11.3, Finding Out What Modules Are in System Memory-- SYSTEM.)

The MAP switches are:

/F FREE - Display amount of free memory bytes available (in decimal). (You must use with the /S switch.)

/S SIZE - Display size (in bytes) of each module listed; decimal number.

/B BASE ADDRESS - Display base memory address for each module (in octal).

/M MODULES - Display information about modules.

/U USER MEMORY - Display information about modules that are in your memory partition.

/R RESIDENT MEMORY - Display information about modules that are in system memory.

/H HASHMARK - Display hashmarks for each memory module in display.

The default switches are: /FSBHMU

11.2.3.1 Limiting the MAP Display - If you omit the MAP switches from a command line, MAP gives you as much information as it can. If you are not interested or do not need quite that much information, you may use the MAP switches to restrict the MAP display. For example, if you merely want to know the size of the modules in your memory partition or their names, use only one or two of the switches discussed above:

To see just the names of the modules in your memory partition:

```
.MAP/UM ↵
LOG     PRG
QDT     PRG
SYSTAT  PRG
```

To see just the sizes and names of the modules in your memory partition:

```
.MAP/UMS ↵
LOG     PRG   1016
QDT     PRG    330
SYSTAT  PRG    566
```

To see just the names and hashmarks of the modules in your memory partition:

```
.MAP/UMH↵
LOG     PRG  512-123-435-601
QDT     PRG  435-713-462-175
SYSTAT  PRG  167-536-542-221
```

To find out how much free memory is left in your memory partition:

```
.MAP/FS ↵
FREE    48416
```

And, so on.

11.2.3.2 Using MAP to Find Out What Modules Are in System Memory - The examples in the section above are concerned with displaying information about the modules in your memory partition. You may also use MAP on the system area of memory by replacing the /U Switch with the /R (resident area) Switch:

```
.MAP/RSHMF↵
TRM     DVR  252  552-107-745-717
```

11.3 FINDING OUT WHAT MODULES ARE IN SYSTEM MEMORY (SYSTEM)

To find out what modules are in the system area of memory, type SYSTEM followed by a RETURN:

```
THE FOLLOWING PROGRAMS ARE ALLOCATED IN SYSTEM MEMORY
        TRM     DVR
TOTAL RESIDENT MONITOR SIZE IS 13426 BYTES
```

The total resident monitor size is the number of bytes taken up by system memory. The contents of system memory consist of those programs that make up the operating system and those programs shared by all users on the system. Use the DIR command with the RES: device to see what programs are in system memory, or the MAP command (see Section 11.2.3.2, Using MAP to Find Out What Modules Are in System Memory).

NOTE: On a hard-disk system you can use SYSTEM to find out if the system is running off the cartridge or a fixed disk. Type SYSTEM followed by a RETURN. If the SYSTEM display says:

```
*** SYSTEM IS RUNNING FROM CARTRIDGE DISK ***
```

the system has booted off the cartridge, not a fixed disk.

11.4 SAVING MEMORY MODULES AS FILES (SAVE)

You can save memory modules in your memory partition as disk files by using the SAVE command.

COMMAND SYNTAX:

.SAVE filespec1{,....filespecN}{/rename extension}↵

COMMAND DEFAULTS:

SAVE uses a default extension of *; if you specify just a filename, SAVE saves all memory modules of that name, whatever their extension.

To save a copy of a module as a disk file, type SAVE followed by a list of filespecs selecting the modules you want to save. SAVE understands the wildcard symbols ? and *. The memory modules bear the same name and extension as their corresponding files on the disk; use the MAP command or DIR MEM: to find out what modules are in your memory partition.

```
.SAVE *.PRG,*.SYS↵
LOG.PRG
QDT.PRG
SYSTAT.PRG
SCNWLD.SYS
.
```

If you ask SAVE to transfer a copy of a module to your account and a disk file of that name and extension already exists, SAVE erases the original disk file and replaces it with a copy of the memory module:

```
.SAVE LOG↵
ERASE LOG.PRG,  SAVE LOG.PRG
.
```

In the example above, a file named LOG.PRG already existed in the account; SAVE erased it, and replaced it with a new copy of the memory module LOG.PRG.

If you do not want SAVE to delete existing files, use the rename option: end the SAVE command line with a slash and an extension; if SAVE finds a file in the account with the same name and extension as the module you want to save, SAVE renames the file in the account with the new extension. For example, let's say that a file EXIT.SBR exists in your account. If you want to save a module in your memory partition named EXIT.SBR, you can either say:

```
.SAVE EXIT.SBR↵
ERASE EXIT.SBR,  SAVE EXIT.SBR
```

which deletes and replaces the file already on the disk, or:

```
.SAVE EXIT.SBR/OLD ↵
RENAME EXIT.SBR,          SAVE EXIT.SBR
```

which renames the disk file EXIT.SBR to EXIT.OLD and saves the memory module as disk file EXIT.SBR. If the account contains a file with the same name and extension as the file being renamed, SAVE deletes that original file. For example, if a file EXIT.OLD already exists in the account, the example above would look like this:

```
.SAVE EXIT.SBR/OLD ↵
ERASE EXIT.OLD, RENAME EXIT.SBR,          SAVE EXIT.SBR
```

SAVE lists all modules that it saves; if you do not see such a list, it means that SAVE was not able to save the modules you specified.

11.5 DELETING MEMORY MODULES FROM YOUR MEMORY PARTITION (DEL)

To delete memory modules from your memory partition, use the DEL command.

COMMAND SYNTAX:

```
.DEL filespec1{,....filespecN} ↵
```

COMMAND DEFAULTS:

DEL assumes an extension of *.

DEL understands the wildcard symbols ? and *. To delete modules, type DEL followed by a list of specifications; then type a RETURN.

```
.DEL *.PRG,*.RUN ↵
 REMOVE.PRG
 MENU.RUN
 XOR.RUN
```

DEL displays no error messages; you will know that it was unable to perform its function if it does not report back with a list of the modules deleted.

As DEL deletes modules from your memory partition, it shifts the remaining modules down in memory (i.e., toward location zero) if they have addresses greater than those of the deleted modules.

CHAPTER 12

SYSTEM INFORMATION COMMANDS

The last group of commands that we document in this manual are those commands that provide you with information about the system. There exist on the system many other programs that reveal details about how the system is functioning. These programs include: diagnostic programs for testing memory and disks; display programs that help you examine and modify memory and the disk; and job commands that allow you to attach jobs and terminals and affect job status. These commands are not within the scope of this manual; for more information on system operation, see the "AMOS System Operator's Information" section of the AM-100 documentation packet and the "Assembly Language Programmer's Manual."

12.1 SYSTEM STATUS COMMAND (SYSTAT)

The System Status command give you a quick summary of what jobs are running on the system, what terminals are attached to what jobs and what each job is doing.

COMMAND SYNTAX:

.SYSTAT ↵

The SYSTAT program prints one line for each job on the system. Each line gives the following information: 1. job name; 2. name of the terminal that the job is using; 3. account the job is logged into; 4. the octal base address in system memory where the system maintains information about the job; 5. status of the job (see Section 12.1.1, Job Status Symbols, below); 6. name of the last program run by the job; and 7. the number of bytes (decimal) in the memory partition being used by the job, and its base address in memory (octal).

Below the job information lines is a list of all storage devices currently mounted on the system and the number of blocks free on those devices.

A typical display for a small system might look like this:

```
JOB1   TRM1  110,4    22276  RN  SYSTAT   51000 BYTES AT LOC 33710
SPOOL  NULL  1,2      22742  EW  LPTSPL     854 BYTES AT LOC 32162

DSK0   5321  FREE BLOCKS
DSK1   5134  FREE BLOCKS
```

On this particular system there is only one user (JOB1) who is using terminal TRM1. JOB1 is logged into [110,4], and is running SYSTAT. JOB1's memory partition begins at memory address 33710, and is 51000 bytes large.

The second job on the system (SPOOL) is a job being used by the sole user of the system to run the line printer spooler program (LPTSPL) so that when the user wants to print a file, he or she can have SPOOL do it without interrupting the user's other activities. (See Section 9.6, Printing a File-- PRINT, for information on line printer spoolers.)

The last two lines of the display show that the devices DSK0: and DSK1: are mounted, and that they have 5321 and 5134 free disk blocks, respectively.

12.1.1 Job Status Symbols

The job status symbols in the SYSTAT display tell you what the jobs on the system are doing:

TI	TERMINAL INPUT WAIT STATE - The job is waiting to receive data from a terminal, but nothing's been entered from the keyboard.
TO	TERMINAL OUTPUT WAIT STATE - The job is waiting to output data to a terminal, but the buffers (data storage areas) are full.
LD	PROGRAM LOAD STATE - The job is in the process of loading a program.
SL	SLEEP STATE - The job has been put into hibernation by using the SLEEP command.
DS	DISK ACCESS IN PROGRESS - Job waiting for the system to finish reading or writing data on the disk.
IO	I/O ACCESS OTHER THAN TERMINAL OR DISK - Job is waiting for data transfer to finish between system and a device other than a terminal or disk.
EW	EXTERNAL WAIT - Job is waiting for someone or something to wake it up.
RN	RUNNING - Job is running a program.

SP SUSPENDED STATE - After a SUSPND command, the job is in suspension; must be revived by a REVIVE command.

^C CONTROL-C - Job is at AMOS command level, waiting for a new command.

12.2 THE SET COMMAND

The SET command allows you to select various options that affect how AMOS communicates with your terminal. For example, you can use the SET command to choose the number base (octal or hexadecimal) AMOS uses when displaying non-decimal numeric data on your screen. You can also choose various error-reporting options. The options you select affect only your job.

COMMAND SYNTAX:

.SET {option}↵

COMMAND DEFAULTS:

The default options are: OCTAL, ECHO, NOVERIFY, NOGUARD, NODSKERR, CTRLC.

To see what options are in effect for your job, type SET and a RETURN:

```
.SET↵
Current settings are:
        OCTAL  ECHO    DSKERR  NOVERIFY        NOGUARD CTRLC
```

To set an option, type SET and the option name (see below); then hit RETURN:

.SET DSKERR↵

The SET options are:

CTRLC	Tells AMOS to recognize Control-Cs (the user-interrupt command).
NOCTRLC	Disables Control-C recognition by AMOS.
OCTAL	Uses base-8 for non-decimal numeric displays.
HEX	Uses base-16 for non-decimal numeric displays.
ECHO	Displays terminal input.
NOECHO	Silences terminal input.
DSKERR	Reports all soft disk-errors and I/O retries.
NODSKERR	Silences reports of soft disk-errors and retries.

VERIFY	Verifies all write operations by reading the data each time written, comparing it to the data in memory.
NOVERIFY	Does not verify write operations.
GUARD	Does not allow other terminals to send messages to your own terminal via the SEND command.
NOGUARD	Allows other terminals to send messages via SEND.
BPI	Sets the magnetic tape unit bits per inch rate for data reading or recording on tape. Specify a device and a BPI value (e.g., SET BPI MTU0:800).
FORMS	Allows you to set the name of the form to be mounted on a specified printer. (See the PRINT command for information on using this option.)

If SET doesn't understand an option name, you see:

```
?Invalid function
```

Try entering the command line again.

If you give an invalid device specification or incorrect BPI value to the SET BPI command, you see:

```
The format for the command is:  SET BPI MTU*:XXXX
Where * = a tape drive in the range 0 thru 7 and
XXXX is either 800 or 1600.
```

Check your device specification. Also, make sure that you have specified a BPI value of either 800 or 1600. The system comes up with a value of 1600.

If you use SET to set the form mounted on a specific printer, and SET does not recognize the printer's name, you see: ?Printer not found. For example:

```
.SET FORMS PRNTR4 TAXTBL↵
?Printer not found
```

The command above tried to associate the form-type TAXTBL with the printer PRNTR4; SET was not able to find PRNTR4. Check with the System Operator for a list of printers set up for line printer spooler use.

12.3 FINDING OUT WHAT DEVICES ARE ON THE SYSTEM (DEVTBL)

To see what devices your system is set up to use, type DEVTBL and a RETURN:

```
.DEVTBL ↵
DSK0 (SHARABLE)
DSK1 (SHARABLE)
STD0 (SHARABLE)
STD1 (SHARABLE)
RES0 (SHARABLE)
MEM0 (SHARABLE)
TRM0 (SHARABLE)
MTM0
```

The devices marked as (SHARABLE) are those which more than one user can access at a time (e.g., a disk or memory); nonsharable devices are those devices that cannot be shared (e.g., a paper-tape punch or a printer).

To find out which devices are mounted and ready to use, use the SYSTAT command (see Section 12.1, The System Status Command-- SYSTAT).

12.4 FINDING OUT THE NAME OF YOUR JOB (JOBS)

If you wish to use the SEND command (see Section 12.5, Sending Messages to Other Jobs-- SEND), each user on the system will need to know the name of his job. You will also want to know your job name if you want to interpret the SYSTAT display mentioned above.

To find out your job name, type JOBS and a RETURN:

```
.JOBS ↵
YOUR JOB NAME IS JOB1
```

12.5 SENDING MESSAGES TO OTHER JOBS (SEND)

The SEND command allows you to exchange messages between several terminals on the system. To exchange messages, the communicating terminals must be connected to the same system and turned on.

COMMAND SYNTAX:

```
.SEND jobname Message ↵
```

where jobname selects the job to whom you are sending a message.

Type SEND followed by name of the job with whom you want to communicate; then enter the message and a RETURN. To find out the names of the jobs on the system, use the SYSTAT command (see Section 12.1, The System Status Command-- SYSTAT). To find out your job name, use the JOBS command (see Section 12.4, Finding Out Your Job Name-- JOBS).

As an example, assume that you are JOB1, and that you want to send a message to the user whose job name is ROBIN:

```
.SEND ROBIN HOW LONG IS THAT LISTING YOU ARE PRINTING?↵
```

The message appears on the terminal attached to job ROBIN prefaced by your job name:

```
; JOB1  - HOW LONG IS THAT LISTING YOU ARE PRINTING?
```

That user may now send a message back to you:

```
.SEND JOB1 ABOUT 17 PAGES↵
```

You can guard your job from receiving messages by using the SET GUARD option (see Section 12.2, The SET Command).

12.5.1 SEND Error Messages

Below are the error messages you may see when using SEND:

1. ?Job not found
   You tried to send a message to a nonexistent job; make sure that you spelled the job name correctly, and that the job does indeed exist. Use the SYSTAT command to see what jobs are currently running on the system.

2. ?Job specification error
   You did not supply a job name (i.e., you typed SEND followed by a RETURN). You must specify a job to whom the message may be sent.

3. ?Busy
   SEND cannot send the message to the job you specified because that job is not in terminal input mode.

4. ?Job has no terminal attached
   The job with which you wish to communicate has no terminal attached; there is no device to receive your message.

5. ?Guarded
   You are trying to send a message to a job that is protected from messages by the SET command's GUARD option (see Section 12.2, The SET Command).

CHAPTER 13

DISK BACKUP PROCEDURES

If there is one lesson learned early in the computer business, it is this: BACK UP YOUR DATA!!! Backing up your programs and data by regularly copying them onto a spare disk is perhaps the most important habit you can develop.

Computers are generally very reliable machines, but nobody can completely guard against the freak occurrence: someone tripping over the power cord, a cup of coffee down the disk drive, power outages, floods, etc. Your programs and data are irreplaceable, so keep a current copy of them in a safe place. Unless it has happened to you, it's hard to imagine the frustration of losing several days of programming or data entry.

We will discuss specific backup techniques in a later section of this chapter. For now, let's talk about some other things that you can do when using the system to keep data loss to a minimum.

1. Whenever you change a disk cartridge or floppy, you MUST use the MOUNT command; if you do not, AMOS may well write over data that is already on the disk.

2. If your text files begin to show strange typos (e.g., "mempry" instead of "memory") ask the System Operator to run a memory diagnostic test program-- you may have a memory problem. (The System Operator should be running memory and disk diagnostic tests on a regular schedule, anyway.)

3. Keep your text editing sessions short; never use one of the text editors for more than a half hour or so without exiting. If that unexpected catastrophe should occur, it's better to have lost a half an hour's worth of work instead of a day's. (The changes and additions that you make while you are editing do not take affect until you exit the editor.)

4. As a general rule, never leave your terminal without signing off the system (use the LOGOFF command). Change your password occasionally. These measures protect the system from unauthorized use.

(Changed 15 October 1979)

5. Treat your backup disks with respect. Whether your backup disks are disk cartridges or floppy disks, handle them carefully and gently. Do NOT stack cartridges in tall piles on top of the disk drives, and do NOT leave them lying about where they can be knocked off counters or desks.

If at all possible, you should have several spare disks that you can use as backup disks. You should never have just one backup disk (what happens if something should really go wrong while you are in the process of backing up, and you lose both your backup and original disks?)

The usual procedure is to follow the "grandfather-father-son" philosophy of disk backup. That is, you always have several backups of varying age. The next time you back up, use the disk with the oldest version of your data (the "grandfather" disk). By rotating your backup disks, you can ensure that you always have several fairly recent backups of your data.

13.1 BACKING UP THE FILES IN YOUR ACCOUNT (USING COPY)

You can use the COPY command to make backup copies of all the files in your account, all of your accounts, and even of all accounts on the disk. The first step to take is to use the SET command to tell AMOS to verify every write operation:

```
.SET VERIFY ↵
```

After you've told AMOS to SET VERIFY, every time it writes data to the disk, AMOS compares the data to the corresponding data in memory to make sure that they are the same.

Use the SET command to tell AMOS to notify you of any soft disk-errors encountered during the backup procedure:

```
.SET DSKERR ↵
```

Now, decide what files you want to back up. For example, let's say that you are logged into DSK0:[300,4], and want to copy all files in that account to the same account on DSK1:.

```
.COPY DSK1:= ↵
CRMINV.BAS to DSK1:CRMINV.BAS
RESET.BAS to DSK1:RESET.BAS
CRITEM.BAS to DSK1:CRITEM.BAS
PRINT.TXT to DSK1:PRINT.TXT
Total of 4 files transferred
```

The fact that nothing appears on the right side of the equal sign in the command line above tells COPY that we want to copy ALL files from the device and account we are logged into. The DSK1: on the left hand side of the equal sign tells COPY to transfer copies of those files under their own names to the same account on DSK1:.

```
.COPY DSK1:[]=[300,*]↵
FUNC.BAS[300,1] to DSK1:FUNC.BAS[300,1]
PRTYP.PRG[300,2] to DSK1:PRTYP.PRG[300,2]
IBUF.MAC[300,2] to DSK1:IBUF.MAC[300,2]
IBUF.PRG[300,2] to DSK1:IBUF.PRG[300,2]
PARSER.PRG[300,10] to DSK1:PARSER.PRG[300,10]
Total of 5 files transferred
```

The command above tells COPY to copy all files from all accounts in project 300 on the device you are logged into over to the same accounts on DSK1:. (You must remember to include the wildcard PPN symbol [] on the left side of the equal sign-- if you do not, COPY will copy every file in all accounts on the device into which you are logged into just the single account that you are using on DSK1:. (In the example above, you must be logged into an account in Project 300 to avoid a Protection Violation error.)

13.1.1 The System Operator and the COPY Command

The System Operator (if logged into DSK0:[1,2]) can use COPY to back up all of the accounts on a disk regardless of their project numbers. For example:

```
.COPY []=DSK1:[]↵
```

copies all files in all accounts on DSK1: over to their corresponding accounts on the device the System Operator is logged into. (NOTE: The System Operator may omit the wildcard PPN symbol in the outfilespec-- when used from account DSK0:[1,2], COPY assumes a default outfilespec of *.*, the device you are logged into, and an account specification of [].)

You should perform file backup frequently if you create and change files often. Perhaps at the end of every working day you should copy those files onto a backup disk. Creating a command or DO file to perform this function for you can make your life simpler; see Section 8.2.3.3, BACKUP.DO, for an example of a DO file that backs up files from one disk cartridge to another.

13.2 BACKING UP ENTIRE DISKS (DSKCPY)

Rather than copying over individual files from one device to another, DSKCPY makes a literal image copy of an entire disk. We will assume for the purposes of this discussion that you want to back up the System Disk.

IMPORTANT NOTE:

> You must not use DSKCPY on any kind of disk device that contains the file BADBLK.SYS[1,2]. Making a literal image of such a device onto another disk might result in bad blocks being written into and the BADBLK.SYS file on the new disk being overwritten. Use the COPY command instead.

NOTE: You can use DSKCPY on any kind of disk device that does not contain the BADBLK.SYS[1,2] file, but you can only copy between devices of the same type (e.g., AMS1: to AMS2:, DSK2: to DSK3:, DDS1: to DDS2:, etc.).

Get the system up and running. If you are using a floppy-disk based system, place the backup disk in a drive other than the System Drive. If you are using a hard-disk based system, place the backup disk cartridge in any of the drives not containing the System Disk. The backup disk must have been formatted at some time in the past. (Check with the System Operator for information on what formatting program to run to format a brand new disk.) Note that DSKCPY completely obliterates any data previously on the backup disk. Mount the backup disk. Enter:

```
.DSKCPY↵
```

Now the DSKCPY command asks you for the INPUT DRIVE. Enter the specification of the device you are going to back up. Now DSKCPY asks you for the OUTPUT DRIVE. Enter the specification of the disk you are going to copy onto (the backup device). For example, if you want to copy DSK0: onto DSK1:, use the DSKCPY command in this way:

```
.DSKCPY↵
Input drive: DSK0:↵
Output drive: DSK1:↵
```

DSKCPY begins to make a literal image of DSK0: onto DSK1:. It tells you how many blocks it is copying. After it has finished the backup, DSKCPY tells you that it has finished copying the disk; then it proceeds to verify the duplication by checking the data on the backup disk against the data on the source disk. After this verification is complete, DSKCPY returns you to AMOS command level. The entire process looks like this:

```
.DSKCPY↵
Input drive: DSK0:↵
Output drive: DSK1:↵
[Copying 9696 records]

[Duplication completed]
[Verification completed]
.
```

NOTE FOR HAWK HARD-DISK SYSTEM USERS: if you have been working off a disk cartridge, and want to back up the entire disk, you can use DSKCPY. However, you must first save the contents of the System Disk, copy the cartridge contents over the contents of the System Disk, then copy the new contents of the System Disk to another disk cartridge. You must restore the contents of the System Disk by copying the System Disk backup cartridge to the System Disk. This process is tricky, since it temporarily destroys the contents of the System Disk, and must not be undertaken lightly. We advise that the System Operator be in charge of this kind of operation. (Of course, if you have more than one Hawk drive on your system, you can simply copy the cartridge from one drive onto a backup cartridge on the other Hawk drive.)

APPENDIX A

AMOS SYSTEM ERROR MESSAGES

Below is a list of most of the error messages that you can see when operating at AMOS command level. For information on the specific error messages you might see when using a particular command, see the documentation for that command.

The most common kind of error message that you will see on the system consists of two parts. The message begins with:

?Cannot {INIT, OPEN, CLOSE, READ, WRITE, INPUT, OUTPUT, WAIT, DELETE, RENAME, ASSIGN, DEASSIGN, or ACCESS} Filespec or Devn: -

and ends with one of the messages below (e.g., ?Cannot READ AMS3: - disk not mounted):

1. - bitmap kaput

   Every time the system writes data to a file-structured device (e.g., DSK0:, AMS1:), it checks that device's bitmap to determine where on the disk to write the data. (A bitmap is a map of the device-- it tells the system what records on the device already contain data, and which are free for use.) Every time the system checks the bitmap, it sums the data in that bitmap and checks it against the value it has previously computed; if there is a discrepancy, it means that something in the bitmap has changed, and you see the - bitmap kaput message. MOUNT the disk. If you still get this message, check with the System Operator for help-- he or she can run a disk analysis program that will reset the bitmap.

2. - buffer not INITed

   Before your assembly language program can perform an open, read or write operation, the program must assign a buffer to the file DDB. (Because you can use the monitor call INIT to do this, we say that the buffer has been INITed.) If you fail to do this procedure, you will see this message. If you should see this message from within BASIC, the message may indicate memory problems; check with the System Operator.

(Changed 15 October 1979)

3. - device does not exist
   You've tried to access a device that the system does not recognize. For example:

```
.TYPE ASM1:FILE.TXT↵
?Cannot INIT ASM1:FILE.TXT - device does not exist
```

4. - device error
   A hard disk-error has occurred; that is, the system was not able to read data from a disk. Try to perform the operation again. If you still have no success, check with the System Operator for help; this message may indicate a hardware problem.

5. - device full
   There is not enough room on the disk to complete the data transfer. Start over again with another device, or make room on the first device by erasing unnecessary files.

6. - device in use
   Another user is using the non-sharable device that you wish to access (e.g., a paper tape punch). Wait and try again later.

7. - device not mounted
   You have tried to access a valid system device, but that device is not mounted. Use the MOUNT command and try again.

8. - device not ready
   You are trying to access a device that is not ready. For example, you will see this message if you try to mount a disk before that device is powered up and ready.

9. - file already exists
   You've tried to create a file that already exists. For example, if the file NEWCPY.MAC already exists in your account, and you try to rename another file to that name:

```
.RENAME NEWCPY=WRKFIL↵
WRKFIL.MAC to NEWCPY.MAC
?Cannot RENAME WRKFIL.MAC - file already exists
```

10. - file already open
    Your assembly language program tried to open a file that is already open. Check your program to see if you are opening the file twice.

11. - file not found

AMOS cannot find the file you've specified. For example:

```
.TYPE LABDAT↵
?Cannot OPEN DSK0:LABDAT.LST - file not found
```

Check your spelling, and make sure that you've specified the correct device and account.

12. - file not open

Your assembly language program has tried to access a file that is not open for input. Check your program to see if you are accessing the correct file.

13. - file type mismatch

You've tried to use a program designed for sequential files on a random file (or vice versa). For example, the TYPE program works only on sequential files; if you try to use it on a random file, you see:

```
.TYPE PRTIDX.DAT↵
?Cannot OPEN DSK0:PRTIDX.DAT - file type mismatch
```

14. - illegal block number

Your program has tried to access a disk block that doesn't exist, or that is beyond the range of the file you are using.

15. - illegal user code

You've tried to access a user account that does not exist. For example, if there is no account [100,2] on DSK1:

```
.TYPE DSK1:MYFILE.TXT[100,2]↵
?Cannot OPEN DSK1:MYFILE.TXT[100,2] - illegal user code
```

16. - insufficient free core

There is not enough room in memory to complete the operation you are attempting. Make sure that no unnecessary modules are in your memory partition.

17. - invalid filename

You've specified a filename that the system does not understand. For example:

```
.MAKE  .TXT↵
?Cannot OPEN  - invalid filename
```

where the filename is a space.

18. - protection violation

You've tried to create a file in an account other than your own that is outside of your project. For example:

```
.COPY [210,3]=ELIPSE.BAS[100,5]
ELIPSE.BAS[100,5] to ELIPSE.BAS[210,3]
?Cannot OPEN ELIPSE.BAS[100,5] - protection violation
```

19. - write protected

You are trying to write data to a device that is write-protected. Write-enable the disk and try again. Make sure that you were writing to the correct disk.

In addition to the error messages above, you can also see:

AM500 ERROR n FOR DRIVE n RECORD n (CYLINDER n HEAD n SECTOR n)

You will see this message if you have SET DSKERR, and a hard error occurs on a system that uses the AM500 Disk Controller Hard Disk Subsystem. This message indicates a hardware problem-- the Disk Controller wasn't able to successfully read data from the disk. The message tells you what kind of error occurred (refer to the information supplied with the disk drive to find out what error conditions corresponds to that error code), and where on the device the data transfer operation failed (the drive, record, cylinder, head and sector). Check with the System Operator for help.

Buss error - PC ##

A buss error indicates that an illegal condition was recognized on the data buss. The number following the letters "PC" tell you the memory address the Program Counter was set to when the buss error occurred. This error can be caused by either software or hardware problems. If you are alone on the system, or other users are at a convenient stopping place, your best course is probably to reset the computer; if it is not convenient to reset the machine, go ahead and continue running.

?Command terminated - insufficient memory

You tried to execute a program from inside a command file, but there was not enough room to load the program into your memory partition. Use the MAP command to see what modules are in your memory partition; delete those you do not need. If you still cannot use the command file, talk to your System Operator about getting more memory area allocated to your job.

ERROR n

You see this message if you SET DSKERR and a soft disk-error occurs on a system using an AM-200 or AM-210 Floppy Disk Subsystem. Frequent soft errors can indicate hardware problems with the drives; check with the System Operator. (The number following ERROR indicates the type of soft error that occurred; refer to the instruction supplied with the disk drive to see what error condition corresponds to that error code.) Because this message

reports a soft error (the system had to retry a data read) and not a hard error (the data could not be read at all), you do not necessarily have to worry unless you see a DEVICE ERROR message (see above), which indicates a hard error.

ERROR n CMD n, STS n, RECORD n (TRACK n, SECTOR n)

You see this message if you have SET DSKERR and a soft disk-error occurs on a system that is using an AM-400 Hard Disk Subsystem. The number following ERROR indicates the type of soft error that occurred; refer to the instructions supplied with the disk drive to find out what error condition corresponds to that error code. The message tells you where on the disk the error occurred. Check with the System Operator.

?File specification error

The format of your command line was confusing. For example:

```
.APPEND ↵
?File specification error
```

Retype the command line, making sure that the syntax you use is the correct form for that particular command.

?Insufficient memory for program load

You do not have enough memory in your partition to load the program you want to execute. Use the MAP command to make sure that no unnecessary modules are in your memory partition. If you still receive this message, check with the System Operator to see if he or she can allocate your job more memory.

?Login please

You've tried to enter an instruction to AMOS, but you are not logged into the system. Use the LOG command (see Section 5.3, Logging into the System). If you need help in figuring out what to do, you can use the HELP command even if you are not logged in (see Section 3.5, The HELP Command).

?Memory allocation failed

You used a monitor call (GETMEM) from within an assembly language program to allocate space for a memory module within a memory partition--there wasn't enough room in the partition to perform the allocation. Make sure that no unnecessary modules are in the memory partition.

?Memory map destroyed

Each module in your memory partition maintains a pointer to the address of the next module in memory; if these connecting links become confused or broken, AMOS is not sure where your memory modules are in your partition. You may not need to reset the computer, but you may want to delete the modules from memory and reload them just to be sure that your memory map is intact.

?No memory available

There is no more free memory available on the system. Consult with the System Operator who may change the amounts of memory allocated to each job on the system.

APPENDIX B

AMOS COMMAND PROCESSING

AMOS follows a specific search procedure when looking for a file in response to a command entered at AMOS command level. Although you usually don't have to think about this search sequence, we provide the information below in case you ever have several files of the same name in different accounts (e.g., DSK0:BACKUP.DO[2,2], DSKU:BACKUP.PRG[1,4] and BACKUP.CMD[100,4]), and want to know which file AMOS will encounter first if you enter just the name of the file without specifying device, account or extension:

```
.BACKUP ↵
```

For example, let's say that you enter the Directory command:

```
.DIR ↵
```

1. AMOS first looks at system memory (the area of memory used by the operating system), to see if the disk file DIR.PRG has already been loaded into system memory; if so, AMOS executes it.

2. If the memory module DIR.PRG is not in system memory, AMOS looks for it in your own memory partitition. If it is there, AMOS executes it.

3. If the memory module DIR.PRG is not in system memory or in your memory partition, AMOS looks for a disk file named DIR.PRG on the System Disk in the System Account (DSK0:[1,4]). If it is there, AMOS loads a copy of the file into your memory partition and executes it.

4. If DIR.PRG does not exist in the System Account, AMOS looks for it in the account and device you are currently logged into.

5. By this time, if AMOS has not found DIR.PRG, it decides that perhaps the file is not a .PRG file (an assembly language program). Now AMOS assumes that the file is a command file. First it looks in your memory partition for a memory module named DIR.CMD; if it is there, it processes the module as a command file.

6. Next AMOS searches for DIR.CMD in the System Command File Library Account (DSK0:[2,2]).

7. If DIR.CMD is not in DSK0:[2,2], AMOS looks for it in the account and device you are currently logged into.

8. If the search for DIR.CMD has failed, AMOS decides that the file must be a DO file. (A DO file is a special kind of command file.) AMOS now begins to look for DIR.DO in your memory partition.

9. AMOS next searches for DIR.DO in the System Command File Library (DSK0:[2,2]).

10. The last place that AMOS searches is the device and account you are currently logged into for DIR.DO.

If AMOS goes through the entire search procedure above without finding DIR, it will tell you that it can't find the file or memory module by repeating the command back to you enclosed in question marks:

```
?DIR   ?
```

You can prevent AMOS from going through this complicated search procedure by giving more information about the file. For example, if you say:

```
.DSK0:DIR.CMD[120,3]
```

AMOS immediately looks for the file on the System Disk in account [120,3], and processes the file as a command file. If AMOS can't find that file, it repeats the entire specification back to you enclosed in question marks:

```
?DSK0:DIR.CMD[120,3]?
```

Index