

Apple II

DOS User's Manual

For II, II+, IIe



Customer Satisfaction

If you discover physical defects in the manuals distributed with an Apple product or in the media on which a software product is distributed, Apple will replace the documentation or media at no charge to you during the 90-day period after you purchased the product.

In addition, if Apple releases a corrective update to a software product during the 90-day period after you purchased the software, Apple will replace the applicable diskettes and documentation with the revised version at no charge to you during the six months after the date of purchase.

In some countries the replacement period may be different; check with your authorized Apple dealer. Return any item to be replaced with proof of purchase to Apple or an authorized Apple dealer.

Limitation on Warranties and Liability

Even though Apple has tested the software described in this manual and reviewed its contents, neither Apple nor its software suppliers make any warranty or representation, either express or implied, with respect to this manual or to the software described in this manual, their quality, performance, merchantability, or fitness for any particular purpose. As a result, this software and manual are sold "as is," and you the purchaser are assuming the entire risk as to their quality and performance. In no event will Apple or its software suppliers be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software or manual, even if they have been advised of the possibility of such damages. In particular, they shall have no liability for any programs or data stored in or used with Apple products, including the costs of recovering or reproducing these programs or data. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Copyright

This manual and the software (computer programs) described in it are copyrighted by Apple or by Apple's software suppliers, with all rights reserved. Under the copyright laws, this manual or the programs may not be copied, in whole or part, without the written consent of Apple, except in the normal use of the software or to make a backup copy. This exception does not allow copies to be made for others, whether or not sold, but all of the material purchased (with all backup copies) may be sold, given or loaned to another person. Under the law, copying includes translating into another language.

You may use the software on any computer owned by you but extra copies cannot be made for this purpose. For some products, a multi-use license may be purchased to allow the software to be used on more than one computer owned by the purchaser, including a shared-disk system. (Contact your authorized Apple dealer for information on multi-use licenses.)

Product Revisions

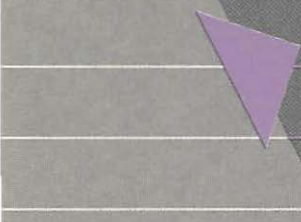
Apple cannot guarantee that you will receive notice of a revision to the software described in this manual, even if you have returned a registration card received with the product. You should periodically check with your authorized Apple Dealer.

© 1983 Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, California 95014

Apple and the Apple logo are registered trademarks of Apple Computer, Inc.
Simultaneously published in the U.S.A. and Canada. All rights reserved.

Apple II

DOS User's Manual



Contents

Read Me First!

ix

- xi Introducing DOS
- xi What Do You Need to Use DOS?
- xii The DOS Manuals
- xii The DOS Disks
- xii How This Manual Is Organized
- xiv A Few Special Notes

A Tutorial for the New DOS User

1

- 4 Getting Started
- 4 About Disks and Disk Drives
- 6 Caring for Disks
- 7 Inserting a Disk Into a Disk Drive
- 9 Removing a Disk From a Disk Drive
- 10 Starting DOS
- 12 Solving Startup Problems
- 13 What Starting Up Is
- 14 Restarting DOS
- 14 Looking at the Contents of a Disk: the CATALOG Command
- 17 Copying the SYSTEM MASTER Disk
- 18 Copying With One Disk Drive
- 24 Copying With Two Disk Drives
- 27 If You Have Copying Problems
- 28 Testing a Copied Disk
- 31 Preparing a Blank Disk: the INIT Command
- 34 Using Programs
- 34 Reading Information From a Disk: the LOAD Command
- 36 Changing the Contents of Memory
- 38 Saving Information on a Disk: the SAVE Command
- 40 Running a Disk Program: the RUN Command
- 41 Turning Off the Computer
- 42 Chapter 1 Summary

An Overview of DOS

43

- 45 How DOS Works
- 46 The Monitor Program
- 46 The Startup Process
- 50 Files and Volumes
- 50 Files
- 52 Volumes
- 53 DOS Commands
- 53 Syntax
- 55 The Arguments
- 59 Chapter 2 Summary

Using Disks

61

- 63 The CATALOG Command
- 65 Understanding a Catalog
- 66 The Copy Programs
- 66 The COPYA Program for Applesoft BASIC
- 67 The COPY Program for Integer BASIC
- 68 Error Messages From the Copy Programs
- 69 Preparing Disks
- 69 Initialized Disks
- 69 Master Disks
- 70 Greeting Programs
- 71 The INIT Command
- 72 Determining Slot and Drive Numbers: The SLOT# Program
- 73 Chapter 3 Summary

Using Files

75

- 77 The RENAME Command
- 80 The LOCK Command
- 81 The UNLOCK Command
- 82 The DELETE Command
- 83 The VERIFY Command
- 84 The FILEM Program
- 85 Specifying Slot and Drive Numbers
- 85 Specifying Filenames With the Wildcard Character
- 87 Using FILEM Options
- 95 FILEM's Messages
- 97 Chapter 4 Summary

Using Programs Already on Disks

99

- 102 The RUN Command
- 104 The LOAD Command
- 105 The SAVE Command
- 107 Talking to Other Devices
- 108 The PR# Command
- 110 The IN# Command
- 111 Ways to Restart DOS
- 111 Conclusion
- 112 Chapter 5 Summary

Dealing With 13-Sector Disks

117

- 118 Converting a 13-Sector Disk: the CONVERT13 Program
- 118 Example
- 122 Duplicate Filenames
- 122 The Wildcard Character
- 123 Running 13-Sector Disks Without Conversion
- 123 Using the START13 Program
- 124 Using the BASICS Disk

Warnings and Error Messages

127

- 128 DOS Messages
- 128 DISK FULL
- 128 END OF DATA
- 129 FILE LOCKED
- 129 FILE NOT FOUND
- 130 FILE TYPE MISMATCH
- 130 I/O ERROR
- 131 LANGUAGE NOT AVAILABLE
- 132 NO BUFFERS AVAILABLE
- 132 NOT DIRECT COMMAND
- 133 PROGRAM TOO LARGE
- 133 RANGE ERROR
- 134 SYNTAX ERROR
- 134 VOLUME MISMATCH
- 134 WRITE PROTECTED
- 135 Stopping a Program

Programs on the DOS Disks

137

- 137 Programs on the SYSTEM MASTER Disk
- 139 Programs on the SAMPLE PROGRAMS Disk

Summary of DOS Operating Concepts and Commands 143

143	Operating Concepts
143	Starting Up
144	Restarting DOS
144	Capacity
145	Notation
145	Syntax
146	Arguments
147	Command Summary
147	CATALOG
147	DELETE
148	IN#
148	INIT
149	LOAD
149	LOCK
150	PR#
150	RENAME
150	RUN
151	SAVE
151	UNLOCK
151	VERIFY

Glossary 153

Index 167

Figures and Tables

Chapter 1

- 5 Figure 1-1. A Disk, Plain and Simple
- 6 Figure 1-2. Putting a Write-Protect Tab on a Disk
- 7 Figure 1-3. Four Risky Ways to Handle a Disk
- 7 Figure 1-4. Opening the Disk Drive Door
- 8 Figure 1-5. Removing the Disk From Its Envelope
- 8 Figure 1-6. Inserting the Disk Into the Drive
- 9 Figure 1-7. Closing the Disk Drive Door
- 10 Figure 1-8. Turning On the Power
- 11 Figure 1-9. The Opening Display
- 16 Figure 1-10. The Catalog of the SYSTEM MASTER Disk
- 20 Figure 1-11. The Slot and Drive Numbers for Copying
With One Disk Drive
- 26 Figure 1-12. The Slot and Drive Numbers for Copying
With Two Disk Drives

Chapter 2

- 47 Figure 2-1. The Opening Display
- 48 Figure 2-2. The Startup Process
- 51 Figure 2-3. Files in a Catalog
- 53 Figure 2-4. The Syntax of DOS Commands

Chapter 3

- 64 Figure 3-1. The Catalog of the SAMPLE PROGRAMS Disk
- 65 Table 3-1. File Types

Chapter 4

- 84 Figure 4-1. FILEM's Menu
- 86 Figure 4-2. Using the Wildcard to Represent
Parts of Filenames

Chapter 5

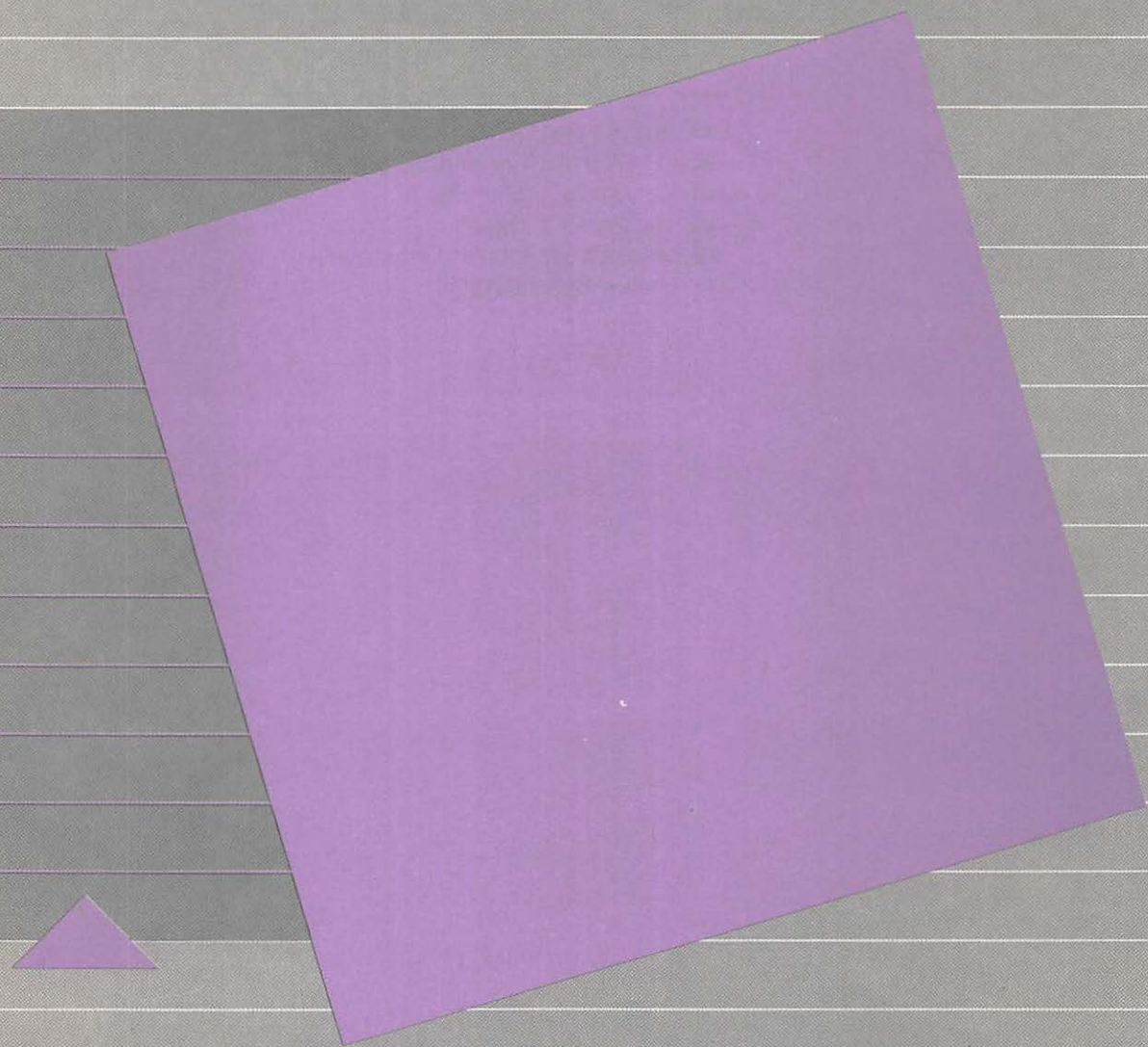
- 101 Figure 5-1. Using DOS Commands With BASIC Programs
- 107 Figure 5-2. The PR# and IN# Commands

Appendixes

- 117 Figure A-1. 13-Sector Versus 16-Sector Disks
- 119 Figure A-2. The CONVERT13 Menu
- 123 Figure A-3. The START13 Title Screen
- 128 Table B-1. DOS Error Messages
- 130 Table B-2. Types of Files According to Command
- 133 Table B-3. Minimum and Maximum Argument Values

Read Me First!

-
- xi** Introducing DOS
 - xi** What Do You Need to Use DOS?
 - xii** The DOS Manuals
 - xii** The DOS Disks
 - xii** How This Manual Is Organized
 - xiv** A Few Special Notes



Read Me First!

A **disk** can be either rigid or flexible. A **flexible**, or **floppy**, **disk** is a thin, flat circle of magnetized material on which information from your Apple II computer can be recorded.

A **kilobyte**, or 1K, consists of 1024 bytes, or 8192 bits. A **byte** consists of eight **bits** and can hold any value from 0 to 255.

Random-access memory is memory that you can change. When the computer is turned off, the contents of RAM are erased.

An **application program** puts the resources and capabilities of the computer to use for some specific purpose or task, such as word processing. DOS is a **system program**. While an application program is used for specific tasks, a system program makes the computer available for general purposes.

A **disk drive** writes and reads information on the surface of a magnetic disk.

Introducing DOS

A disk operating system is a set of computer programs that manages information on **disks**: it lets you store information on disks and read the information back into your computer.

DOS is a specific disk operating system for the Apple II. It is a manager of information. The information can take the form of programs, which perform particular jobs, or data, which can be a collection of names or numbers. DOS does not do specific tasks, such as helping you create a business letter, keep track of an inventory, or create a graph that shows your weight loss. DOS does allow you to store and manage such information after you have created it.

DOS 3.3 organizes the space on a disk into 16 sectors. To use a 13-sector disk with 16-sector DOS, see Appendix A.

It's a Toss Up: Most people pronounce DOS so it rhymes with "toss." Others spell it out: DEE OH ESS. Use the one that comes easier to you; they're both right.

What Do You Need to Use DOS?

Before continuing, you should set up your Apple II computer. See the owner's manual that came with your computer for information on setting up your system.

To use DOS, your Apple II computer should have a minimum of 16 **kilobytes**, or K, of **random-access memory**, called RAM for short. DOS uses about 10.5K of memory. To have enough room for DOS and an **application program**, 48K is recommended.

Your Apple II computer also must have at least one **disk drive**. To connect your disk drive to your Apple II, follow the instructions that came with your disk drive.

The DOS Manuals

The *DOS User's Manual* will guide you step by step through the basics of using DOS. If you are learning to program or want to run ready-made programs or games, this manual will give you the information you need.

After you've read the material that came with your Apple II computer, you are ready for this manual.

The *DOS Programmer's Manual* describes more sophisticated DOS commands and how to work with complex BASIC programs. After you've read this manual and had some experience programming, you'll be ready for the programmer's manual.

The DOS Disks

The disk labeled DOS 3.3 SYSTEM MASTER contains the DOS programs that manage disks and their files. When you use the SYSTEM MASTER, the commands and capabilities of DOS are added to the other computing capabilities that are already available with your Apple II computer.

The disk labeled DOS 3.3 SAMPLE PROGRAMS contains demonstration programs. You will use this disk to learn the DOS commands outlined in this manual.

In addition, the SAMPLE PROGRAMS disk contains examples of BASIC programs. For the most part, only a person who is writing BASIC programs will want to look at them.

How This Manual Is Organized

This manual is for a person who is beginning to use a computer: the user who will be running programs that are on disks or writing a simple BASIC program.

Chapter 1, "A Tutorial For New DOS Users," is designed for you if you are a new computer user. If you already have some experience with computers, you can begin with Chapter 2 and refer to Chapter 1 only occasionally.

Each chapter in this manual begins with an introduction to the topics presented in the chapter. The examples in each chapter are for you to work through; trying them will help you learn to use DOS. The end of each chapter has a summary of the commands and concepts that are covered.

Here is an overview of what this manual contains.

- Chapter 1, "A Tutorial for the New DOS User," includes all you'll need to be using DOS within a few hours. Specifically, it tells you how to insert a disk into a disk drive, how to start up the computer, how to copy a disk for safekeeping, how to prepare a disk to receive information, how to work with files on a disk, and how to turn the computer off.

In addition to having a starter set of commands, Chapter 1 introduces some basic computer concepts and techniques, including what a file is, how to load and run a program, how to save a file, and how to deal with the unexpected.

- Chapter 2, "An Overview of DOS," tells what you need to know to use DOS. It describes files and how to name them. It explains the rules for using DOS commands.
- Chapter 3, "Using Disks," deals with disks as a whole—finding out what is already on a disk, copying an entire disk, and preparing a new disk to receive information.
- Chapter 4, "Using Files," goes into managing individual files: keeping track of them, renaming them, protecting them from accidental erasure, and deleting them.
- Chapter 5, "Using Programs Already on Disks," discusses using application programs: how to run them, and how to save the data that they produce. This chapter also covers getting DOS to communicate with another device, such as a printer.
- Appendix A, "Dealing With 13-Sector Disks," guides a user who has 13-sector equipment. It describes how to run 13-sector disks and how to convert 13-sector disks to a 16-sector format.
- Appendix B, "Warnings and Error Messages," discusses several sources of warnings and error messages. It lists each error message from DOS and suggests ways to fix the problem that caused the message.
- Appendix C, "Programs on the DOS Disks," briefly describes each program on the SYSTEM MASTER and the SAMPLE PROGRAMS disks.
- Appendix D, "Summary of DOS Operating Concepts and Commands," summarizes DOS. You can use this section for quick reference later.
- The Glossary defines computer terms with which you may be unfamiliar. These terms are boldfaced when they are introduced in the text.

A Few Special Notes

In this manual, Apple II refers to every model of the Apple II computer: the Apple II, the Apple II Plus, and the Apple IIe. When the information applies only to a specific model, the manual says so.

The first time computer terms are used in this book, they are defined and boldfaced. The terms are also defined in the Glossary.

The DOS manuals use some special visual aids:

A special type is used for what you see on your display:
IT LOOKS LIKE THIS.

What you should type looks like the keys on the computer's keyboard. For instance, when you see

R U N SPACE C O L O R SPACE T E S T

you should press the corresponding keys on your Apple II. Although the **SPACE** bar isn't labeled on your keyboard, it is important to know when you must type a space character, so it is treated as a key.

When you should press a key that is labeled with two characters, only the character you should type is indicated. And **SHIFT** is included when you must press that key to get the desired character.

When you see a hyphen joining two keys, it means to press the keys simultaneously. For instance, **CONTROL-RESET** means you should press **CONTROL** and **RESET** at the same time. In actual practice, you probably will press **CONTROL** first and then, while still holding down **CONTROL**, press **RESET**.

By the Way: The gray box provides a reminder or additional clarification—a faster or better way to do something.



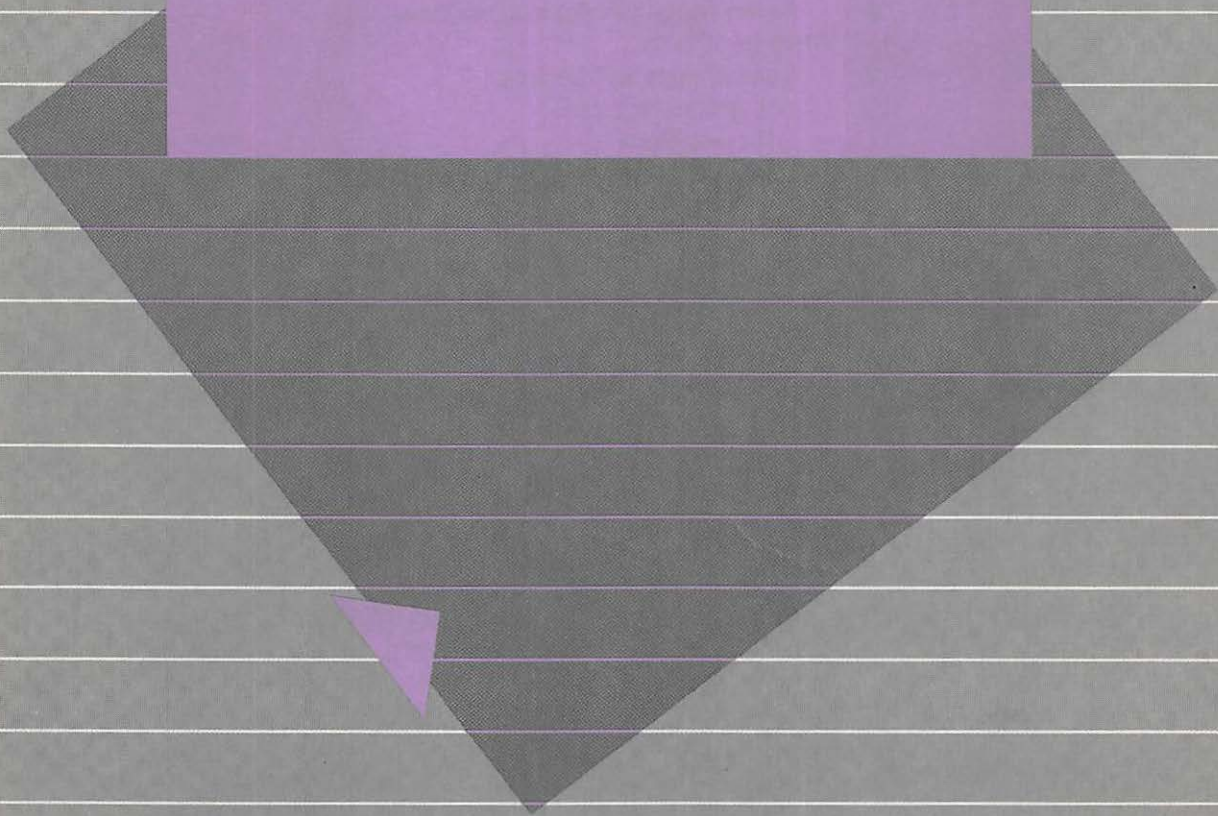
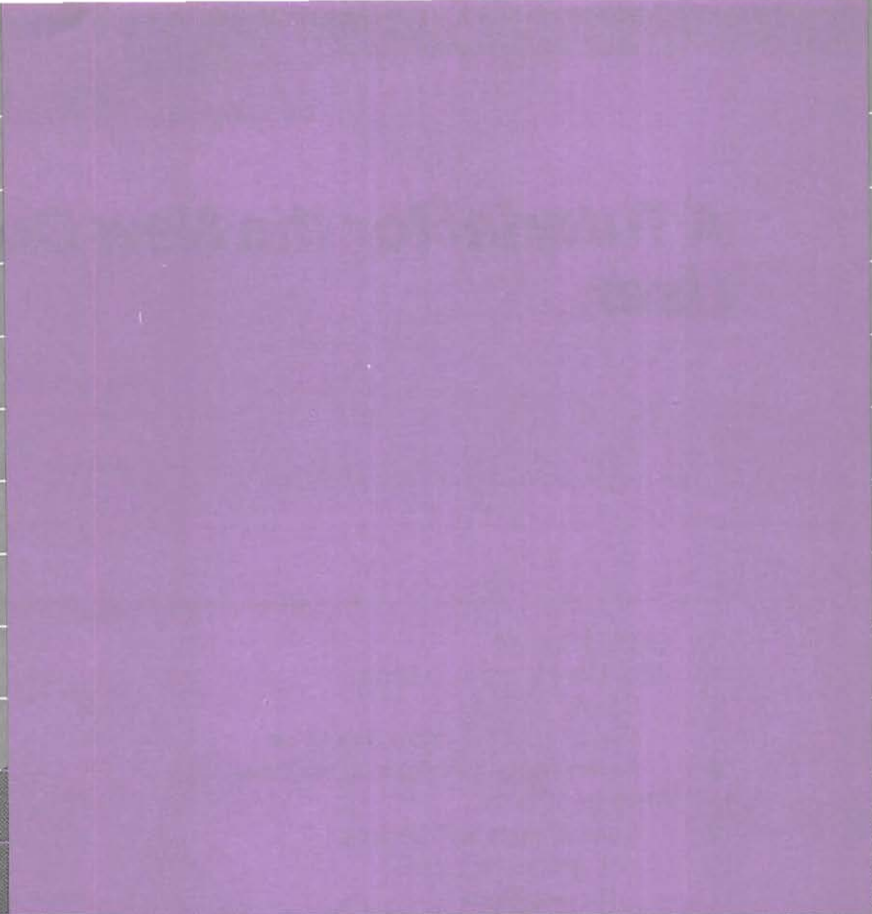
Warning

A box like this indicates a potential problem or disaster. The information in the box describes the danger and suggests ways to avoid it.

Notes in the margins reinforce new terms or point to useful information contained in other manuals.

A Tutorial for the New Dos User

4	Getting Started
4	About Disks and Disk Drives
6	Caring for Disks
7	Inserting a Disk Into a Disk Drive
9	Removing a Disk From a Disk Drive
10	Starting DOS
12	Solving Startup Problems
12	Lights or No Lights
12	Screen Blank
12	No Messages or Mysterious Messages
13	Strange Sounds
13	What Starting Up Is
14	Restarting DOS
14	Looking at the Contents of a Disk: the CATALOG Command
17	Copying the SYSTEM MASTER Disk
18	Copying With One Disk Drive
24	Copying With Two Disk Drives
27	If You Have Copying Problems
28	Testing a Copied Disk
31	Preparing a Blank Disk: the INIT Command
34	Using Programs
34	Reading Information From a Disk: the LOAD Command
36	Changing the Contents of Memory
38	Saving Information on a Disk: the SAVE Command
40	Running a Disk Program: the RUN Command
41	Turning Off the Computer
42	Chapter 1 Summary



A Tutorial for the New DOS User

This chapter introduces the group of DOS commands you're likely to use over and over. It briefly describes what a disk is and how to care for it, what a file is, what it means to run and to save a program, and a little about how to deal with the unexpected.

This chapter is presented as a tutorial. Each exercise begins with a quick overview of the task. Then the task is presented step by step.

Start at the beginning, read and try each step of each task as it comes along. Go through each of the procedures in succession. By the end of the tutorial, you'll know:

- how to care for disks
- how to insert a disk into a disk drive
- how to start up the computer
- how to copy a disk for safekeeping
- how to prepare a disk to receive information
- how to get information from a disk
- how to put information on a disk
- how to run a program
- how to turn the computer off

If you have not yet read your owner's manual or the introductory material that came with your Apple II computer, take the time to read that background material now.

By the Way: The preceding section, "A Few Special Notes," describes the special notations, such as this box, that are used in this manual.

DOS, for Disk Operating System, is a manager of your disks and the information on them.

A **program** is a set of instructions describing actions for the computer to perform to accomplish some task.

Getting Started

Learning to use **DOS** means learning a few special commands, like the command **CATALOG**; a few operations, like inserting and removing a disk from a disk drive; and a few new words, like *load* and *run*. These words are not meant to mystify or impress you. In every case, the words are technical terms for which there is no single, everyday word.

DOS helps you control information on disks. That control means you can store information on disks and read the information back into your computer. What the information is, is up to you. It can be a **program** that performs a particular job for you, data such as a collection of names or numbers, or just about whatever you like.

As you work through this tutorial and learn about DOS, you should have in front of you:

- an Apple II computer with one or two disk drives
- the DOS 3.3 SYSTEM MASTER disk
- the DOS 3.3 SAMPLE PROGRAMS disk
- at least two blank disks

The DOS 3.3 SYSTEM MASTER disk is very special. It has the programs that make up DOS. You'll be using the SYSTEM MASTER disk as you do this tutorial.

The SAMPLE PROGRAMS disk has demonstration programs and games. This chapter will introduce one of its programs to you.

About Disks and Disk Drives

A **disk drive** reads from and writes to the surface of a magnetic disk.

A **disk drive** is a little smaller than a shoe box. If you could see through its covering, you'd see that it is a magnetic storage device with two mechanisms. One mechanism spins the disk. The other mechanism reads from and writes to the surface of the disk.

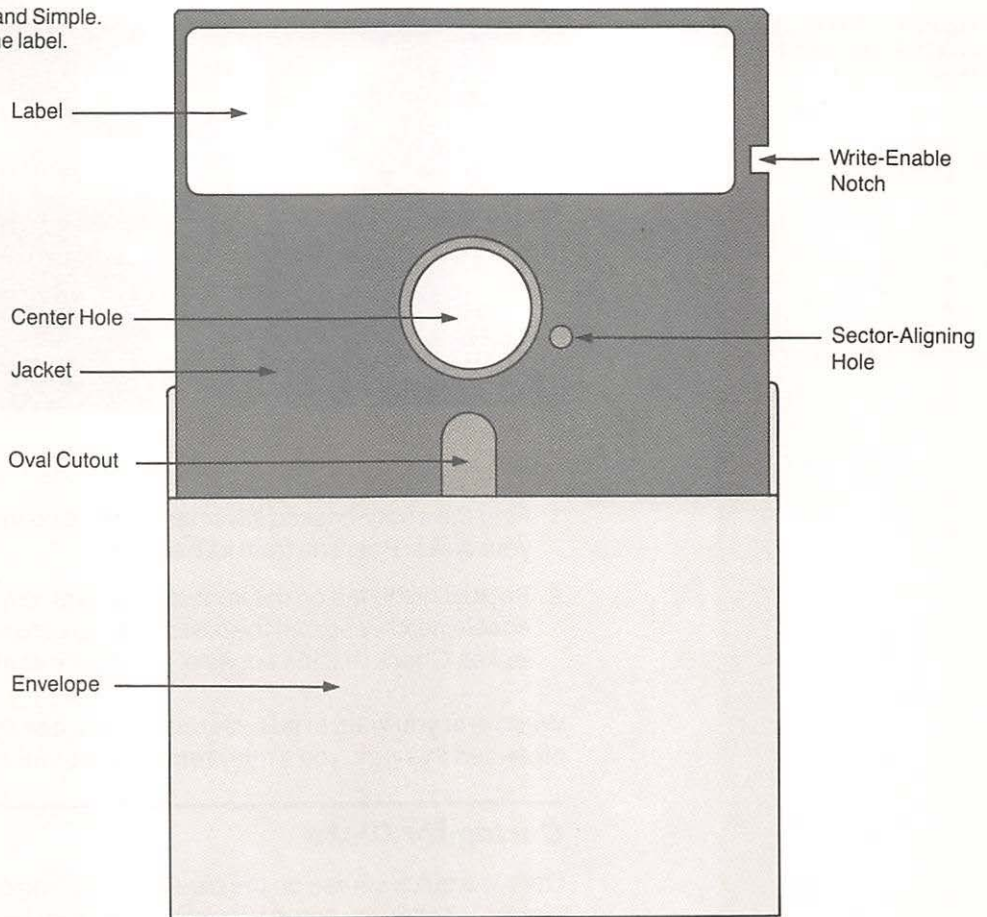
When the disk drive picks up information from the disk, it functions like a record player. However, the information that is picked up from the disk goes to the computer rather than to a speaker. A disk drive also can record, or write, information from the computer to the disk.

A **flexible**, or **floppy**, **disk** is used to store information.

A **flexible disk** is small, round, flat, plastic, and about five inches across; it has a hole in the center. Sometimes a flexible disk is called a **floppy disk**.

The disk is coated so that information can be stored on its surface. The coating is similar to the coating on magnetic recording tape. Be careful to protect this coated surface whenever you handle a disk.

Figure 1-1. A Disk: Plain and Simple. Notice the placement of the label.



A **disk jacket** is a permanent protective covering for a disk. It is usually made of black plastic.

A **write-enable notch** permits information to be written on the disk. If there is no notch, or it is covered with a write-protect tab, information can be read from the disk but not written onto it.

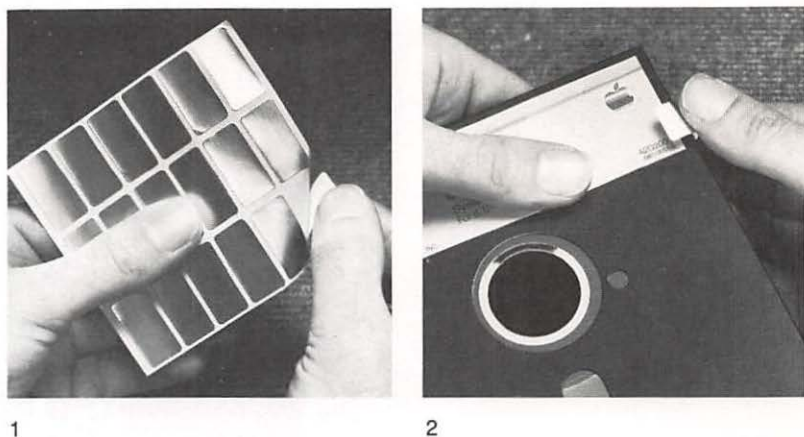
Normally, you can see only a little of the recording surface of the disk because the disk is enclosed in a square, black, plastic **jacket** (Figure 1-1). The jacket, which lubricates and protects the disk, has two cutouts. The larger oval one (about one inch long) is the window through which information passes. The smaller square one, the **write-enable notch**, allows you to change information on the disk. Without the notch, the computer can only read the information on the disk.

Some disks (like the SYSTEM MASTER) do not have a write-enable notch to protect the important information on them.

Silver **write-protect tabs** come with disks and are used to prevent accidental erasure of information on the disk

Figure 1-2. Putting a Write-Protect Tab on a Disk. The numbers correspond to those in the text.

To protect the information on your disks from accidental erasure, you can cover the write-enable notch with a **write-protect tab**. Figure 1-2 illustrates the steps to protect a disk with a write-protect tab.



1. Find the sticky-backed silver tabs that came in the package with your disks. Peel one from its backing.
2. Put half of the tab on the front of the jacket, covering the write-enable notch, and fold the rest of the tab onto the back of the jacket. Check that the edges of the tab are smooth.

Whenever you want to add information to a disk that is write-protected this way, you simply remove the silver tab.

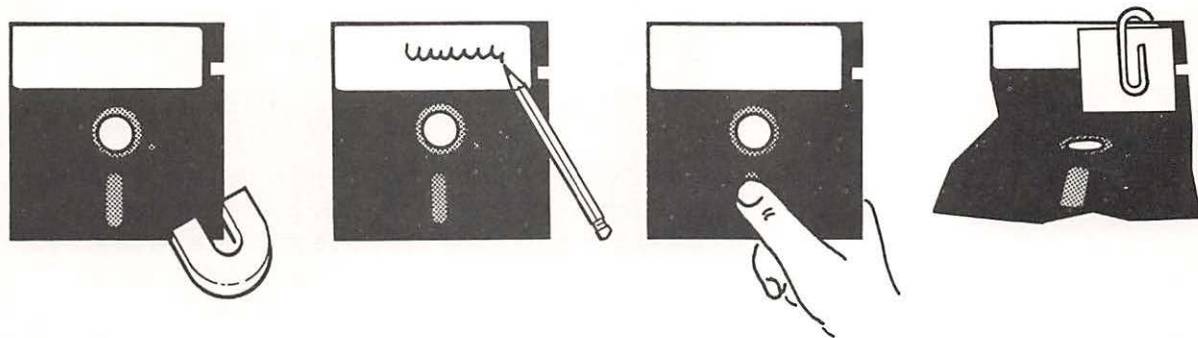
Caring for Disks

Here is a quick course on the care and handling of disks. This list and Figure 1-3 give you some very important guidelines.

- Handle a disk by its jacket and label. Never remove the disk from its jacket. Never touch the surface of the disk itself. Be especially careful when labeling a disk not to touch the surface through the oval cutout.
- To write on a label already attached to a disk, use a *felt-tip* pen. Do not press hard. It is better to write on the label before attaching it to the disk.
- Do not write on an attached label with a pencil or ball-point pen. Doing so may put dents in the recording surface.

- Do not use an eraser on the label. Eraser dust is abrasive and can damage the disk.
- When you're not using a disk, keep it in a paper envelope.
- Store disks upright in their envelopes. Do not bend disks, put them into a typewriter, or attach paper clips to them.
- Store disks away from direct sunlight, moisture, and extremes of heat and cold.
- Keep disks away from magnets and electrical devices, especially television sets and large motors. It is all right to put disks temporarily on the computer or disk drive.

Figure 1-3. Four Risky Ways to Handle a Disk



Inserting a Disk Into a Disk Drive

- Lift the small door on drive 1 as shown in Figure 1-4. If there is anything in the drive, carefully pull it out and put it in a disk envelope.

Figure 1-4. Opening the Disk Drive Door



- Remove the DOS 3.3 SYSTEM MASTER disk from its envelope (Figure 1-5).

Figure 1-5. Removing the Disk From Its Envelope



- Holding the disk with your thumb on the label, gently insert the disk into the horizontal slot of drive 1. The disk goes in oval cutout first, label up (Figure 1-6). Be careful not to bend or force the disk. If you feel any resistance, slowly pull the disk out of the drive and try again.

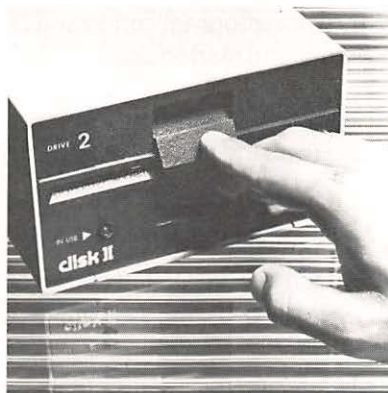
If a disk has no label, hold the disk so the write-enable notch is on the left.

Figure 1-6. Inserting the Disk Into the Drive



- When the disk is all the way inside, push the small door on the disk drive down until it clicks shut (Figure 1-7).

Figure 1-7. Closing the Disk Drive Door



Removing a Disk From a Disk Drive

When a disk drive's light is on, DOS is reading or writing to the disk. Before you remove any disk from a disk drive, make sure the drive's light is off. Then just open the drive door and gently pull the disk out. Try it now.

1. Lift the disk drive door.
2. Gently remove the SYSTEM MASTER disk from the drive.

Warning

Never remove a disk while the drive's light is on. Doing so may damage the disk or garble some of the information on it.

Sometimes a disk drive won't stop whirring. You can stop a runaway disk by pressing **CONTROL** and **RESET** at the same time.

Main memory is the part of the Apple II computer system that you will use to store information.

A **display device** exhibits information visually.

Starting DOS

To use DOS, you have to turn on the computer and read DOS from a disk into **main memory**. Reading DOS into memory means that a copy of the program containing DOS is put into the computer's memory and started up.

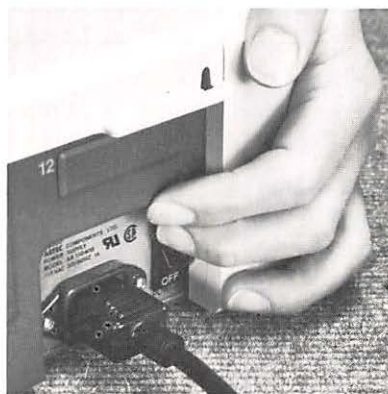
1. Turn on your **display device**, either a video monitor or a television set (with the volume down), so it can warm up.
2. Check that the SYSTEM MASTER disk is properly inserted in drive 1 and that the drive door is closed.
3. Reach behind the left corner of the computer case and find the power switch with your left hand (Figure 1-8). Turn your computer on.

The power indicator on the keyboard and the small light on the front of drive 1 will light up. The disk drive will make whirring sounds as it reads the disk. After a few seconds, the sounds will stop, and the red light on the disk drive will go out.

If you have trouble, check the next section, "Solving Startup Problems," for what to do.

The SYSTEM MASTER disk must be in drive 1 to start DOS no matter how many drives are connected.

Figure 1-8. Turning On the Power



When you start DOS successfully, your display screen will look like Figure 1-9. This display indicates that DOS is in memory and is ready for your commands. If you don't see this display, the next section will help you figure out what to do.

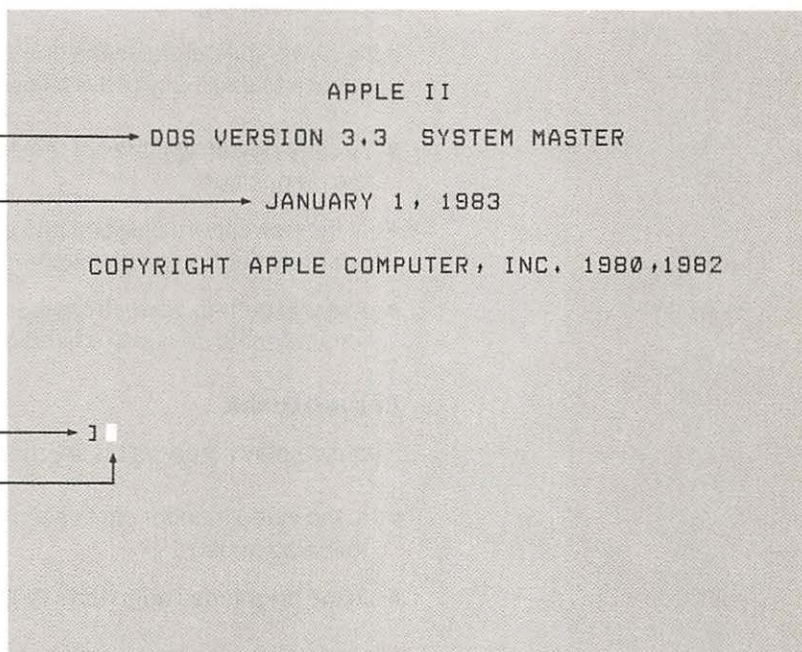
Figure 1-9. The Opening Display

The DOS version number, and the name of the startup disk.

The actual date of the system program; yours may be different.

The Applesoft BASIC prompt.

The cursor.



A Pressing Matter: You'll see the message `BE SURE CAPS LOCK IS DOWN` if you start DOS on an Apple IIe computer. Press `(CAPS LOCK)` until it clicks into its on, or down, position.

If you have an Apple II or Apple II Plus, you won't see this message, and you don't need to worry about `(CAPS LOCK)`.

Solving Startup Problems

If the startup procedure didn't work for you, this section will tell you what to investigate. The solution to the problem is usually simple.

Lights or No Lights

If the power indicators on the computer or display device didn't light up, check to see if any of the following apply.

- Are the power switches of the display device and the computer in the *on* position?
- Is the power cord plugged into the wall socket? Is the power cord securely connected to the computer?
- Are you getting power from the wall socket? (Test the outlet with a small electric device like a radio.)

Screen Blank

If your display screen is blank, check the following.

- Is the video monitor cable securely connected to the monitor and to the computer?
- Is the brightness adjusted? What about the contrast?

No Messages or Mysterious Messages

If you see nothing about DOS on your display screen (only `APPLE II` or perhaps an entirely different message), the computer failed to bring DOS into the computer from the disk.

- Make sure the disk drive is connected properly.
- Check that the SYSTEM MASTER disk is in drive 1.
- If you have two disk drives and the empty drive keeps spinning (the red light stays on, and the drive makes whirring sounds), turn off the computer to stop the drive. Your disk drives are labeled incorrectly. Switch the labels on the drives; or switch the drives. Move the SYSTEM MASTER from drive 2 to drive 1.
- If you see an asterisk (*) on your display, type

`[6] [CONTROL]-[P]`

to start DOS.

Read-only memory, or ROM, is memory whose contents can only be read. Information is written into read-only memory once, during manufacture. The information is stored permanently and can never be erased or changed.

See Appendix A, "Dealing With 13-Sector Disks," for further assistance with disks that cause your disk drive to continue whirring.

If you see an asterisk (*) when you turn on your Apple computer, it means you are in the Monitor program and don't have an autostart ROM. You will have to remember to type **⌘ (CONTROL)-P** whenever you want to start DOS. To obtain an autostart ROM, see your dealer.

Strange Sounds

If the drive keeps **whirring**, making strange, dragging noises, try opening and then closing the drive door. Make sure that it clicks shut. If you have been successful, the drive will make different sounds as it reads the disk.

If the disk drive keeps spinning, stop the drive by pressing **⌘ (CONTROL)** and **⌘ (RESET)** at the same time. See the **Warning** box. When the drive's red light goes out, take the disk out, then put it back in. Make sure the disk is seated properly in the drive, not curled or on an angle.



Warning

Do not try to remove a disk from a drive while its light is on. You may damage the disk.

To stop a runaway disk drive, press **⌘ (CONTROL)** and **⌘ (RESET)** at the same time.

After you have followed the suggestions for solving your problem, go back to "Starting DOS" and try again.

What Starting Up Is

Now that you have started up DOS successfully, perhaps you're wondering what starting up is and what it does. When you start up DOS, you add DOS's commands and capabilities to the computing capabilities that are already available with your Apple II computer. The DOS commands are read into memory from the SYSTEM MASTER disk.

A **prompt character** shows that the computer is ready for your command. It is so named because it prompts, or reminds, you that some action is needed.

Once in memory, DOS starts BASIC, and BASIC prints a **prompt character** (often called a **prompt**) on the screen. The prompt lets you know that the computer is ready for your command.

When Applesoft BASIC is running, you will see a square bracket (**[**) prompt. When Integer BASIC is running, you will see a greater than (**>**) prompt.

A **cursor** marks where your next action will take effect or where the next character you type from the keyboard will appear on the display screen.

Next to the prompt is a **cursor**. The cursor tells you where your next command will take place or where the next character you type will appear on the display screen. The cursor may be a square, a rectangle, a checkerboard box, or a bar; it may blink, or it may glow steadily. How it appears depends on the variety of Apple II you are using, and the screen width capabilities of your system.

Restarting DOS

There may be times when you will want to restart DOS. For instance, you will want to restart DOS when your program is doing strange things and you can't figure out how to get back to DOS or the prompt; or when the monitor is displaying erratic patterns; or when a disk drive spins continuously.

When the power to your computer is on, you can restart DOS by holding down **CONTROL** and, at the same time, pressing **RESET**. Then release the keys. DOS is restarted.

Try it now. You'll hear a beep and then see BASIC's prompt. The prompt tells you that you've restarted DOS.

When you start DOS using **CONTROL-RESET**, the prompt lets you know that DOS is ready for your commands. DOS doesn't display the screen shown in Figure 1-9.

This method interrupts the computer and resets it to its original state. It saves time and extends the life of the computer parts.

If you have a program in memory, this method usually restarts DOS without losing your program, whereas turning the computer off and then back on always loses what is in main memory.

Looking at the Contents of a Disk: the CATALOG Command

Once you've started DOS, you can look at the contents of a disk. The contents are listed in the disk's **catalog**. The catalog is simply a list of filenames and some information about each file.

A Note About Spaces: Before you start typing commands and using DOS, you need to know about how the computer understands what you type. The computer takes everything you tell it literally, and this includes the space character. That's right, a space is a character. And when you want one, you have to press the **SPACE** bar. Moving the cursor with the arrow keys does not insert space characters.

A **catalog** is a list of all the files stored on a disk.

The CATALOG command lists the contents of a disk.

For more on the Apple IIe keyboard, see the owner's manual.

What You Do...

1. Make sure the SYSTEM MASTER disk in drive 1 and DOS is started.

2. Type **CATALOG**

Look at your display and check what you've typed. As long as you have not pressed **RETURN**, you can correct a typing mistake by pressing **←** to move the cursor back to the mistake. Then retype the command.

Press **RETURN** when you have the command typed correctly.

What You Get...

You will see BASIC's prompt on your screen.

If you misspell the command and press **RETURN**, your computer will beep, print **SYNTAX ERROR**, and display the prompt. Nothing terrible happens. Retype the command and check the spelling before you press **RETURN**.

After a few seconds, you'll see the SYSTEM MASTER's catalog on your display.

All DOS and BASIC commands must be given in uppercase. If you are using an Apple IIe, pressing the **⌈CAPS LOCK** will give you an uppercase keyboard (you still have to use **⇧** to get the upper-position characters on double keys).

If you get a **SYNTAX ERROR** message, it may be because you typed the command in lowercase. Press **⌈CAPS LOCK** until it clicks into its on, or down, position and try the command again.

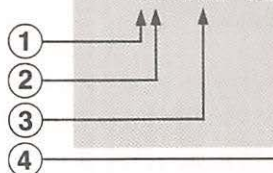
Apple II and II Plus users automatically have an uppercase keyboard.

3. Press the **SPACE** bar to see the rest of the catalog.

Look at your screen. It will look like Figure 1-10.

Note: Whenever a disk's catalog is longer than one screen, you must press another key to continue the catalog. Pressing the **SPACE** bar to do this is a good habit to get into.

Figure 1-10. The Catalog of the SYSTEM MASTER Disk. The numbers correspond to those in the text.



DISK VOLUME 254			
*A	003		HELLO
*I	003		APPLESOFT
*B	006		LOADER.OBJ0
*B	042		FPBASIC
*B	042		INTBASIC
*A	003		MASTER
*B	009		MASTER CREATE
*I	009		COPY
*B	003		COPY.OBJ0
*A	009		COPYA
*B	003		CHAIN
*A	014		RENUMBER
*A	003		FILE
*B	020		FID
*A	003		CONVERT13
*B	027		MUFFIN
*A	003		START13
*B	007		BOOT13
*A	003		SLOT#

Figure 1-10 shows the catalog of the SYSTEM MASTER disk. An explanation of the catalog follows.

A **file** is a collection of information stored on a disk. That information can be text, data, or a program.

To **lock** a file means to protect it from being accidentally changed or erased. To **unlock** a file means to remove that protection.

Chapter 4 discusses how to **unlock** files.

For more on **file types**, see Chapter 3.

- ① The first column shows whether the **file** is **locked** or not. An asterisk (*) means the file is locked. You cannot change the contents of a locked file until you unlock it.

A file that is not locked has a space in the first column instead of an asterisk. A space indicates that the file is **unlocked** and that you can change its contents.

- ② The second column shows the **file type**. The file types that you see on the SYSTEM MASTER are

- A for “Applesoft” BASIC
- I for “Integer” BASIC
- B for “binary”

File type indicates what language environment you need to be in and what commands you need to use to access the files. Certain DOS commands work only on certain types of files. You will only be working with Applesoft and Integer files in this manual.

Sectors are explained in Appendix B of the *DOS Programmer's Manual*.

Appendix C describes each file on the SYSTEM MASTER.

You should make a **backup** copy of all your disks just in case something happens to the original.

A **system disk** is one that holds the system or application program that runs the system.

③ The third column tells you how big the file is. For example, the file HELLO is three **sectors** long. A three-sector file is a small one; a 110-sector file is a large one.

④ The fourth column indicates the name of the file. You'll be using some of these files later on.

You will use the programs on the SYSTEM MASTER disk again and again to manage your disk files. Later in this chapter, you will learn how to copy this disk.

Copying the SYSTEM MASTER Disk

This part of the tutorial shows you how to copy the entire contents of the SYSTEM MASTER disk and how to test the copy.

The process of making a copy of a disk is called **backing up**. Once you have made a **backup** copy of the SYSTEM MASTER, you should store the original in a safe place and use the copy.

You should do this with all your **system disks**. Then if something happens to the copy, you will still have the original from Apple Computer, Inc.

It is also a good idea to protect all your system disks by covering the write-enable notch with a silver write-protect tab. See "About Disks and Disk Drives" earlier in this chapter for more information.

To make a copy of a disk, you will use a copy program on your SYSTEM MASTER disk.

The copy program automatically prepares a blank disk to receive information. Then it reads as much information as it can from the original disk, putting that information temporarily into main memory. The program then writes, or copies, that same information onto the blank, or duplicate, disk. Then the program reads more information from the original disk into memory before copying it to the duplicate disk.

There can be so much information on a single disk that the copy program might repeat this procedure—reading from one disk and writing to another—about six times before it makes a complete copy.

Use the copy programs on the SYSTEM MASTER disk to copy an entire disk. COPYA is for Applesoft BASIC users; COPY is for Integer BASIC users.

WS
Applesoft
Basic

There are two programs on the SYSTEM MASTER for copying a disk as a whole. Both copy programs, called COPYA and COPY, function exactly alike. They have different names because they work with different languages. You use COPYA (notice the A at the end) when your computer is running Applesoft BASIC; you use COPY when your computer is running Integer BASIC.

Remember: You can tell which BASIC you have by looking at the prompt. A `]` prompt means your computer is running Applesoft BASIC; a `>` prompt means your computer is running Integer BASIC.

Once you have gone through the copy procedure, you'll find it's easy and quick. However, it can be a little confusing the first time through. So be sure to follow the instructions carefully. In particular, only press **RETURN** when the manual tells you to.

There are three terms you will see on your display during the copy procedure that need some explanation.

- **ORIGINAL** refers to the disk you want to copy, in this case, the SYSTEM MASTER.
- **DUPLICATE** refers to the blank disk that will receive the information.
- **DEFAULT** refers to the slot or drive number that will be used if you don't specify a different one.

Note: If you get an unfamiliar message during copying, see "If You Have Copying Problems."

The copying procedure is divided into two step-by-step sections: one for users with one disk drive and one for users with two disk drives. Read only the section that applies to you.

Copying With One Disk Drive

Two Drive Users: See the section that follows if you have two disk drives.

What You Do...

What You Get...

1. Insert the SYSTEM MASTER disk into drive 1. Check that the drive door is completely closed.
-

2. Look at the cursor and determine which BASIC language you are using.

If you see the `_` prompt, type

`R U N SPACE C O P Y A` ✓

If you see the `>` prompt, type

`R U N SPACE C O P Y`

3. After checking what you typed for errors, press `(RETURN)`.

Pressing `(RETURN)` starts the copy program running. You'll see APPLE DISK DUPLICATION PROGRAM on your display.

4. Press `(RETURN)` *three* times to accept the values the computer assumes you will want. These values are called **defaults**.

You should see this on your display screen:

ORIGINAL SLOT: 6
DRIVE: 1
DUPLICATE SLOT: 6

DRIVE: DEFAULT = 2

A **default** is any value that is automatically used by a computer system if you don't give another value. It is the computer's best guess.

An **expansion slot** is a connector inside the Apple II in which a peripheral card, such as a disk controller card, can be installed.

A **disk controller card** is a peripheral card that connects one or two disk drives to the Apple II and controls their operation. A **peripheral card** is a removable printed-circuit board that plugs into an expansion slot and expands or modifies the computer's capabilities.

The computer has several **expansion slots**. The copy program assumes your **disk controller card** is in slot 6 for both the original and duplicate disk drives and that you'll be using drive 1 for the original disk. By pressing `(RETURN)` three times, you accept these default values.

5. Press **1**. This tells the copy program that the blank disk, the soon-to-be duplicate of the original, also will be in drive 1.

The default is set to 2 in the copy program because it's possible to have two disk drives connected via the same slot in the computer. If you do have two drives, you should be reading the next section, "Copying With Two Disk Drives."

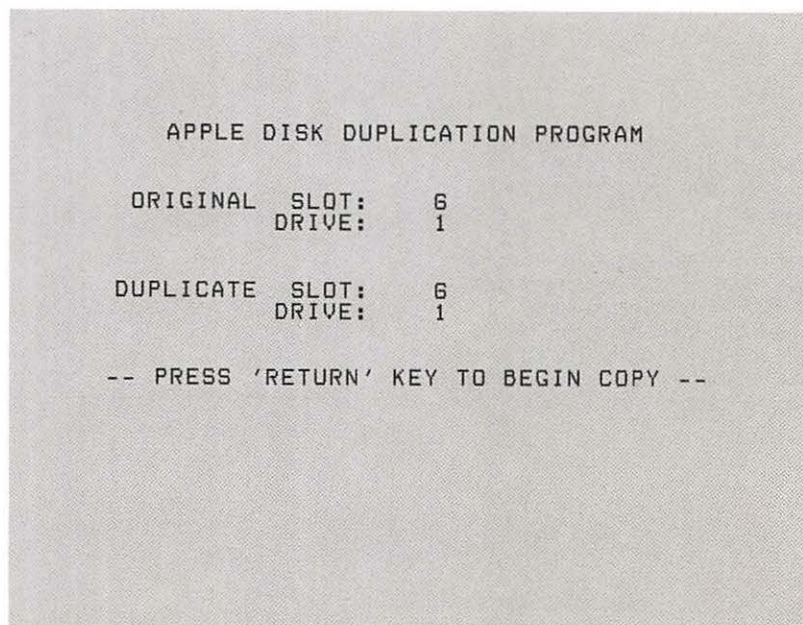
Notice that, on your display, DRIVE: DEFAULT = 2 has changed to DRIVE: 1.

The copy program is now displaying all the slot and drive numbers that it will use.

When the message PRESS 'RETURN' KEY TO BEGIN COPY appears on the display, it means that the program knows where to find the information and where to put a copy of the information. It has not transferred anything from the disk yet.

Your display should look like Figure 1-11. If your display does not match Figure 1-11, and if you are using only one drive, you won't be able to copy correctly. Press **CONTROL-RESET**. When you see BASIC's prompt, go back to step 2.

Figure 1-11. The Slot and Drive Numbers for Copying With One Disk Drive



6. Press **RETURN** to begin the copy process. If you get an unfamiliar message during copying, see "If You Have Copying Problems."

Pressing **RETURN** tells the program to begin.

You will see **INSERT ORIGINAL DISK AND PRESS RETURN**. This message indicates that the program is now ready to read information from the disk you intend to copy. If you wanted to copy a disk other than the **SYSTEM MASTER**, you would put it into drive 1 at this point.

7. Double-check that the **SYSTEM MASTER** is in drive 1. Then press **RETURN** to tell the computer you have inserted the original disk.

Drive 1's light goes on, and you'll see **READING** displayed in inverse next to **ORIGINAL SLOT: 6**.

This means the copy program is reading information from the **SYSTEM MASTER** into the computer's main memory, which temporarily stores as much information as it can.

Then you'll see **INSERT DUPLICATE DISK AND PRESS RETURN** on the display. This message indicates that the program is ready to transfer a copy of the information it has temporarily stored in main memory to the blank disk.

8. Carefully remove the SYSTEM MASTER from drive 1, and put the blank disk into drive 1. Then press **RETURN**.

You'll see **INITIALIZING** displayed in inverse next to **DUPLICATE SLOT: 6**. This means the computer is preparing the blank disk to receive information. More on this later.

Then you'll see **WRITING** displayed in inverse next to **DUPLICATE SLOT: 6**. This means the program is putting a copy of the information in main memory on the formerly blank disk.

Finally, you'll see **INSERT ORIGINAL DISK AND PRESS RETURN**.

9. Exchange the duplicate disk in drive 1 for the original (the SYSTEM MASTER).

The SYSTEM MASTER should be in drive 1.

Press **RETURN**.

You'll see **READING** displayed in inverse next to **ORIGINAL SLOT: 6** and drive 1's light will go on.

This means the program is reading and temporarily storing more information from the SYSTEM MASTER.

10. Wait for the message
INSERT DUPLICATE
DISK AND PRESS
RETURN. Then remove the
SYSTEM MASTER from drive 1
and insert the duplicate (formerly
blank) disk into drive 1.

Press **RETURN**.

You'll see drive 1's light go on
and **WRITING** displayed in
inverse next to **DUPLICATE
SLOT: 6**.

This means the program is
putting a copy of the information
on the duplicate disk.

11. Repeat steps 9 and 10 until
you see the message **DO
YOU WISH TO MAKE
ANOTHER COPY?**

This prompt indicates that the
program has finished copying
the original disk. You now have
two copies of the **SYSTEM
MASTER** disk. Exciting, isn't it?

For guidelines on **labeling**, see "Caring
for Disks" earlier in this chapter.

12. Label your new copy
appropriately.

Be Persistent: The process with one disk drive takes about six swaps to
make a copy of the **SYSTEM MASTER**. How many times you have to
swap the original disk for the duplicate disk depends on how much data is
on the original.

13. If you would like to make another copy, type **Y** for “yes.” Then follow the instructions on the display.

You do not have to enter information about the slot and drive numbers again. The process picks up with step 6.

If you are going to make a copy of the **SAMPLE PROGRAMS** disk, cover its write-enable notch with a silver write-protect tab. Then insert the **SAMPLE PROGRAMS** disk every time the computer prompts you to **INSERT ORIGINAL DISK AND PRESS RETURN**.

If you do not want to make another copy type **N** for “no” and press **RETURN**. Then go on to the section “Testing a Copied Disk.”

You’ll see BASIC’s prompt on your display.

Note: It is a good idea, whenever you are copying a disk that has a write-enable notch, to cover the notch before copying. That way the original will not be damaged if something goes wrong. Remember, when the notch is covered, as with a write-protect tab, information can only be read from the disk, not written onto it.

Copying With Two Disk Drives

What You Do...

What You Get...

1. Insert the **SYSTEM MASTER** disk into drive 1. Check that the drive door is completely closed.

2. Look at the cursor and determine which BASIC language you are using.

If you see the `]` prompt, type

`R U N SPACE C O P Y A`

If you see the `>` prompt, type

`R U N SPACE C O P Y`

3. After checking what you have typed for errors, press `RETURN`.

Pressing `RETURN` starts the copy program running. You'll see `APPLE DISK DUPLICATION PROGRAM` on your display.

4. Press `RETURN` four times to accept the values the computer assumes you will want. These values are called **defaults**.

The computer has several **expansion slots**. The copy program assumes your **disk controller card** is in slot 6 for both the original and duplicate disk drives and that you'll be using drive 1 for the original disk and drive 2 for the duplicate disk. By pressing `RETURN` four times, you accept these default values.

The copy program is now displaying all the slot and drive numbers that it will use. When the message `PRESS 'RETURN' KEY TO BEGIN COPY` appears on the display, it means that the program knows where to find the information and where to put a copy of the information. It has not transferred anything from the disk yet.

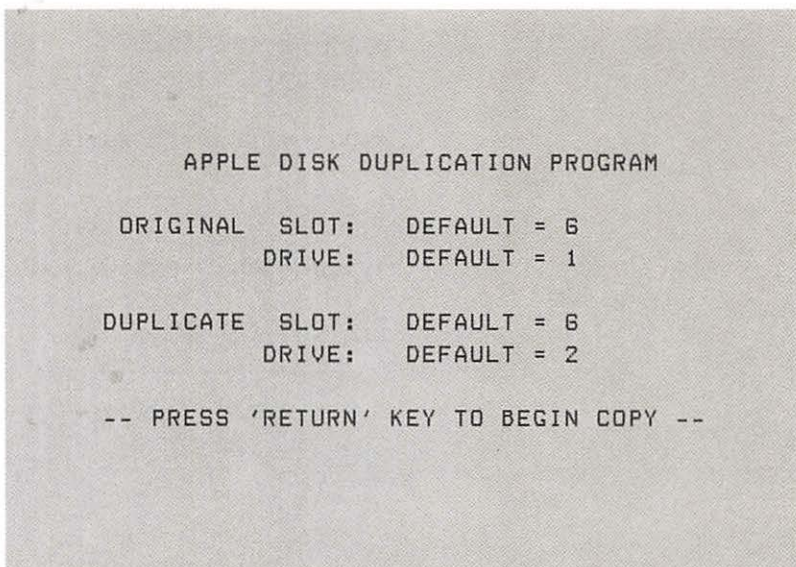
Your display should look like Figure 1-12. If your display does not match Figure 1-12, and you are using two drives, you won't be able to copy correctly. Press `CONTROL-RESET`. When you see BASIC's prompt, go back to step 2.

A **default** is any value that is automatically used by a computer system if you don't give another value. It is the computer's best guess.

An **expansion slot** is a connector inside the Apple II in which a peripheral card, such as a disk controller card, can be installed.

A **disk controller card** is a peripheral card that connects one or two disk drives to the Apple II and controls their operation. A **peripheral card** is a removable printed-circuit board that plugs into an expansion slot and expands or modifies the computer's capabilities.

Figure 1-12. The Slot and Drive Numbers for Copying With Two Disk Drives



5. Be sure to insert your original disk in drive 1 and a blank disk in drive 2 before you press **RETURN** to begin copying.

You'll see **READING** as the program reads and temporarily stores information from the **SYSTEM MASTER** in main memory. Then you'll see **INITIALIZING** as the computer prepares the blank disk to receive the information. Then you'll see **WRITING** as the program puts a copy of the information on the duplicate disk. And then you'll see **READING** again as the program reads and temporarily stores more information from the **SYSTEM MASTER**. The program alternately reads and writes until all the information is copied onto the new disk.

If you get an unfamiliar message during copying, see "If You Have Copying Problems."

When copying is complete, you'll see **DO YOU WISH TO MAKE ANOTHER COPY?**

For guidelines on **labeling**, see “Caring for Disks” earlier in this chapter.

6. Label your new copy appropriately.

7. If you would like to make another copy, type **Y** for “yes.” Be sure to put another blank disk in drive 2, and then follow the instructions on the display.

You do not have to enter information about the slot and drive numbers again. The process picks up with step 5.

If you are going to make a copy of the **SAMPLE PROGRAMS** disk, cover its write-enable notch with a silver write-protect tab. Then insert the **SAMPLE PROGRAMS** disk in drive 1.

If you do not want to make another copy, type **N** for “no” and press **RETURN**. Then go on to the section “Testing a Copied Disk.”

You’ll see BASIC’s prompt on your display.

Note: It is a good idea, whenever you are copying a disk that has a write-enable notch, to cover the notch before copying. That way the original will not be damaged if something goes wrong. Remember, when the notch is covered, as with a write-protect tab, information can only be read from the disk, not written onto it.

If You Have Copying Problems

You may get an ******* UNABLE TO READ ******* or **I/O ERROR** message if the **SYSTEM MASTER** is not in drive 1, if the disk isn’t completely inserted, if the door to drive 1 isn’t completely closed, or if you indicated the wrong slot or drive number.

You may get an ******* UNABLE TO WRITE ******* or **I/O ERROR** message if your duplicate disk isn’t properly seated in the disk drive, if the duplicate disk is write-protected (see the section “About Disks and Disk Drives”), if the drive door isn’t completely closed, or if you indicated the wrong slot or drive number.

If **COPYA** cannot read from the source disk or write to the duplicate disk, you’ll be asked **DO YOU WISH TO MAKE ANOTHER COPY?** Type **N** for “no” and press **RETURN**. You’ll see the Applesoft BASIC prompt.

If COPY cannot initialize the duplicate disk, you'll see `I / O ERROR` and then the Integer BASIC prompt. If COPY cannot transfer the information, you'll be asked `DO YOU WISH TO MAKE ANOTHER COPY?` Type `N` for "no" and press `RETURN`. You'll see the Integer BASIC prompt.

Now, before you try to copy again, check to see that

- your SYSTEM MASTER disk is properly inserted in drive 1
- the drive door(s) is completely closed
- the blank disk you are using for copying has a write-enable notch (and it is not covered with a write-protect tab)
- you are using the correct slot and drive numbers
- you are following the instructions in "Copying With One Disk Drive" if you have one drive or in "Copying With Two Disk Drives" if you have two drives.

Chapter 3 has more on slot and drive numbers.

After you have checked these things, go back to step 2 of the appropriate section and try again.

If you keep having problems, it's possible that the blank disk is faulty. As a last resort, try another blank disk.

If you still cannot copy a disk, contact your Apple dealer.

Testing a Copied Disk

After you have copied a disk, it is a good idea to check the copy.

See "Looking at the Contents of a Disk: the CATALOG Command" for more information.

If the duplicate disk is not a system disk, you can do this by checking the catalog. If you get a listing and it is the same as that of the original, you have a good copy. If you get an error message, or if the catalog is incomplete, try the copy procedure again.

Remember: You type

`CATALOG`

and press `RETURN` to get a listing of a disk.

You should always test your copies.

If the new disk is a system disk, you should check the disk by restarting the computer with the copy. For instance, to make sure that your copy of the SYSTEM MASTER is good, try to start DOS with the new disk.

Two restart methods are described in this section. The method you use depends on the kind of Apple II computer you have. Other methods are discussed in Chapter 5.

- If you have an Apple IIe, both procedures will work.
- If you have an Apple II or Apple II Plus, only the second procedure will work.


You can restart the Apple IIe without turning the computer off and then on again.


You can restart an Apple IIe computer without turning the computer off and then on again. This method saves time and extends the life of the computer's mechanical and electronic parts.

What You Do...

What You Get...


1. Place the new copy of the SYSTEM MASTER in drive 1 and close the drive door.

2. Press  and **CONTROL**. Hold them down while you press **RESET**.

Let go of **RESET**, then **CONTROL**, and then release .

You'll see drive 1's light go on and hear the disk drive start up. After a bit, the whirring will stop and the light will go out.

If the SYSTEM MASTER has been copied accurately, you'll see DOS's opening display (Figure 1-8.) If you don't, your disk may not be a good copy of the SYSTEM MASTER. Go back to the appropriate section in "Copying the SYSTEM MASTER Disk" and make a fresh copy.

The previous method of restarting the computer only works with the Apple IIe, which has .

If you have questions about the Apple IIe keyboard, see the owner's manual.

You can use the next procedure to restart all three kinds of Apple II computers (Apple II, Apple II Plus, and Apple IIe). Remember, successfully starting the program on the new disk proves that the disk is good.

What You Do...

What You Get...

1. Reach behind the left rear corner of the computer. Find the power switch with your left hand and turn it off. (If you need help finding the power switch, see Figure 1-8.)

2. Place the new copy of the SYSTEM MASTER in drive 1 and close the drive door.

3. Now, turn the computer on.

You'll see drive 1's light go on and hear the drive start up. After a bit, the whirring will stop and the light will go out.

If the SYSTEM MASTER has been copied accurately, you'll see DOS's opening display (Figure 1-8.) If you don't, your disk may not be a good copy of the SYSTEM MASTER. Go back to the appropriate section in "Copying the SYSTEM MASTER Disk" and make a fresh copy.

Remember that if you see an asterisk (*), you'll have to type **6 CONTROL-P** to start DOS up after you turn the computer on.

If you have an asterisk on your display, see "No Messages or Mysterious Messages" in "Solving Startup Problems" earlier in this chapter.

See Chapter 5, "The PR# Command" and "Other Ways to Start DOS," for more ways to restart the computer.

Preparing a Blank Disk: the INIT Command

Now that you've learned how to start DOS and how to copy everything on one disk to a new disk, it is time to prepare a blank disk so it can receive information.

When you buy blank disks, they have nothing recorded on them—just like blank tape for a tape recorder. Before a blank disk can receive information from DOS, you must prepare it using the INIT command. This preparation process is called **initializing** or **formatting**.

Initializing, also called **formatting**, means to prepare a disk to receive information.

The **INIT command** prepares the disk and puts DOS and the greeting program on the disk.

You use two distinct operations to prepare any blank disk. The first is to write a greeting program; the second is to issue the INIT command.

The greeting program is called by that name since it greets you: each time you start the disk, the program will be run automatically. The greeting program is commonly named HELLO. It helps keep life simple to use a standard name for greeting programs as you initialize disks.

This section explains how to create a greeting program for initializing your blank disks. The content of such a program can (and does) vary. Sometimes hello programs contain a single line. Sometimes—as on the DOS 3.3 SYSTEM MASTER disk—they are quite complex.

Warning

When you initialize a disk, everything on the disk is erased.

Follow these steps to initialize a disk:

What You Do...

What You Get...

1. Remove the SYSTEM MASTER disk from drive 1; put a blank disk into drive 1 and close the door. Be sure the write-enable notch is not covered.

The **NEW** command clears main memory for a new program.

2. Type

NEW

and press **RETURN** to erase anything that might be in main memory.

3. Now type this short program. Press **RETURN** after you have typed each line correctly. Substitute your name and the actual date when you type line 20.

Type the spaces and quotation marks carefully. Remember, you can use the **←** key to back the cursor up and correct any mistakes before you press

RETURN.

Your display should look like this:

PRINT is a BASIC command. It is used to print. Whatever you type in quotation marks after **PRINT** will be displayed when the program is run.

```
10 PRINT "THIS HELLO PROGRAM CREATED BY"  
20 PRINT "(your name) ON (date)"  
30 END
```

The **END** command ends a program.

This program will be used by the **INIT** command to prepare the blank disk. Then, each time this disk is started up, the program will be run.

4. Test your program by typing

RUN

and pressing **RETURN**.

You'll see the lines you just typed appear on the display.

5. Now type

`INIT SPACE HELLO`

and press `RETURN`.

HELLO is the name of the three-line program you just wrote. INIT is a DOS command that, when used in combination with a program name, tells the disk operating system to initialize the blank disk. INIT puts HELLO on the disk and makes it the greeting program that DOS will run each time that disk is started up.

When the disk drive quiets down, the light goes off, and the prompt appears, initializing is complete: DOS initialized the disk.

If you get an `I/O ERROR` message, check the disk and the disk drive; then type the INIT command again.

6. To test the disk to make sure it is, in fact, initialized, type

`CATALOG`

and press `RETURN`.

You should see a short catalog like this:

`DISK VOLUME 254
A 002 HELLO`

DOS assigned your disk the volume number 254 (which it always does unless you specify a different number) and wrote your BASIC program on it. The program, as you can see from the catalog, is two (002) **sectors** long, and is named HELLO.

7. Carefully remove the disk from the drive and attach a label. See Figure 1-1 for the location of the label. You may want to note on the label that the disk is initialized. Keep the disk in a **disk envelope** to protect it from dust when you are not using it.

See Chapter 3, "The INIT Command," for how to specify a volume number.

A **sector** is a fixed portion of the disk. Initializing divides the disk in tracks and sectors. See the *DOS Programmer's Manual* for more on sectors.

A **disk envelope** is a protective paper sleeve.

The disk you've just prepared and tested is ready to use with DOS. You can use it to hold programs, data, text, graphics, or whatever information you'd like. Leave the write-enable notch of this disk uncovered so that you can write to and save information on the disk.

Any disk that has been initialized is memory-size dependent: the size of the system that initializes the disk determines the size of the system that can use the disk. If the disk is initialized on a 32K system, then the disk can only be used on a system with 32K or more of memory.

Transferring a copy of the information on a disk into memory is known as **loading**.

Changing or modifying information in memory is known as **editing**.

Transferring a copy of the information in memory to a disk is known as **saving**.

Using Programs

Some programs, like games, run automatically when you start the disk drive. They take care of just about everything for you—displaying instructions, following your orders, bringing in information from the disk, and saving information on the disk.

However, some programs require you to know how to do some of this work. In particular, you need to know how to bring information from a disk into the computer's memory (called **loading**); how to change information while it is in memory (called **editing**); and how to save information from the computer's memory onto a disk (called **saving**).

You also need to know how to instruct the computer to perform these functions if you plan to do any programming. This part of the tutorial explains loading, editing, and saving information.

Reading Information From a Disk: the **LOAD** Command

To look at the contents of a file that is stored on a disk, you must first bring a copy of the information in the file from the disk into the main memory of the computer. This procedure is called **loading** and is accomplished with the **LOAD** command.

LOAD is a DOS command that finds the named BASIC program on the disk and puts a copy of that program into the computer's main memory.

The **LIST** command shows a listing of the program in memory on the display.

What You Do...

1. Put the **SAMPLE PROGRAMS** disk in drive 1.

2. Type

LOAD **SPACE** **VERIFY**
.ME

and press **RETURN**.

What You Get...

The disk drive will work away as DOS finds **VERIFY.ME** and puts a copy of it in main memory.

When the program is in memory, you'll see BASIC's prompt again.

If you see **FILE NOT FOUND**, type the command again. Note that the dot is part of the filename; be sure and type it.

3. To look at this program, type

LIST

and press **RETURN**.

You'll see the contents of **VERIFY.ME**, which is only one line:

10 REM VERIFY.ME

Memory Isn't Permanent: Information brought into memory from a disk is erased from main memory when you turn off the computer. However, only a copy is put in memory, so you still have the information on the disk whenever you need it.

Changing the Contents of Memory

To change the information saved in a file you must first load a copy of the contents into the computer's memory. Since you have already loaded the VERIFY.ME program into memory, it is handy to use for learning how to make changes to something in memory and how to save the new version.

Files and Memory: A *file* is a collection of information stored as a named unit on a peripheral storage device, such as a disk. In other words, the term *file* is only used to refer to something on a disk. When that same information is put into memory, it is no longer called a file, but a document, a program, a picture, a worksheet, data, or whatever.

What You Do...

What You Get...

1. Remove the SAMPLE PROGRAMS disk from the disk drive and insert the disk you just initialized into drive 1.

2. To make sure you have inserted the correct disk, type

C A T A L O G

and press RETURN.

You'll see that the HELLO program is the only program on your disk.

3. Type

L I S T

and press RETURN.

You'll notice that a copy of VERIFY.ME is still in memory, unless you have turned off your computer, in which case you should load the program again.

As you see, the contents of VERIFY.ME consist of one line.

10 REM VERIFY.ME

4. Now add another line to the program so your display looks like this:

```
20 PRINT "NEW LINE"
```

and press **RETURN**.

Pressing **RETURN** adds line 20 to the VERIFY.ME program that is stored in the computer's memory. It does not, however, change the copy of the program that is on the disk.

5. Add one more line:

```
30 END
```

and press **RETURN**.

This line also is added to the program in memory.

6. To check the new program, issue the LIST command and press **RETURN**.

The program now in memory is displayed:

```
10 REM VERIFY.ME
20 PRINT "NEW LINE"
30 END
```

You have now changed the contents of the program in memory by adding two lines to it. So long as you leave your computer on, and do not issue a NEW command or replace those lines with some others, the program VERIFY.ME will remain in the computer's memory. To keep the program so you can use it over and over, you must put a copy of it on a disk. The next section shows you how to save your program.

Saving Information On a Disk: the SAVE Command

Putting a program into a file on a disk stores it permanently—or at least until you change its contents. The SAVE command allows you to save any changes that you've made to a program (for example, a new date in a greeting program); information that you've added to a program (for example, a new entry in a phone list); and whole new programs that you've written.

SAVE puts a copy of the BASIC program in memory onto a disk.

Whenever you **save** information, you must use a disk that you can write on, that is, one that is not write-protected. Use the disk you just initialized for the following procedure.

What You do...

1. To put a copy of the program in memory on your disk, type

S **A** **V** **E** **S** **P** **A** **C** **E** **S** **P** **A** **C** **E** **P** **R** **O** **O** **F**
S **P** **A** **C** **E** **P** **O** **S** **I** **T** **I** **V** **E**

and press **R** **E** **T** **U** **R** **N**.

What You Get...

If you see the message **SYNTAX ERROR**, try again, making sure that you are using uppercase and that you haven't made any typos.

The disk whirs as DOS copies the information in memory, a program, to a disk file named **PROOF POSITIVE**.

The revised program (lines 10-30) is still in memory, and now it is also on disk.

Note: You have to type a space character between **PROOF** and **POSITIVE** if you want the filename to be two words. Remember that to the computer a space is a character just like any letter or number. If you want a space character, you must press the **S** **P** **A** **C** **E** bar.

-
2. To check that the program was saved on the disk, issue the **CATALOG** command and press **R** **E** **T** **U** **R** **N**.

Your display should look like this:

D **I** **S** **K** **V** **O** **L** **U** **M** **E** **2** **5** **4**

A **0** **0** **2** **H** **E** **L** **L** **O**

A **0** **0** **2** **P** **R** **O** **O** **F** **P** **O** **S** **I** **T** **I** **V** **E**

Seeing **PROOF POSITIVE** is indeed proof that you have successfully saved the program in memory to a file on the disk.

3. Type

NEW

and press **RETURN**.

Main memory is cleared. If you issue a LIST command, all you will get is BASIC's prompt. You can now start another program or load something from a disk.

4. To double-check your new program, type

LOAD SPACE PROOF
SPACE POSITIVE

and press **RETURN**.

DOS brings a copy of the program you just saved from the disk into memory.

5. Type

LIST

and press **RETURN**.

The listing of PROOF POSITIVE is displayed.

By the Way: The original one-line program VERIFY.ME is still on the SAMPLE PROGRAMS disk. You saved the three-line version with a different name, PROOF POSITIVE, so now you have both versions safely stored on disk.

Whenever you change a program, if you want to keep both versions, you must save the new version under a different filename. If you use the same filename for both versions, the new version will be written over the old version, and the old version will be lost.

The **RUN** command finds a BASIC program on the disk, puts a copy into memory, and starts the program running.

A **menu** is a list of choices.

Running a Disk Program: the RUN Command

Many programs do not run automatically when you start the disk. To execute such programs, you use the DOS command RUN.

What You Do...

What You Get...

1. Remove your initialized disk from the disk drive. Insert the SAMPLE PROGRAMS disk into drive 1.

2. Type

R U N SPACE C O L O R SPACE
T E S T

DOS finds, loads, and starts the program called COLOR TEST. You'll see COLOR TEST's **menu** of demonstrations.

and press RETURN.

3. COLOR TEST lets you adjust the colors on your color screen. Even if you don't have a color screen, you'll see a two-tone display. The program gives you instructions on your screen. To operate the demonstration, type

1, 2, 3, or 4

and press RETURN.

The demonstration begins.

4. To end demonstration 1, 2, 3, or 4 and return to the menu, press RETURN.

You'll see the menu again.

5. To quit using the COLOR TEST program, type

You'll see BASIC's prompt on your display again.

5.

The RUN command is the command that you'll use again and again to start programs running. If you want to try it out again, there are several interesting programs on the SAMPLE PROGRAMS disk.

Some older models of the Apple II may not be able to run COLOR TEST, and you will see the message LANGUAGE NOT AVAILABLE. If so, try running the APPLEVISION program. Then follow that program's instructions.

Turning Off the Computer

You are finished with the tutorial. All you need to do is turn off the computer. Follow these steps:

1. Remove the disk and close the door of your disk drive. Carefully put all your disks away.
2. Reach behind the left rear corner of the computer. Find the power switch with your left hand. Turn it off.
3. Turn off your video monitor or TV set.

This ends the tutorial. If you feel you need more practice with the commands and programs in this chapter, go ahead and play around with them. If you're feeling adventurous, try to run a few demonstration programs on the SAMPLE PROGRAMS disk. Check Appendix C for an explanation of the programs on the DOS disks. Feel free to try those that look interesting to you.

When you're ready to move on, Chapter 2 begins the reference section of this manual with an overall picture of DOS—what happens behind the scenes when you start DOS and how DOS relates to the other programs that come with your computer.

Chapter 1 Summary

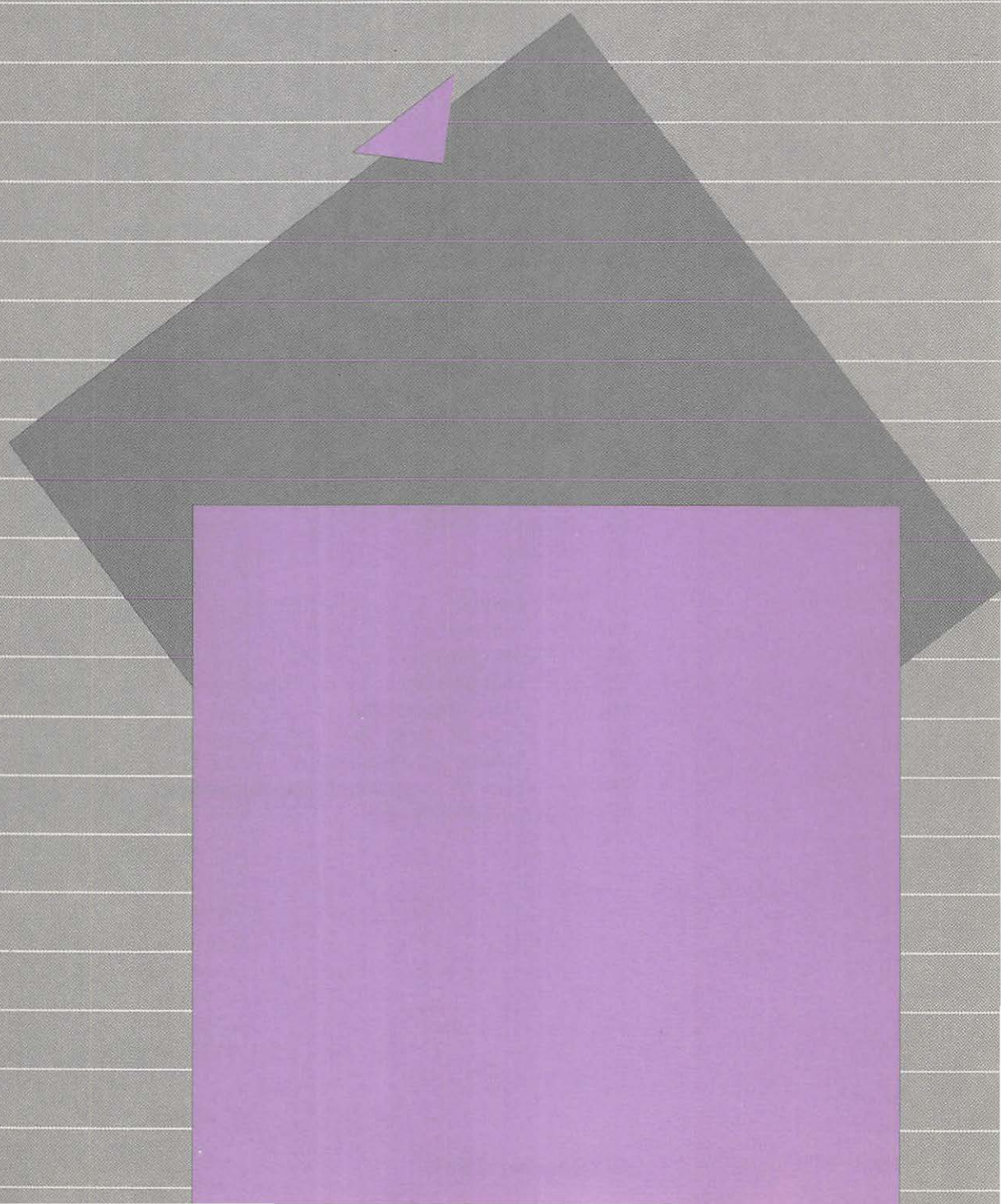
This tutorial has taken you through the basics of using DOS. In addition to all the new terminology, you've learned how to

- take care of disks
- start up your computer with DOS
- insert and remove a disk from a disk drive
- look at the contents of a disk by using the CATALOG command
- copy a disk with the copy programs on the SYSTEM MASTER disk
- test a disk copy by checking the disk's catalog or by seeing if the disk can restart the computer
- create a greeting program for a blank disk
- prepare a blank disk to receive information by using the INIT command
- transfer a copy of the contents of a file from a disk by using the LOAD command
- save the information in memory to a disk by using the SAVE command
- start a program that's stored on a disk by using the RUN command
- turn off the computer

That's quite a lot; give yourself a pat on the back.

An Overview of DOS

45	How DOS Works
46	The Monitor Program
46	The Startup Process
49	The Startup Drive
49	Startup Disks
49	Two BASIC Languages
50	Files and Volumes
50	Files
52	Filenames
52	Volumes
53	DOS Commands
53	Syntax
54	Options
54	Defaults
55	Numbers
55	The Arguments
56	Filename: fn
56	Slot Number: [,Sn]
57	Drive Number: [,Dn]
57	Volume Number: [,Vn]
58	Error Messages From Bad Arguments
59	Chapter 2 Summary



An Overview of DOS

DOS is a program, or set of instructions, that keeps track of your disk files, handles the flow of information between disks and memory, and takes care of many other housekeeping tasks. The DOS program is on the SYSTEM MASTER disk.

This chapter begins the reference part of this manual. It describes how DOS works and the form, or syntax, used by DOS commands. It also explains how files and volumes relate to each other and to DOS.

It's important to understand how DOS fits in with other programs in your Apple II computer. What happens when you start DOS? How do you identify the information you want to work with? How do you know that DOS is ready to obey your commands? You will find most of the answers in this chapter.

If you have worked through the tutorial (Chapter 1), you may have some questions about how DOS works. This chapter will try to answer those questions.

How DOS Works

When you start DOS, you add the DOS commands and capabilities to the computing powers already available with your computer. Once you have brought the DOS program into your computer's memory, you can run programs that are stored on disks, and you can save information and programs on a disk.

You must bring DOS into memory each time you start the system and want to use disks. To use DOS, you give it instructions in a form that DOS accepts.

The **Monitor program** is a system program built into the Apple II. It supervises main memory.

A **peripheral device** is used in conjunction with a computer. It is often separate from the computer, and it is typically connected by means of a peripheral card.

Main memory is the part of the Apple II computer system that you will use to store information.

If you want to know more about the Monitor program, see the reference manual for your computer.

The disk containing DOS must be in drive 1 to start up the operating system program.

The Monitor Program

The **Monitor program** is a built-in computing capability. The program is always present and acts as a supervisor, controlling all other programs. One of its tasks is starting the resident BASIC. Another task is reading DOS into memory. The Monitor program resides in its own part of memory, separate from the areas in which BASIC and DOS reside.

The Monitor program also controls communication between the keyboard and the display, and between your Apple computer and **peripheral devices**, such as disk drives and printers.

The Monitor program reads DOS into memory and then lets DOS control communication.

The Startup Process

When you turn on your Apple computer, several electrical and mechanical events take place to bring DOS into **main memory**.

As soon as the computer recognizes that it has power, it checks to see if a disk drive is connected. If a drive is connected, the computer tries to bring the information on the disk in drive 1 into memory. These actions are performed by the Monitor program.

If the disk in drive 1 contains DOS, DOS is transferred into memory and started by the Monitor program. When the disk in drive 1 is the DOS 3.3 SYSTEM MASTER, you get the opening display announcing DOS (Figure 2-1), and DOS tries to bring into memory the alternate BASIC. DOS then gets BASIC running.

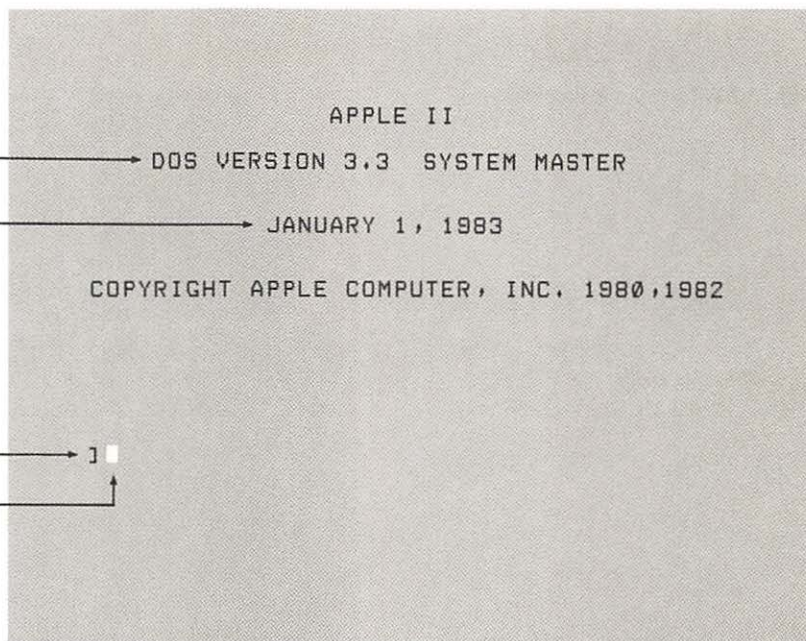
Figure 2-1. The Opening Display

The DOS version number, and the name of the startup disk.

The actual date of the system program; yours may be different.

The Applesoft BASIC prompt.

The cursor.





The **prompt character**, or **prompt**, is used by the computer to let you know that it is ready for your command. It is also used to identify the program or component of the system that is doing the prompting.

BASIC prints its **prompt character**, often called the **prompt**, on the display to let you know the computer is ready for your command.

If you have an Apple II, you may have an older version of the Monitor program that does not include autostart. When you start DOS, all you'll see on your display is an asterisk (*).

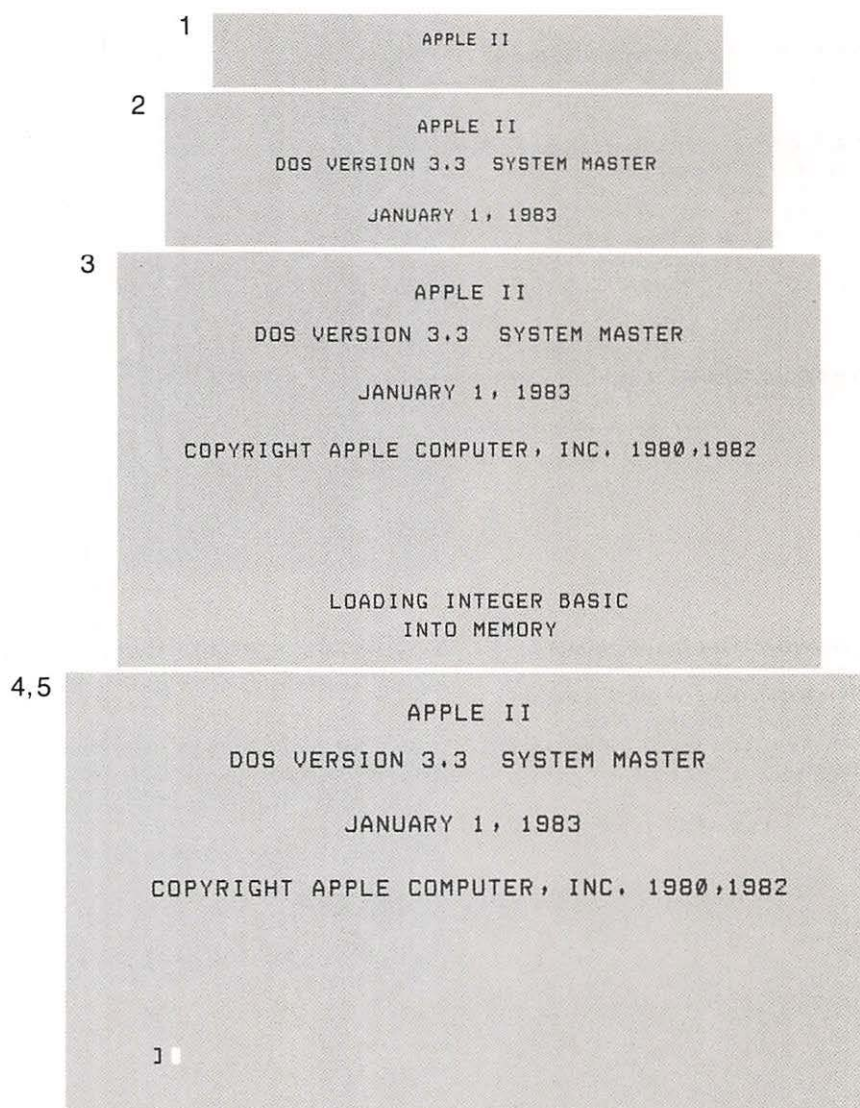
To begin the startup process described here, you'll have to type

 CONTROL 

You will have to do this every time you want to start up DOS.

Figure 2-2 illustrates the steps in the startup process.

Figure 2-2. The Startup Process



1. Monitor program brings DOS into memory and starts it.
2. DOS notes the size of memory and checks to see if the alternate BASIC is available.
3. DOS runs HELLO, its startup program.
4. If your computer's memory is large enough, DOS puts the alternate BASIC in memory.
5. DOS starts BASIC. BASIC prints its prompt.

An **expansion slot** is a connector inside the Apple II in which a peripheral card, such as a disk controller card, can be installed.

A **disk controller card** is a peripheral card that connects one or two disk drives to the Apple II and controls their operation. A **peripheral card** is a removable printed-circuit board that plugs into an expansion slot and expands or modifies the computer's capabilities.

The **startup drive** is the disk drive the computer checks first for a system disk.

A **startup disk** has the information necessary to set the system in operation. It is sometimes called a **boot disk**.

The INIT command is used to **initialize**, or **format**, a disk so that it can receive information. See Chapter 3 for more on the INIT command.

Hardware refers to those components of a computer system consisting of physical (electronic or mechanical) devices.

Firmware refers to those components of a computer system consisting of programs stored permanently in read-only memory.

The Startup Drive

The autostart ROM in the Apple II computer looks for and tries to read from the highest-numbered **expansion slot**, beginning with slot 7.

When slot 7 is empty, the autostart ROM tries to read from slot 6. Your **disk controller card** should be in slot 6.

In addition, your first disk drive's cable should be attached to the drive 1 connector of the controller card, the second disk drive's cable should be attached to the drive 2 connector.

That way, turning on your computer causes the information on the disk in drive 1 to be put into memory. And, accordingly, drive 1 is known as the **startup drive**.

If you do not know which disk drive is drive 1, and it is not labeled, you can determine the startup drive by turning on the computer and watching to see which disk drive's light comes on first. The first drive the computer looks in is the startup drive, or drive 1.

Startup Disks

A **startup disk** is set up so that it automatically runs one of its programs when read. The DOS command INIT designates a startup disk. A startup disk contains all the information that the computer needs to bring the program from the disk into memory and start the program running. The SYSTEM MASTER and the SAMPLE PROGRAMS disks are both startup disks. They automatically start the DOS program.

Starting versus Booting: This manual uses the term *startup* instead of *boot*. Other manuals may tell you to boot your system or boot a disk. Don't let the difference in terminology throw you, the two terms mean the same thing.

Boot came from the word *bootstrap*, which describes something that appears to use its own action to start operating.

Two BASIC Languages

What language you can use depends on the size of your Apple computer and what **hardware** and **firmware** it has. Many Apple II computers are equipped to use both BASIC languages: Integer and Applesoft.

Resident means the language is stored permanently in the computer.

Read-only memory is memory whose contents can only be read. It is used for storing firmware. The contents can never be erased or changed, unless you physically remove the integrated circuit.

Random-access memory is memory whose contents you can read from or write to. The information in RAM is erased when the computer is turned off.

An **integrated circuit**, or **chip**, is an electronic component made up of many circuit elements fabricated on a single piece of silicon.

The LANGUAGE NOT AVAILABLE message is explained in Appendix B.

The BASIC that is available when you turn on your computer is called the **resident** language. It starts automatically when the system starts. The Apple II computer has Integer BASIC in **ROM**, which stands for **read-only memory**. The Apple II Plus and the Apple IIe computers have Applesoft BASIC in ROM.

If you have an Apple II or Apple II Plus computer, you can add a language card to increase the size of **random-access memory**, or **RAM**. A language card cannot be used in an Apple IIe.

When you start up your system and when it has enough memory (at least 64K) for both BASIC languages, DOS brings a copy of the nonresident, or alternate, BASIC from the disk and puts it in the additional RAM of an Apple II or in a language card, if you have one.

When you start DOS, it reports which BASIC it is bringing into memory. When Applesoft BASIC is resident, you will see the message `LOADING INTEGER BASIC INTO MEMORY`. When Integer BASIC is resident, you will see `LOADING APPLESOFT BASIC INTO MEMORY`. When your Apple computer does not have enough temporary storage space for the alternate BASIC, you do not see any message.

Files and Volumes

This section describes **files** and **volumes** and their relation to disks. It also covers the general rules for naming files.

Files

DOS organizes the information on each disk, or volume, into units of disk storage known as files. For the sake of analogy, you can think of disk files as manila folders in the drawer of a filing cabinet. Each folder contains a particular kind of information.

For instance, a drawer labeled "Personnel" might have one folder with information on full-time employees organized by name, another folder with comparative salary data by position, and another folder with the formulas for calculating taxes.

If you were going to computerize your file cabinet, the drawer "Personnel" would become the volume PERSONNEL and each folder would become a file and be appropriately, and uniquely, named: for instance, EMPLOYEES, SALARY, and TAXES.

A **file** is a collection of information stored as a named unit on a storage medium such as a disk.

A **volume** is a collection of all the files on a storage medium.

Any kind of information can be stored in a file, including telephone lists, memos, recipes, pictures, programs, and data.

The important difference between folders and files is efficiency.

- Disk files can store large amounts of information in a small space.
- Disk files can be accessed and changed quickly.

File types are listed in Chapter 3.

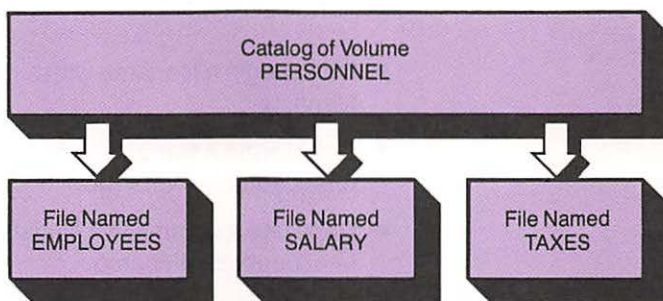
Each file must have a name and a type. When you want to use the information stored in a particular file, you must refer to that file by its name. The file's type indicates the kind of information the file contains. You must know the file's type to use the correct DOS commands.

A **catalog** is like a table of contents for a disk.

On each disk, DOS keeps a list, or **catalog**, of all the files—their names, their types, their size, and their location. Figure 2-3 diagrams a catalog that keeps track of the files in PERSONNEL. Each of these files can be any file type.

A catalog can have entries for 105 files. In actual practice, however, a disk usually runs out of storage space before the catalog reaches its limit.

Figure 2-3. Files in a Catalog



A **filename** is the unique name under which a file is stored.

Filenames

DOS creates and polices files by using **filenames**. A filename can have up to 30 characters. The first character must be a letter; the rest of the name can contain any character you can type except a comma. Each filename on a particular disk must be unique.

Here are a few examples of legal filenames. Note that spaces and periods can be used to separate parts of a filename.

- A SOMNAMBULIST RUNNING RAMPANT is an example of the longest possible name.
- Z is an example of the shortest possible name.
- A.1DERFUL.NAME.3 is an example of how to use letters, numbers, and periods in a filename.

DOS ignores any characters typed after you reach the 30-character limit. For instance, if you type the filename TWO.SOMNAMBULISTS.RUNNING.RAMPANT, which is 33 characters long, DOS shortens it to TWO.SOMNAMBULISTS.RUNNING.RAMP, which is 30 characters long. DOS does this automatically and does not print an error message. But as long as you don't make a typing error, and as long as the shortened name is unique, DOS will still find your file even when you type all 33 characters.

DOS has three don'ts when naming files:

- Don't begin a filename with a number: 2HT2HL is an unacceptable filename.
- Don't begin a filename with a period: .PRINTER is an unacceptable filename.
- Don't use a comma in a filename: BOOP,BETTY is an unacceptable filename.

If you use an unacceptable filename, DOS displays the SYNTAX ERROR message.

Volumes

The collection of all the information placed on a disk is called a **volume**. You can identify a volume with a number and use that number to make sure that DOS is working with the correct disk.

The volume number is assigned when you prepare a disk to receive information with the INIT command. No DOS command *requires* a volume number, although you do have the option of specifying a number when the disk is initialized. Otherwise, DOS assigns the default number, 254.

The **INIT** command is used to **initialize**, or **format**, a disk so that it can receive information. See Chapter 3 for more on the INIT command.

DOS Commands

DOS commands, like the words used in programming languages, must follow certain rules. Some of the rules, like the spelling of a command, always apply. Other rules, like using a drive number with a command, only apply in certain circumstances. This section covers the general rules that apply to DOS commands.

Most DOS commands may be issued not only from the keyboard but also from within a program. This manual covers issuing DOS commands only from the keyboard. The *DOS Programmer's Manual* describes issuing DOS commands from programs.

Syntax

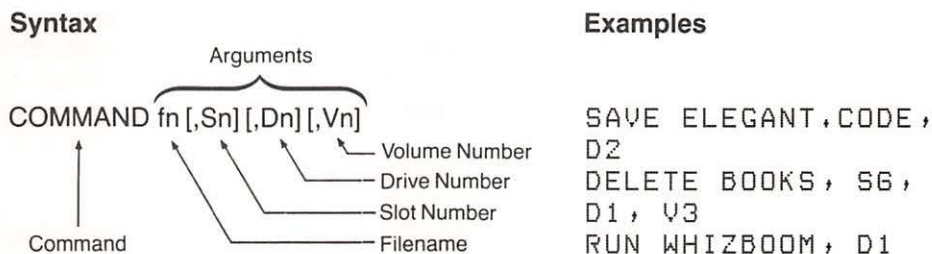
The **syntax** (the order and form) of the various parts of a DOS command is expressed in a kind of shorthand, which is described here:

UPPERCASE	indicates the actual name of something, like a DOS command. Type it exactly as indicated.
lowercase	indicates something you supply, like the name of a program.
fn	indicates a filename that you supply.
[]	indicates an optional argument in a command. You may include the option or not, as you choose. In any case, <i>don't type the brackets</i> .
n	indicates a number you supply.

An **argument** is a value the command uses. It can be required or optional.

All the possible forms of each DOS command are presented in a one-line description of the command, as in Figure 2-4.

Figure 2-4. The Syntax of DOS Commands



In Figure 2-4, **COMMAND** represents any DOS command (for example, **SAVE**, **DELETE**, or **RUN**) and **fn**, **[,Sn]**, **[,Dn]**, and **[,Vn]** are the command's arguments. The brackets indicate that the argument is optional.

Options

An argument within brackets is called an option, something you can omit or include when you give a command. Use a comma to separate an option from the preceding argument. You may put a space before or after the comma. Type the uppercase letter to identify which option you are using. When *n* is used with the argument, you must supply a number.



Warning

The square bracket notation is used in this manual only to indicate that an argument is optional. When you type a DOS command, *do not* type the brackets.

A **default** is a value used by DOS when no other value is specified. (That is, the value is used by default.)

The default disk drive is the one that was last used.

Defaults

When you don't tell DOS anything one way or another about an optional argument, DOS makes an assumption. This assumption, which is called the **default**, is what DOS uses when no other value is explicitly given.

DOS defaults to the disk drive, slot, and volume used during the startup process. Normally, this means that the defaults are

- drive 1
- slot 6
- volume 254

DOS uses these defaults until you indicate a different drive, slot, or volume number. Then DOS defaults to that number.

For example, when you start up DOS, drive 1 is the default (DOS must always be started from drive 1). If you issue a **CATALOG** command, DOS will list the contents of the disk in drive 1. If you want to see the catalog of the disk in drive 2, you must type

```
C A T A L O G , D 2
```

And once you have used drive 2, DOS assumes that you want to keep using that drive. So, drive 2 becomes the default until you specify otherwise.

The default works in the same manner with the slot number. Slot 6 (assuming that's the slot where DOS found your disk controller card) is the default until you tell DOS to use another slot.

If, for example, you type

```
CATALOG,S5
```

DOS would look in slot 5 for a disk controller card. If it found one, DOS would give you a catalog of the disk in drive 2 (the most recently specified). And from then on DOS would use slot 5 as the default value for the slot option. To change the default back to slot 6, drive 1, you would have to use those values in another command, such as `CATALOG,S6,D1`.

Changing Defaults: When you want to change the default value, you must tell DOS explicitly. Include the argument in the next appropriate command.

Numbers

This manual uses a lowercase *n* in syntax notation to indicate that you should use an actual number as part of the argument.

For example, the notation `[,Dn]` means you should substitute the number of the disk drive you want when you type the argument, as in

```
[,D1]
```

or

```
[,D2]
```

An **integer** is a whole number, a number with no fractional part.

A **hexadecimal number** is expressed in terms of powers of sixteen, using the sixteen digits 0 to 9 and A to F.

You can use an **integer** or a **hexadecimal number** for the number. Usually an integer is used. If you want to use hexadecimal numbers, refer to the *DOS Programmer's Manual*.

An **argument** is a value the command uses. It can be required or optional.

The Arguments

The four arguments—`fn`, `[,Sn]`, `[,Dn]`, and `[,Vn]`—are used to specify the filename, slot number, drive number, and volume number, respectively. Together they identify the file you want to use. The arguments in square brackets are optional. When you don't use an optional argument, DOS uses the default, as previously explained. You may use the slot number, drive number, and volume number arguments in any order. For example, `√36,D2,S6` works the same as `S6,D2,√36`.

Filename: *fn*

In syntax notation, *fn* stands for “filename.” Nearly all DOS commands must be followed by a filename. In other words, they *require* this argument. You should supply the name of the file you want as the first argument after the DOS command.

If the default disk drive is the one you want to use, all you need to do is type the command and the filename. For instance, in the commands

```
SAVE PROGRAM
```

and

```
RUN OLD.HAT
```

DOS would assume the default for slot, drive, and volume numbers. If the default drive was drive 2, both commands would use the disk in drive 2. You can see how using the default can be a handy shortcut.

Note: DOS will always assume that the first argument in a command is the filename.

Slot Number: [*,Sn*]

The optional slot number argument is used to specify a particular expansion slot number in DOS commands.

When you use this option, replace the *n* with an actual number from 1 to 7. The value you specify becomes the default slot number. That is, DOS uses this slot number until you specify a different one.

Chapter 5 describes how to use the slot number argument to send information to a printer.

If you don't specify a slot number, DOS looks in the most recently used slot. If you never specify a slot number in a given session with DOS, DOS uses the slot from which you started DOS.

A **controller card** is a peripheral card that connects a device, such as a printer or a disk drive, to the computer and controls the operation of the device.

You would be most likely to use the slot number argument if you have more than one disk controller card or if you have a printer **controller card**.

If you have two disk controller cards, the slot number argument is essential to differentiate between your disk drives. If, for example, you have two controller cards with four disk drives connected, they would be identified as follows:

- drive 1: ,S6 ,D1
- drive 2: ,S6 ,D2
- drive 3: ,S5 ,D1
- drive 4: ,S5 ,D2

This argument is also useful for accessing printers and for transferring information between devices connected to the computer via expansion slots.

When You Don't Have to Specify a Slot Number: If you are using only one or two disk drives and you do not have any other peripheral devices connected to your Apple computer, you don't have to specify a slot number in your DOS command. As long as you don't specify a different number, DOS assumes you will want the slot number you used when you started up your system.

Drive Number: [,Dn]

The optional drive number argument specifies the number of the disk drive that holds the disk you want to use.

When you use this option, replace the *n* with a drive number, 1 or 2. When you omit the drive number argument, DOS uses the drive that was last used. If you never specify a drive number in a given session with DOS, DOS uses the drive number of the startup drive.

Here are some DOS commands that use the slot and drive options:

CATALOG D2	displays the catalog of the disk in drive 2. DOS uses the slot it used last.
RUN WHIZBOOM ,S6 ,D1	reads the program WHIZBOOM from the disk in drive 1 connected to the disk controller card in slot 6 and runs the program.

When You Don't Have to Specify a Drive Number: If you are using one disk drive and do not have any other peripheral devices connected to your Apple computer, you don't have to specify a drive number. As long as you don't specify a different number, DOS assumes you will want the drive number you used when you started up your system.

Volume Number: [,Vn]

The optional volume number argument specifies the number of the volume, or disk, you want to use. Although you can use the volume number argument with most DOS commands, it is rarely used.

If you do use this option, replace *n* with an actual volume number from 1 to 254. Once you specify a number, it becomes the default volume number.

The INIT command is discussed in Chapter 3.

DOS assigns a default volume number of 254 to a disk when it is initialized with the INIT command unless you specify a different volume number.

When you give other DOS commands, if you do not specify a volume number, if you specify volume number 0, or if you type ☐ without a number, DOS will assume that whatever volume number is on the disk is the one you want.

Here is an example of a DOS command that uses the volume option:

LOAD DICE, V21

DOS checks to see if the number of the disk in the default disk drive is 21. If it is 21, DOS reads the program DICE from that volume into memory.

Error Messages From Bad Arguments

If you make an error when using arguments, you might get one or more of these messages:

- The **RANGE ERROR** message indicates that you used a slot number larger than 7 or a drive number other than 1 or 2.
- The **VOLUME MISMATCH** message indicates that you specified a volume number that DOS can't find. This message allows you to retype the command with the correct number before DOS exercises your command on the wrong disk.
- The **I/O ERROR** message can indicate several things:

You may get it because the specified disk drive's door is open. Close the door and try the command again.

You may get it because a connection is loose. Check to see if the disk drive cables are securely connected and that the controller card is properly inserted in the expansion slot.

You may get it if you have specified a slot that doesn't contain a controller card. If a named slot doesn't have a controller card, the system may stop, and you will lose whatever you have in memory. If this is the case, you must restart DOS.

See Chapter 5 for ways to restart DOS.

Appendix B lists all DOS error messages and suggests ways to fix what caused the message.

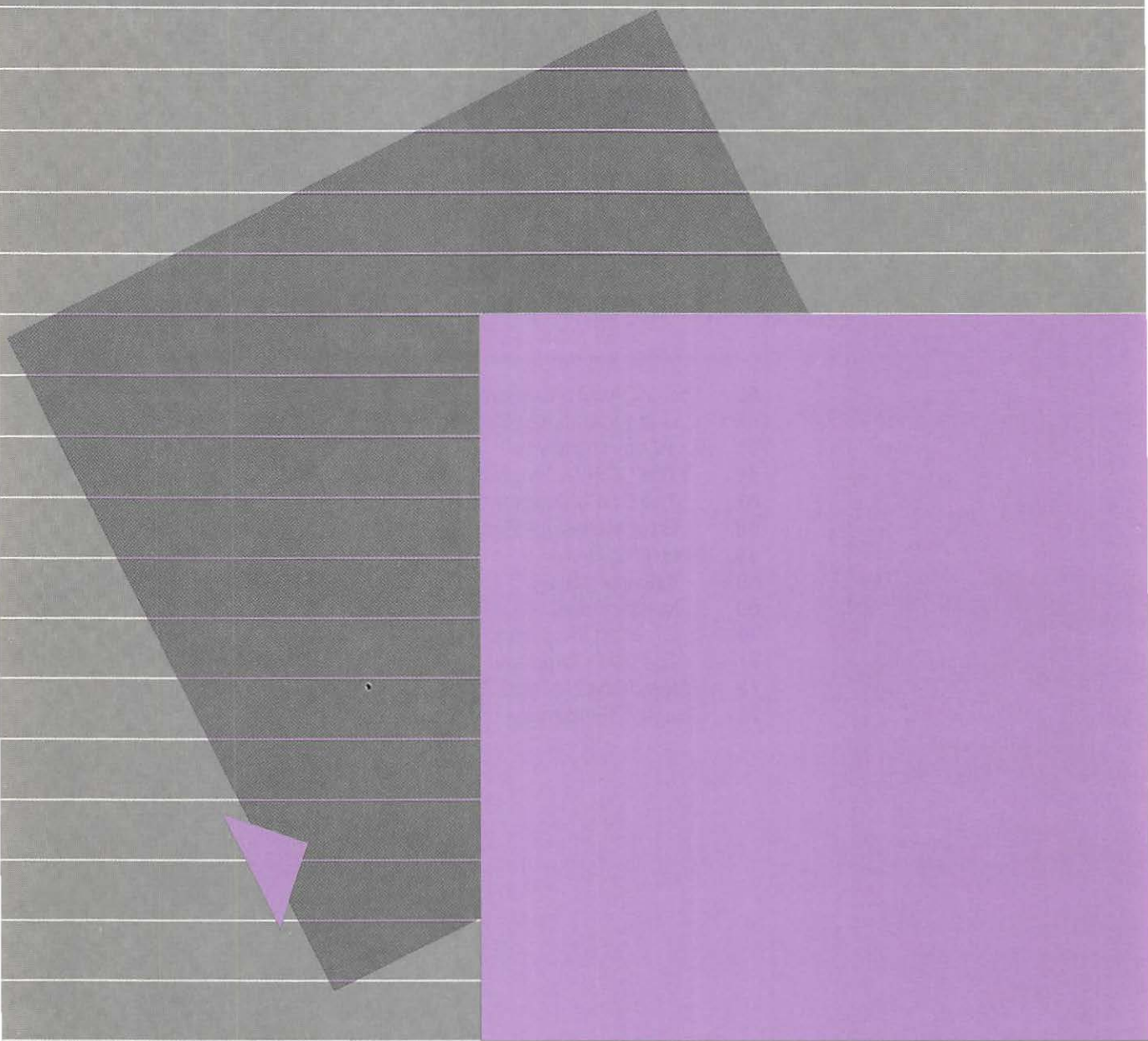
Chapter 2 Summary

In this chapter, you learned how DOS works and the form, or syntax, used by DOS commands. You also learned how files and volumes relate to each other and to DOS.

And you found out what happens when you start DOS and how DOS fits in with other programs in your Apple II computer.

Using Disks

63	The CATALOG Command
65	Understanding a Catalog
66	The Copy Programs
66	The COPYA Program for Applesoft BASIC
67	The COPY Program for Integer BASIC
68	Error Messages From the Copy Programs
69	Preparing Disks
69	Initialized Disks
69	Master Disks
70	Greeting Programs
71	The INIT Command
72	Determining Slot and Drive Numbers: The SLOT# Program
73	Chapter 3 Summary



Using Disks

This chapter discusses how to manage disks. It will explain how to get a listing of the files that are on a disk, copy the contents of entire disks, prepare a disk for DOS, and determine the current slot and drive numbers. These topics are covered more comprehensively than in Chapter 1.

The CATALOG Command

A **catalog** is a list of all the files stored on a disk.

The listing of the contents of a disk is called a **catalog**. When you issue a CATALOG command, a list is displayed of all the files on a disk with a few statistics about each one.

In addition to the file's name, the catalog shows whether a file is locked, what its type is, and its size.

On the disk, the catalog is a series of file entries that contain the name, location, and characteristics for each file. A disk's catalog has room for 105 file entries. Usually, the disk runs out of file space before the disk's catalog fills up.

When a catalog is longer than the display can show at one time, you must press a key to continue the listing. Pressing the **(SPACE)** bar to do this is a good habit to get into. If you want to see the beginning of the catalog again, issue another CATALOG command.

An explanation of the syntax notation in this manual can be found in Chapter 2, "DOS Commands."

The command has this syntax:

CATALOG [,Sn] [,Dn]

If you include a volume number argument in the CATALOG command, DOS will ignore it. If you do not use the slot number (Sn) and drive number (Dn) arguments with the CATALOG command, DOS uses the default values, which are the numbers of the disk drive and slot last specified.

Reminders:

- DOS commands must be typed in uppercase. If you are using an Apple IIe computer, press **(CAPS LOCK)** until it locks into its "on" position.
- Press **(RETURN)** after you have typed the command and corrected any mistakes.

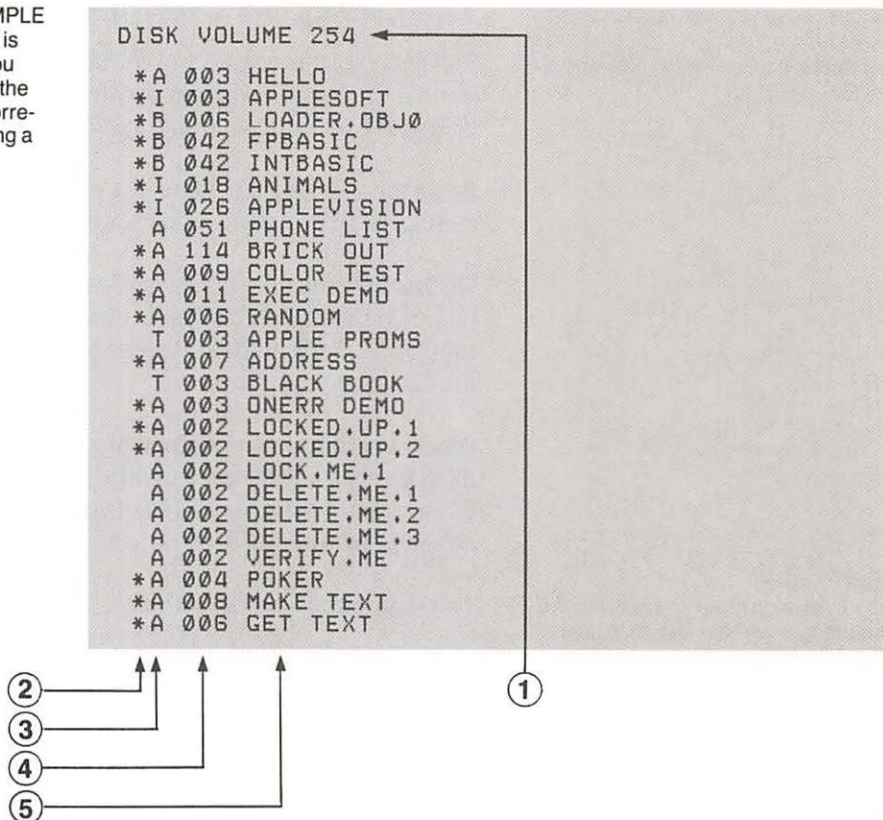
For instance, if you have DOS running and you want to see the contents of the SAMPLE PROGRAMS disk, put the SAMPLE PROGRAMS disk in drive 1 and type

C A T A L O G (SPACE) D 1

Notice that you do not need a comma in this command when you use only one argument.

In response to this command, your display will fill with the catalog of the SAMPLE PROGRAMS disk (Figure 3-1). Press the **(SPACE)** bar to continue the catalog.

Figure 3-1. The Catalog of the SAMPLE PROGRAMS Disk. When a catalog is longer than one screen can hold, you should press **(SPACE)** bar to continue the listing. The numbers in the figure correspond to those in the "Understanding a Catalog" section.



Understanding a Catalog

The following is an explanation of the parts of a catalog. The numbers correspond to those in Figure 3-1.

- ① Volume Number The number assigned to the disk when initialized, the volume number, is shown in the catalog's heading.

Reading left to right across the catalog:

- ② Lock Status The first column indicates whether or not the file is locked. An asterisk (*) indicates that a file is locked, or protected, and you cannot change its contents until you unlock that file.

- ③ File Type The one-letter code in the second column indicates the file type.

- ④ File Size The third column shows the number of sectors the file uses.

Note: If an individual file exceeds 255 sectors, the display of file size starts over at 000. This does not affect the use of the file, but gives a misleading impression of the size of the file.

- ⑤ Filename The last column shows the filename.

The **LOCK** and **UNLOCK** commands are explained in Chapter 4.

File types are defined in Table 3-1.

A **sector** holds 256 characters. The *DOS Programmer's Manual* has more on sectors.

Table 3-1. File Types. File types determine the form of DOS commands necessary to access a file and, in some cases, which of two similar programs to use. The languages available in a particular system determine whether Integer files, Applesoft files, or both can be used.

For an explanation of the files on the **SAMPLE PROGRAMS** disk, see Appendix C.

File Type	Code	Used With
Applesoft BASIC	A	Applesoft BASIC; Created By SAVE
Binary	B	Binary Commands; Created By BSAVE (*)
Integer BASIC	I	Integer BASIC; Created By SAVE
Text	T	Text File Commands; Created By OPEN (*)
Relocatable	R	Assembly Language (+)
Unused	S	Reserved for Future Use

* See the *DOS Programmer's Manual*
+ See the *6502 Assembler/DOS Tool Kit*

The Copy Programs

There are two similar programs that copy a whole disk. Which one you use depends on which BASIC you are using. The COPYA program is used when you are running Applesoft BASIC. The COPY program is used when you are running Integer BASIC.



Warning

The copy programs begin by initializing the duplicate disk. If any information is on the disk, it will be erased.

Chapter 1, the tutorial, has step-by-step instructions for using the copy programs with one or two disk drives.

The copy programs start by asking about slot and drive numbers. That is, before using a default slot or drive number, the program checks with you. It displays a default number that you can accept by pressing **RETURN** or you can type the number you want.

A Word to the Wise: It is strongly recommended that you make at least one backup copy of your SYSTEM MASTER disk. Put the original SYSTEM MASTER in a safe place (someday you may want to copy it again) and use the copy.

To **write-protect** a disk means to prevent any new information from being written onto it. This is accomplished by covering the disk's **write-enable notch** with a **write-protect tab**. See Chapter 1.

Write-protect the copy by covering the write-enable notch with a silver write-protect tab.

The COPYA Program for Applesoft BASIC

When you're running Applesoft BASIC, use the COPYA program to copy the entire contents of a disk to a blank disk. You can tell you are using Applesoft by the **⌈** prompt.

To use the COPYA program on the SYSTEM MASTER, type

```
R U N SPACE C O P Y A , D 1
```

After the program is read into main memory, the display will show **APPLE DISK DUPLICATION PROGRAM** and the default values for slot and drive numbers for the original and duplicate disks.

In this program, the disk that is copied is called the *original* disk; the copy is called the *duplicate* disk.

If you do not know which slot your disk controller card is in, you can run the **SLOT#** program, which is discussed later in this chapter. Or you can open the computer and see for yourself—check your owner's manual.

If you want to accept the default value, press **RETURN**. If you want a slot or drive number other than that shown, type the number. Then press **RETURN** when you are ready to begin copying.

If you are using one disk drive, the computer will tell you when to switch between original and duplicate disks. If you are using two or more drives, put the original in drive 1, the duplicate in drive 2, and accept all the default values. You will not have to move any disks during the process.

The program will advise you of its progress with the messages **INITIALIZING**, **WRITING**, and **READING**. When the copy is completed, you will be asked if you want to make another copy. If you type **N** for "no," you will leave the copy program, and the prompt will return.

If you want to make another copy, type **Y** for "yes," and the program will start over.

If you want step-by-step instructions for using the copy programs with one or two disk drives, see Chapter 1.

Before copying any original disk, it's good practice to protect the contents from being erased accidentally. Write-protect the disk.

The COPY Program for Integer BASIC

When you're running Integer BASIC, use the COPY program to copy the entire contents of a disk to a blank disk. You can tell you are using Integer BASIC by the **>** prompt.

To use the COPY program on the SYSTEM MASTER, type

```
R U N SPACE C O P Y , D 1
```

And follow the messages on the display. The COPY program functions exactly like the COPYA program. Refer to the previous section for details.

If you want step-by-step instructions for using the copy programs with one or two disk drives, see Chapter 1.

Error Messages From the Copy Programs

Depending on the program you are using, you may see
***** UNABLE TO READ ***** (COPYA) or
I/O ERROR STOPPED AT (line number) (COPY)
if there is no disk in the drive specified, if the disk isn't completely
inserted into the drive, if the door to the drive isn't fully closed, or if
you indicated the wrong slot or drive number.

Depending on the program you are using, you also may see
***** UNABLE TO WRITE ***** (COPYA) or
I/O ERROR STOPPED AT (line number) (COPY) if
your duplicate disk isn't properly seated in the disk drive, if the
duplicate disk has no write-enable notch or the notch is covered, if the
drive door isn't fully closed, or if you indicated the wrong slot or drive.

If COPYA cannot read from the source disk or write to the duplicate,
you'll see DO YOU WISH TO MAKE ANOTHER COPY?
Type (N), press (RETURN), and return to BASIC. If COPY reports
I/O ERROR, it returns to BASIC.

If COPY cannot initialize the duplicate disk, you'll see I/O
ERROR and then the Integer BASIC prompt. If COPY cannot
transfer the information, you'll be asked DO YOU WISH TO
MAKE ANOTHER COPY? Type (N) for "no" and press (RETURN).
You'll see the Integer BASIC prompt.

Before you run the copy program again, check your SYSTEM
MASTER disk, the duplicate disk and its write-enable notch, and your
slot and drive numbers.

One More Possibility: There is another, but slower, way to copy an entire
disk. Use the FILEM program. FILEM is useful when you want to copy a
single file or a set of files that have similar names, like FOO, FOO1, and
FOO2.

FILEM is discussed in Chapter 4.

Preparing Disks

There is no information at all on a new disk, fresh from the manufacturer. To use a disk with DOS, you must put special information on it. Placing this information on the disk is called **initializing**, or **formatting**.

You can prepare a disk two different ways to receive information: you can simply initialize the disk or you can make it a **master** disk.

Initialized Disks

An initialized disk is prepared with the INIT command. The command organizes the surface of the disk into **tracks** and **sectors** so it can receive information, and it puts a copy of DOS and a greeting program on that disk.

Any disk that has been initialized is memory-size dependent: the size of the system that initializes the disk determines the size of the system that can use the disk.

For example, if a disk is initialized on a 32K system

- it can only be used on a system with 32K or more of memory
- it will not execute programs that together with DOS use more than 32K of memory (since DOS uses 10.5K of memory, other programs can use up to 21.5K)

If you will be using an Apple system with the same memory size every day, disks initialized on your system will serve you very satisfactorily. However, if you can use a disk initialized on a smaller system (for example, if you borrow a friend's disk), you may get an **OUT OF MEMORY** message.

By the Way: In some manuals, an initialized disk is called a *slave* disk because it is chained to the size of the computer in which it was prepared.

Master Disks

A master disk has a self-relocating DOS that uses memory efficiently on any size system. That is, each time you start up a master disk, DOS is put as high as possible into memory. That way, a larger block of memory is available for your program.

Master disks, unlike initialized disks, are not memory-size dependent. They can be run on systems with various memory sizes. The SYSTEM MASTER is an example of a master disk. You can run it on any Apple computer.

The MASTER program is described in the *DOS Programmer's Manual*.

If you have two or more Apple computers with different memory sizes, you may wish to convert your initialized disks to master disks using the MASTER program.

Greeting Programs

All disks that have been initialized with DOS have a greeting program. The contents of that program can vary from a simple
INITIALIZED DISK CREATED ON MY BIRTHDAY
to a complex program that runs automatically when the disk is started up.

When you want a simple greeting program, you can copy an existing one from a disk, or you can write a new one.

You can put whatever information you want in a greeting program. Since it announces the disk, it is handy to include

- The date you initialize the disk
- The size of the system you are using
- A message to make you smile whenever you use the disk

To write your own greeting program, clear main memory by issuing a NEW command, and then type a BASIC program to display whatever message you want. For example:

```
10 PRINT "MERRY CHRISTMAS!"  
20 PRINT "64K DISK INITIALIZED 25 DECEMBER 1983"  
30 END
```

Incidentally: You may use whatever name you like for the greeting program. Since a greeting program looks like any other program in a disk's catalog, it helps to use a standard name for all your greeting programs. In this manual, HELLO is used as the standard name.

Once you have created a greeting program, you use the INIT command to put the greeting on the disk and to complete the initializing procedure.

The **INIT** command initializes a disk.

A **turnkey program** runs automatically when the disk is started. For more information, see the *DOS Programmer's Manual*.

The INIT Command

The **INIT** command is used to create an initialized disk that you can use with DOS. The command organizes the surface of the disk into tracks and sectors and puts a copy of DOS and a greeting program on that disk.

When initializing is complete, DOS does not display any confirming message; you just see the prompt again. However, you can now use this disk to start DOS up, and DOS can now write to and read from the disk.

When an initialized disk is started, DOS checks the directions **INIT** stores on the disk and runs the program that they point to. For example, when those directions point to a greeting program, the greeting message from the program is displayed. When the directions point to some other program (perhaps a game program or application program), that program starts running. A program that is started automatically is called a **turnkey program**.

An initialized disk is reusable. When you no longer need the information on a disk, you can reinitialize it. This erases any information on the disk and gives you a fresh start.

Warning

When you initialize a disk, anything stored on it is erased. Make sure you do not initialize a disk that contains information or programs you want to save.

Use the **INIT** command only in immediate execution; that is, directly from the keyboard.

The command has this syntax:

INIT fn [,Sn] [,Dn] [,Vn]

When you replace **fn**, which stands for “filename,” with the name of a program that is currently in memory, DOS puts the program on the specified disk.

For example, if the three lines of the Christmas greeting program from the previous section were in memory, you would simply type

```
INIT SPACEHELLO
```

and DOS would put that program on the disk in the default drive.

If there is no program currently in memory, DOS creates an empty file on the disk.

The slot, drive, and volume arguments are optional. When you omit the slot and drive options, DOS uses the values of the previous DOS command. If you do not know the default, specify exactly what you want.

When you omit the volume argument altogether, specify volume number 0, or type ☐ without a number, DOS uses the default volume number, 254. If you specify a volume number, DOS assigns that volume number to the disk.

Use INIT to assign a particular volume number.

INIT is the only command that assigns a volume number. When the volume option is used with other DOS commands, the number specified must match the number assigned by INIT.

The step-by-step procedure for initializing a disk is described in Chapter 1.

Hint: If you have several blank disks, it would be a good idea to initialize a few of them now. Then the initialized disks will be ready when you need them.

Determining Slot and Drive Numbers: The SLOT# Program

The SLOT# program reports the slot and drive numbers that you are currently using, that is, the current default values.

Some programs, such as the copy programs, ask you for slot and drive numbers. With some commands, such as the INIT command, you may damage the contents of a disk unless you know which slot and drive DOS will use by default.

To confirm the current slot number and the current drive number, you can run the SLOT# program on your SYSTEM MASTER disk. With the SYSTEM MASTER in drive 1, type:

```
R U N SPACE S L O T SHIFT #
```

The program's report will look something like this:

```
CURRENT SLOT IS 6 AND DRIVE IS 1
```

Chapter 3 Summary

Commands

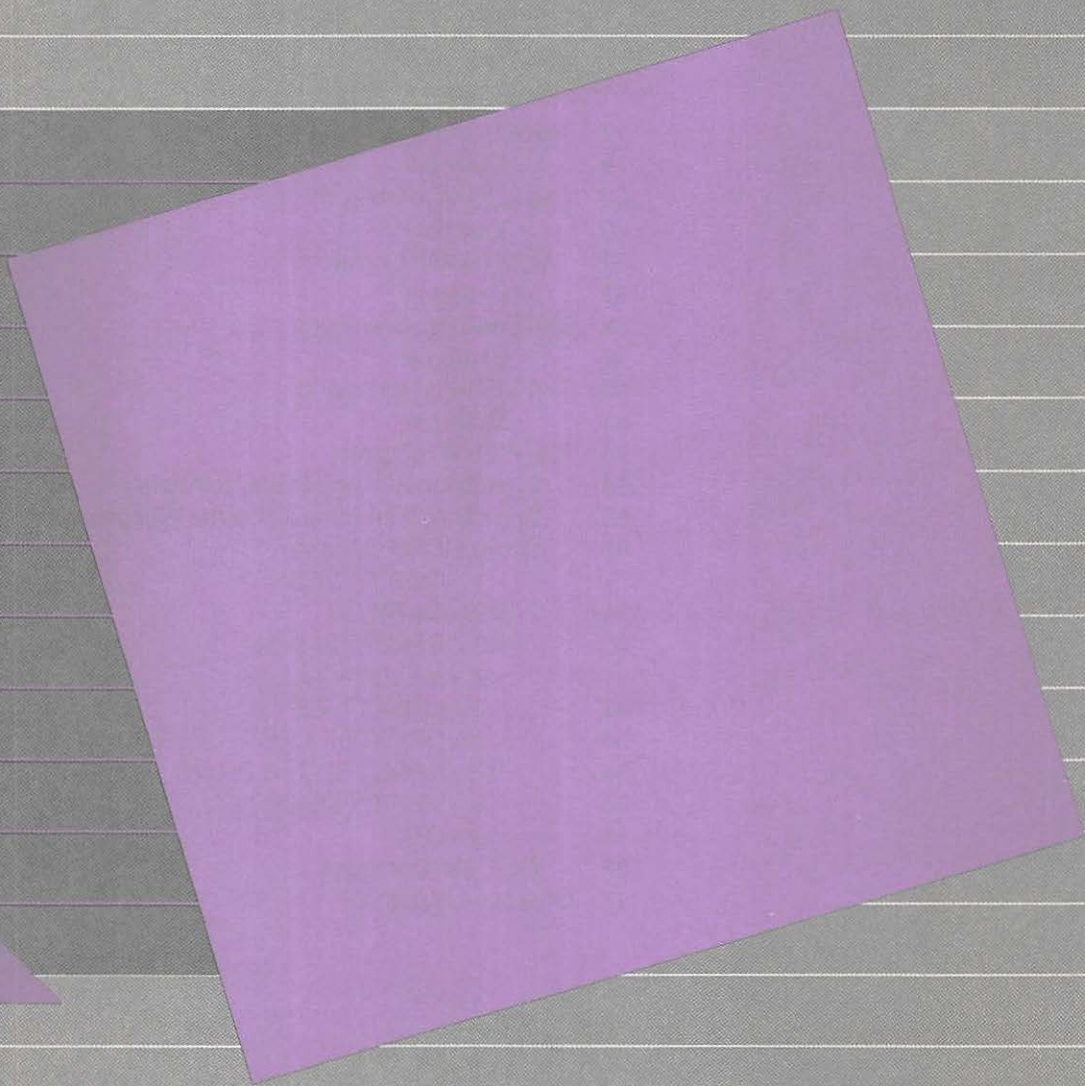
CATALOG [,Sn] [,Dn]	displays a listing of all the files on your disk. In addition to the filename, the catalog lists the size, type of file, and whether the file is locked.
INIT fn [,Sn] [,Dn] [,Vn]	prepares a disk to receive information. The INIT command places DOS and a greeting program on the disk.

Programs

COPYA	copies an entire disk when you are running Applesoft BASIC. It is not necessary to initialize the duplicate disk.
COPY	copies an entire disk when you are running Integer BASIC. It is not necessary to initialize the duplicate disk.
SLOT#	reports the current default values for slot and drive numbers.

Using Files

77	The RENAME Command
79	Example
80	The LOCK Command
80	Example
81	The UNLOCK Command
81	Example
82	The DELETE Command
82	Example
83	The VERIFY Command
83	Example
84	The FILEM Program
85	Specifying Slot and Drive Numbers
85	Specifying Filenames With the Wildcard Character
87	Using FILEM Options
87	<1> COPY FILES
90	<2> CATALOG
90	<3> SPACE ON DISK
91	<4> UNLOCK FILES
92	<5> LOCK FILES
94	<6> DELETE FILES
94	<7> RESET SLOT & DRIVE
94	<8> VERIFY FILES
95	<9> QUIT
95	FILEM's Messages
97	Chapter 4 Summary



Using Files

This chapter describes the DOS commands that allow you to keep track of and manipulate the files on your disks. You can use these commands to change the name of a file; to create more room on a disk; to protect some information from being accidentally destroyed; and to verify that a file on the disk can be read.

The FILEM program, discussed at the end of the chapter, allows you to perform operations on several or all the files on a disk at the same time.

This chapter will introduce each command and then use an example to illustrate how the command works. When you are trying the examples, it's a good idea to use a copy of the SAMPLE PROGRAMS disk rather than the original. That way, if someone else wants to learn about DOS after you're finished, the sample programs will be in their original state.

The RENAME Command

The RENAME command allows you to change the name of any file on a disk. You can use it to change the name of a BASIC file, text file, or binary file.

The command has this syntax:

```
RENAME fn1,fn2 [,Sn] [,Dn] [,Vn]
```

The RENAME command changes the name of a file from the name indicated by fn1 to the name indicated by fn2. The filename represented by fn1 is the name of an existing file, which must be unlocked. The new name, fn2, should be unique for the disk.

The slot number, drive number, and volume number arguments are optional. You only need to specify a number when you don't want to use the default. These arguments are described in Chapter 2.

An explanation of the syntax notation in this manual can be found in Chapter 2, "DOS Commands."

The **UNLOCK** command is explained later in this chapter.

If the file you specified as `fn1` does not exist, you'll get the `FILE NOT FOUND` error message. If the file is locked, you'll get the `FILE LOCKED` error message.

Incidentally, you can also change the name of a file by loading the file into memory and then saving a copy to the disk with a new name.



Warning

When `RENAME` is executed, DOS does not check to see if the second filename (`fn2`) already exists on the disk. DOS will carry out the `RENAME` command and leave you with two files with the same name.

When there are duplicate filenames on a disk, DOS only recognizes the first file, which makes it hard to access the second file. For example, if your disk's catalog looked like this:

```
HELLO
LETTERS
SORT
PROGRAM, 1
```

and you issued the command

```
RENAME PROGRAM, 1, SORT
```

Your catalog would then look like this:

```
HELLO
LETTERS
SORT
SORT
```

You would not be able to use the second `SORT` until you renamed the first one.

To correct the situation, rename the first file named `SORT`. Be sure the new name is unique to the disk. Then you can use the second file named `SORT`.

To prevent this from happening, you should always check the catalog before changing the name of a file.

Example

1. Find the SAMPLE PROGRAMS disk and put it into drive 1. Check the catalog by typing

```
C A T A L O G , S 6 , D 1
```

Press **(RETURN)**. As well as giving you a listing of the files on the SAMPLE PROGRAMS disk, this command sets the defaults to slot 6, drive 1. The catalog includes the files HELLO, ANIMALS, BRICK OUT, and PHONE LIST.

Press the **(SPACE)** bar to continue the catalog.

2. To change the name of PHONE LIST (fn1) to DIRECTORY (fn2), type

```
R E N A M E SPACE P H O N E SPACE L I S T ,  
D I R E C T O R Y
```

When you press **(RETURN)**, DOS changes the file's name.

3. Now issue the CATALOG command again to see the result of your order. Remember to press **(SPACE)** bar to see the rest of the catalog.
4. Now change the name back to give yourself a little more practice with the command. Type

```
R E N A M E SPACE D I R E C T O R Y , P H O N E SPACE  
L I S T
```

Write-protecting a disk is explained in Chapter 1, "About Disks and Disk Drives."

The LOCK Command

The LOCK command locks a file: it protects the file from being accidentally changed, deleted, or renamed. However, a locked file can be loaded and run.

Trying to change, delete, or rename a locked file results in a `FILE LOCKED` error message. To change a locked file, you must first unlock it with the UNLOCK command.

When you look at the catalog of a disk, the locked files are marked by asterisks to the left of their file types.

By the Way: The LOCK command only works on one file at a time. To protect a whole disk, simply cover its write-enable notch with a silver write-protect tab.

The command has this syntax:

`LOCK fn [,Sn] [,Dn] [,Vn]`

The filename, `fn`, indicates the file to be locked.

The slot number, drive number, and volume number arguments are optional. You only need to specify a number when you don't want to use the default. These arguments are described in Chapter 2.

Example

1. With the SAMPLE PROGRAMS disk in drive 1, display the catalog of its files by issuing a CATALOG command.

Notice that the file LOCK.ME.1 does not have an asterisk, indicating that it is not locked.

2. Lock the file LOCK.ME.1 by typing

```
LOCK SPACE LOCK.ME.1
```

3. Use the CATALOG command again to check that LOCK.ME.1 is now protected from accidental change. It should have an asterisk at the beginning of its catalog entry.

The UNLOCK Command

The UNLOCK command unlocks a file; that is, it removes protection from a file so that you can delete, rename, or change the file. The UNLOCK command works on one file at a time.

The command has this syntax:

```
UNLOCK fn [,Sn] [,Dn] [,Vn]
```

The filename, fn, indicates the file to be unlocked.

The slot number, drive number, and volume number arguments are optional. You only need to specify a number when you don't want to use the default. These arguments are described in Chapter 2.

Example

1. With the SAMPLE PROGRAMS disk in drive 1, display the catalog of its files by issuing a CATALOG command.

Notice that the files LOCKED.UP.1 and LOCKED.UP.2 have asterisks indicating that they are locked.

2. Unlock the file LOCKED.UP.1 by typing

```
UNLOCK SPACE LOCKED.UP.1
```

3. Use the CATALOG command again to verify that the file is now unlocked. The file should not have an asterisk in front of its file type.

The **DELETE** Command

The **DELETE** command lets you remove a file from a disk. It is used to get rid of unwanted versions of files and to make space on a disk for new files.

The command has this syntax:

DELETE fn [,Sn] [,Dn] [,Vn]

The file indicated by fn must be unlocked. (That is, you do not see an asterisk in front of the catalog entry for the file). If the file does not exist, you'll get a **FILE NOT FOUND** message when you try the command. If the file is locked, you'll get a **FILE LOCKED** message.

The slot number, drive number, and volume number arguments are optional. You only need to specify a number when you don't want to use the default. These arguments are described in Chapter 2.

Example

1. With the **SAMPLE PROGRAMS** disk in drive 1, display the catalog of its files by issuing a **CATALOG** command.

Included in the catalog is the file **DELETE.ME.1**.

2. Delete the file **DELETE.ME.1** by typing

DELETE SPACE DELETE.ME.1

3. Use the **CATALOG** command again to check that **DELETE.ME.1** is no longer listed and therefore no longer on the disk.

The VERIFY Command

The VERIFY command lets you make sure a file was written on the disk correctly and that DOS can still read it. If the file can be verified, it's safe to assume that the information on the disk has been stored correctly and can be retrieved whenever you want.

If DOS can read the file successfully, it displays the prompt. If DOS finds that it cannot read the file (the file is damaged or written incorrectly), it displays the message `I/O ERROR`.

To verify a file, DOS simply reads the file from the disk into an area in memory that is not currently in use. This does not affect a program that might be in memory already.

The command has this syntax:

`VERIFY fn [,Sn] [,Dn] [,Vn]`

Any type of file may be verified. If the file does not exist, you'll get a `FILE NOT FOUND` message.

The slot number, drive number, and volume number arguments are optional. You only need to specify a number when you don't want to use the default. These arguments are described in Chapter 2.

Example

1. With the SAMPLE PROGRAMS disk in drive 1, display the catalog of its files by issuing a CATALOG command.

Included in the catalog is the file VERIFY.ME.

2. Verify the file by typing

`VERIFY SPACE VERIFY.ME`

Remember, when DOS can read the file, no special message is displayed.

FILEM lets you manage individual files as well as whole volumes.

The FILEM Program

So far, this chapter has discussed the DOS commands that rename, lock, unlock, delete, and verify files one at a time. The FILEM program on your SYSTEM MASTER disk offers you these same functions and a few more. It allows you to perform operations on an individual file or a set of files on a disk at the same time.

To use FILEM, insert the SYSTEM MASTER disk in drive 1. Then issue a RUN command. Type

```
R U N SPACE F I L E M , D 1
```

After the message EXECUTING FID appears briefly, you'll see FILEM's **menu** on the display (Figure 4-1).

A **menu** is a list of choices.

Figure 4-1. FILEM's Menu. You see FID VERSION M because FILEM runs the binary program FID.

```
*****
*                APPLE II FILE DEVELOPER                *
*                FID VERSION M                          *
*                COPYRIGHT APPLE COMPUTER, INC., 1979    *
*****

CHOOSE ONE OF THE FOLLOWING OPTIONS
      <1> COPY FILES
      <2> CATALOG
      <3> SPACE ON DISK
      <4> UNLOCK FILES
      <5> LOCK FILES
      <6> DELETE FILES
      <7> RESET SLOT & DRIVE
      <8> VERIFY FILES
      <9> QUIT

WHICH WOULD YOU LIKE? █
```

To choose a command, type the number in the angle brackets (< >) next to the command. Then press **RETURN**. Type only the number, not the angle brackets.

Specifying Slot and Drive Numbers

The first time FILEM executes options 1, 2, 3, 4, 5, 6, or 8 it will ask for slot and drive numbers. You'll see `SOURCE SLOT?` then `DRIVE?` *Source* refers to the disk from which you want to read information.

Switching from one option (between 1, 2, 3, 4, 5, 6, and 8) to another also will cause prompting for slot and drive numbers. If you want to repeat the option, the program will use the numbers you've already specified.

If you want to repeat the option, but you want to change the defaults, you must execute option 7 first.

By the Way: If you give numbers for non-existent slots or drives, FILEM beeps and displays the message `INVALID SLOT` or `INVALID DRIVE`. Simply type the correct number.

Specifying Filenames With the Wildcard Character

A **wildcard** is a character that stands for any character sequence in a filename.

Some of FILEM's commands ask for a filename. You may spell out the filename fully, letter by letter. Or you may use only a few of the letters in the name and a special character called a **wildcard**. This special character stands for any sequence of characters in a corresponding position in filenames. The wildcard character is the equal sign (=).

For example, `A=S` can represent any filename beginning with A and ending with S. `A=S` can refer to all these filenames:

- | | |
|-----------------------|---------------------------------------|
| • AS | The = stands for no characters. |
| • ACTS | The = stands for the characters CT. |
| • ACCESS | The = stands for the characters CCES. |
| • ALLS WELL THAT ENDS | The = stands for LLS WELL THAT END. |

When you use the wildcard character, FILEM asks if you want to approve each file before FILEM processes it. It asks `DO YOU WANT PROMPTING?` To have FILEM quietly process all the files that match your filename specification, type `N` for "no" prompting and press `RETURN`.

When you want to confirm each file in a command, type **Y** and press **RETURN**. FILEM then displays each filename before doing anything to the file. To have FILEM process that file, type **Y** and press **RETURN**. To have FILEM skip over that file, type **N** and press **RETURN**. By displaying the message **CANCELLED**, FILEM tells you it skipped that file.

Helpful Hint: When you've processed all the files on a disk that you want to process, type **Q** and press **RETURN** to return to the menu and cancel the command.

Figure 4-2. Using the Wildcard to Represent Parts of Filenames

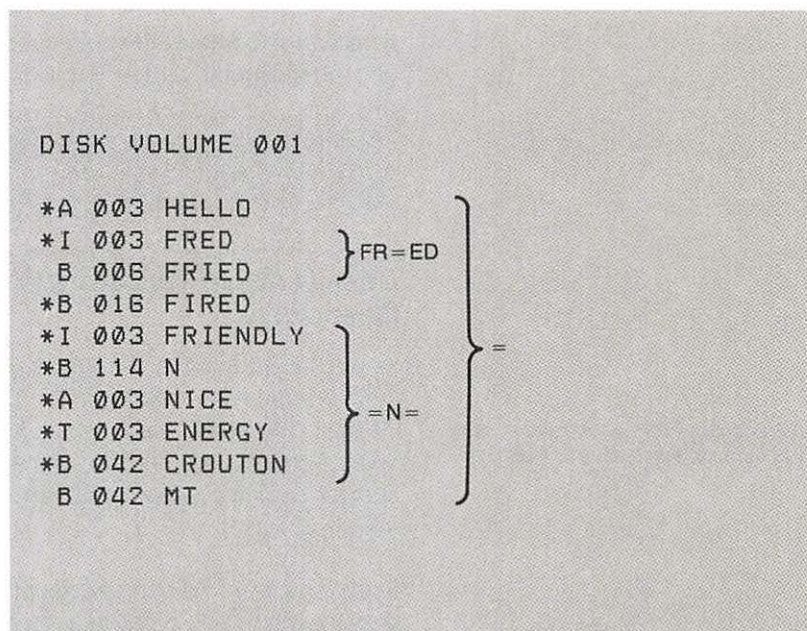


Figure 4-2 shows some examples of using the wildcard character to work with groups of files.

- **FR=ED** represents all the files with names that begin with *FR* and end with *ED*. It does not represent the file named *FIRED* because an *I* is the second character, not *R*, or the file named *FRIENDLY* because the filename ends with *LY*, not *ED*.
- **=N=** represents all files with names that contain an *N*. *NICE* has one at the beginning; *ENERGY* and *FRIENDLY* have one in the middle; and *CROUTON* has one at the end. You'll also get the file named *N*.
- **=** represents all the files on the disk. If you use the wildcard by itself with FILEM, all the files on the disk will be used in the option.

Using FILEM Options

Six of the eight options in the FILEM program correspond to DOS commands that you use to process individual files. However, using the FILEM wildcard lets you process several files with one command.

When FILEM finishes processing an option, you'll see `PRESS ANY KEY TO CONTINUE`.

Press Almost Any Key: `PRESS ANY KEY TO CONTINUE` means type any key except `(RESET)`, `(SHIFT)`, `(CONTROL)`, `(F1)`, `(F2)`, or `(CAPS LOCK)`. We suggest that you use the `(SPACE)` bar.

<1> COPY FILES

This option copies one or more files from one disk to another.

Note: Unlike the COPY and COPYA programs, FILEM does not initialize the duplicate, or destination, disk.

If you plan to use FILEM to make copies, initialize some blank disks before running the program. The INIT command is covered in Chapter 3.

Before FILEM executes option 1, it asks for the slot and drive numbers of the source disk. You'll see `SOURCE SLOT?`, then `DRIVE?` *Source* refers to the disk from which you want to read information.

It also asks for the slot and drive numbers of the destination disk. You'll see `DESTINATION SLOT?`, then `DRIVE?` *Destination* refers to the disk that will receive the information.

- When you have one disk drive connected to your computer, the slot and drive numbers for the source and destination disks are the same. FILEM will tell you when to insert the source and the destination disks into the disk drive.
- When you are using two drives, the slot number is the same for both disks, but the drive numbers are different. FILEM says simply: `INSERT DISKS`.

After FILEM has the information on slot and drive numbers, you'll see `FILENAME?` prompting you for the name of the file to copy.

This example takes you through the process of using FILEM to copy with two drives. You should have the SYSTEM MASTER in drive 1. If you do not have FILEM's menu on your display, start the program with a RUN command. Type

R U N SPACE F I L E M , D 1

What You Do

What You Get

1. To select option 1 from the menu, type

1

and press RETURN.

You'll see COPY FILES on your display and a prompt from FILEM asking for the slot number of the source disk, SOURCE SLOT?

2. Type

6

for slot 6 and press RETURN. If your disk controller card is installed in a different slot, type that number.

You'll see DRIVE? on your display. The program is asking which disk drive holds the original disk.

3. Type

1

for drive 1 and press RETURN.

You'll see DESTINATION SLOT? on your display. Now the program wants information about the destination disk.

4. Type

6

for slot 6 and press RETURN. If your disk controller card is installed in a different slot, type that number.

You'll see DRIVE? on your display. The program is asking which disk drive holds the destination disk. Remember: the destination disk must already be initialized.

5. Type

for drive 2 and press .
If you are using only one disk drive, type instead.

FILEM should be displaying all the slot and drive numbers that it will use. Up to this point, it has not copied anything from the disk. FILEM has simply found out where to get information and where to put it.

FILENAME? will prompt you for the name of the file or files to copy.

6. This is the point at which you can use a wildcard to represent part of a filename. You could type , , or if you wanted to copy the file MASTER.

But will copy the five files on the MASTER disk that have an *M* in their filename, which you don't want to do. Although if you know that using a wildcard will only copy the file or files that you want, it can be a convenient shortcut, such as . And then you can always use the prompting option to OK each file for the command individually.

Whenever you use a wildcard, the program will ask DO YOU WANT PROMPTING? If you do, type for yes, and you will be allowed to decide whether each file indicated by the wildcard should be copied or not. If you want the file copied, type after the name of the file. If you want to skip the file, type .

When you are only copying one file, however, it is safer to type the whole name. So type

M A S T E R

and press **RETURN**.

The message **INSERT DISK(S) PRESS <ESC> TO RETURN TO MAIN MENU OR ANY OTHER KEY TO BEGIN** tells you that the program is ready to begin.

7. Be sure the source disk is in drive 1 and the initialized disk is in drive 2. Then press the **SPACE** bar when you are ready to start copying.

FILEM finds the file you asked for and begins the copy operation; it displays **FILE MASTER**. If the same filename already exists on the disk in drive 2, you'll hear a beep, and FILEM will display more instructions. When copying is finished, you'll see **DONE**.

Finally, you'll see **PRESS ANY KEY TO CONTINUE**.

8. Press the **SPACE** bar.

FILEM's menu of options returns.

Copying Locked Files: The FILEM copy command will copy locked files. The copied files will also be locked on the destination disk.

<2> CATALOG

This option displays a disk's catalog. It is handy to use when you're running FILEM and need to check a disk's contents: you don't have to exit from the FILEM program, issue the DOS CATALOG command, and then return to FILEM with a RUN command.

For example, if you want to look at the catalog of the **SAMPLE PROGRAMS** disk, remove the **SYSTEM MASTER** from drive 1 and insert the **SAMPLE PROGRAMS** disk. Select option 2, **CATALOG**. FILEM will ask for slot and drive numbers. Be sure to indicate drive 1.

FILEM will display a catalog of the **SAMPLE PROGRAMS** disk. To return to the options menu, press the **SPACE** bar.

<3> SPACE ON DISK

When you select option 3, FILEM reports how many sectors are used and how many are unused on your disk.

Try the option now. With the SAMPLE PROGRAMS disk still in drive 1, select option 3. You'll see something like this on your display:

```
SPACE ON DISK
0117 SECTORS FREE
0443 SECTORS USED
```

To return to the menu, press the **SPACE** bar.

<4> UNLOCK FILES

This option lets you unlock a file.

This example shows how to use FILEM to unlock a file on a disk. Be sure the SAMPLE PROGRAMS disk is still in drive 1.

What You Do	What You Get
1. Select option 4 from the menu and press RETURN .	You'll see UNLOCK FILES on your display. FILEM asks for the slot and drive numbers of the source disk if it has not already done so.
2. Type the appropriate numbers for the disk whose files you want to unlock.	FILENAME? is displayed.

3. Type

H E L L O

and press **RETURN**.

You can use a wildcard in response to the **FILENAME?** prompt. If you do, the program will ask **DO YOU WANT PROMPTING?** If you do, type **Y** for "yes;" the name of each file indicated by the wildcard will appear in succession. When each filename is displayed, you must tell FILEM whether to unlock the file or not.

Type **Y** if you want the named file unlocked; the disk whirs briefly, **DONE** appears, and the program proceeds to the next file indicated by the wildcard. Type **N** if you do not want the named file unlocked; you'll see **CANCELLED** underneath the filename, and the prompting will continue.

You'll see **INSERT DISK(S), PRESS <ESC> TO RETURN TO MAIN MENU OR ANY OTHER KEY TO BEGIN.**

4. Be sure the disk whose files you want to unlock is in the appropriate disk drive. Then press the **SPACE** bar to begin.

You'll see **FILE HELLO** and then **DONE** when **FILEM** has unlocked the file named **HELLO**.

Finally, you'll see **PRESS ANY KEY TO CONTINUE.**

5. To return the **FILEM** menu, press the **SPACE** bar.

<5> LOCK FILES

This option locks a file so that it cannot be changed, deleted, or renamed.

The following is an example of how to use **FILEM** to lock all the files on a disk. Be sure the **SAMPLE PROGRAMS** disk is still in drive 1.

What You Do	What You Get
1. Select option 5 from the menu by typing 5 and pressing RETURN .	You'll see LOCK FILES on your display. FILEM asks for the slot and drive numbers of the source disk if it has not already done so.
2. Type the appropriate numbers for the disk whose files you want to lock.	FILENAME? is displayed.

3. Type

=

and press **RETURN**. When the wildcard is used by itself, it stands for all the files on the specified disk.

The **DO YOU WANT PROMPTING?** message is displayed.

4. Type

N

for "no" prompting since you know you want to lock all the files. Press **RETURN**.

If you wanted to confirm each file that should be locked, you would type **Y** and then respond with either **Y** or **N** as each file's name is displayed.

You'll see **INSERT DISK(S) PRESS <ESC> TO RETURN TO MAIN MENU OR ANY OTHER KEY TO BEGIN.**

5. Press the **SPACE** bar to begin.

FILEM will go ahead and process all the files that match your filename specification. You'll see **FILE** followed by a filename for each file on the disk. When **FILEM** has finished locking all the files, you'll see **DONE**.

Finally, you'll see **PRESS ANY KEY TO CONTINUE.**

5. Press the **SPACE** bar.

You'll return to **FILEM**'s menu.

<6> DELETE FILES

This option deletes a file or set of files. FILEM deletes the file from the disk and from the catalog. Once you have deleted a file, it is gone for good.

When you select option 6, you'll see `DELETE FILES` on your display, and you will be asked for the slot and drive numbers of the source disk if FILEM has not already done so. Check that the correct disk is in the disk drive.

When you see `FILENAME?`, type the name of the file that you want to delete and press `(RETURN)`. When the file has been deleted, FILEM displays `DONE` and `PRESS ANY KEY TO CONTINUE`. To return to the menu, press the `(SPACE)` bar.

<7> RESET SLOT & DRIVE

This option cancels whatever numbers you had previously set for slot and disk drive. The next time you ask for an option that requires slot and drive numbers, FILEM will request them. When you wish to use FILEM's catalog command on two different drives in succession, use this command.

When you see FILEM's menu, select option 7. FILEM displays `RESET SLOT & DRIVE`, `DONE`, and then `PRESS ANY KEY TO CONTINUE`. To return to the menu, press the `(SPACE)` bar.

<8> VERIFY FILES

When you select option 8, FILEM verifies a file. That is, FILEM tries to read the file from the disk into memory. FILEM is able to verify all types of files. This command is the same as the `VERIFY` command of DOS.

When you select option 8, you'll see `VERIFY FILES` on your display, and you will be asked for the slot and drive numbers of the source disk if FILEM has not already done so. Check that the correct disk is in the disk drive.

When you see `FILENAME?`, type the name of the file that you want to verify and press `(RETURN)`. When the file has been verified, FILEM displays `DONE` and `PRESS ANY KEY TO CONTINUE`. To return to the menu, press the `(SPACE)` bar.

If the file isn't found, you'll get a `FILE NOT FOUND` message. If the file can't be verified, you'll get the `I/O ERROR` message.

<9> QUIT

When you select option 9, FILEM returns you to BASIC. You will see the same prompt character on the display as you did before you executed the FILEM program.

Incidentally: FILEM is an Applesoft BASIC program that runs FID, a binary program on the SYSTEM MASTER disk. You can run FID directly if you choose. Issue the binary command to run a program: BRUN FID.

FILEM's Messages

Here are some of the messages you might get while using the FILEM program:

- When you are using the copy option and the destination disk already contains a file with the name you used, FILEM displays the message:

```
FILE [filename]
ALREADY EXISTS.
TYPE IN NEW FILENAME FOR THE COPY OR
<RETURN> TO REPLACE EXISTING FILE OR
<CONTROL-C><RETURN> TO CANCEL COPY
:
```

If you press **RETURN** and the destination file is unlocked, FILEM writes the contents of the source file over the contents of the destination file. If you enter a new filename, the source file is stored under that new name.

- When you are using the copy option and the destination file is locked, you will see:

```
FILE LOCKED
DO YOU WISH TO REPLACE IT ANYWAY?
```

If you type **Y** for "yes" and press **RETURN**, FILEM writes over the locked destination file with the contents of the source file. If you type **N** for "no" and press the **RETURN**, FILEM gives you a chance to assign another name. The display shows the message:

```
TYPE IN NEW FILENAME FOR THE COPY OR
<RETURN> TO REPLACE EXISTING FILE OR
<CONTROL-C><>RETURN> TO CANCEL COPY
:
```

If you press **RETURN**, FILEM writes over the locked destination file with the contents of the source file. However, if you press **CONTROL-C** and **RETURN**, FILEM does not copy the file. **CANCELLED** is displayed and you return to the menu if you were copying only one file. If you were using the wildcard to copy more than one file, only the command for the file in question is canceled.

- When FILEM cannot find a source file, it displays the message:

NO FILES SELECTED

Check the catalog and then try the command again.

- If you give an improper slot or drive number, or an invalid filename, you'll see **INVALID SLOT**, **INVALID DRIVE**, or **INVALID FILENAME**.

Chapter 4 Summary

Commands

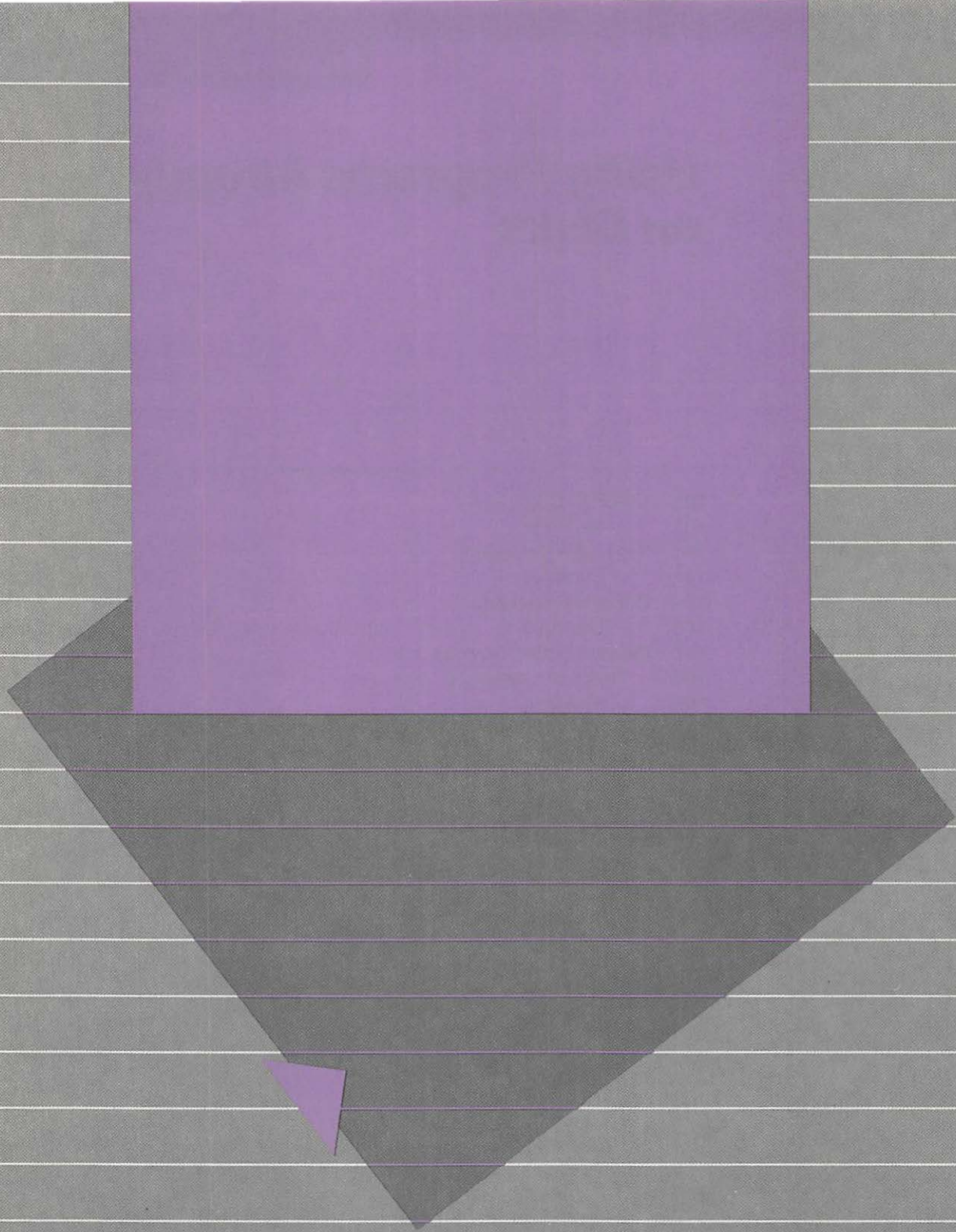
RENAME fn1,fn2 [,Sn] [,Dn] [,Vn]	changes the name of a file from the name fn1 to fn2.
LOCK fn [,Sn] [,Dn] [,Vn]	protects a file from being accidentally destroyed. A locked file cannot be renamed, deleted, or changed until it has been unlocked.
UNLOCK fn [,Sn] [,Dn] [,Vn]	removes protection from a locked file, making it possible to rename, delete, and change the file.
DELETE fn [,Sn] [,Dn] [,Vn]	removes a file from a disk. The file fn must be unlocked.
VERIFY fn [,Sn] [,Dn] [,Vn]	checks that a file is properly stored on the disk.

Program

FILEM	combines several useful tasks with DOS commands that handle files in a program that can work with groups of files. FILEM uses a wildcard and can operate on locked files.
-------	---

Using Programs Already on Disks

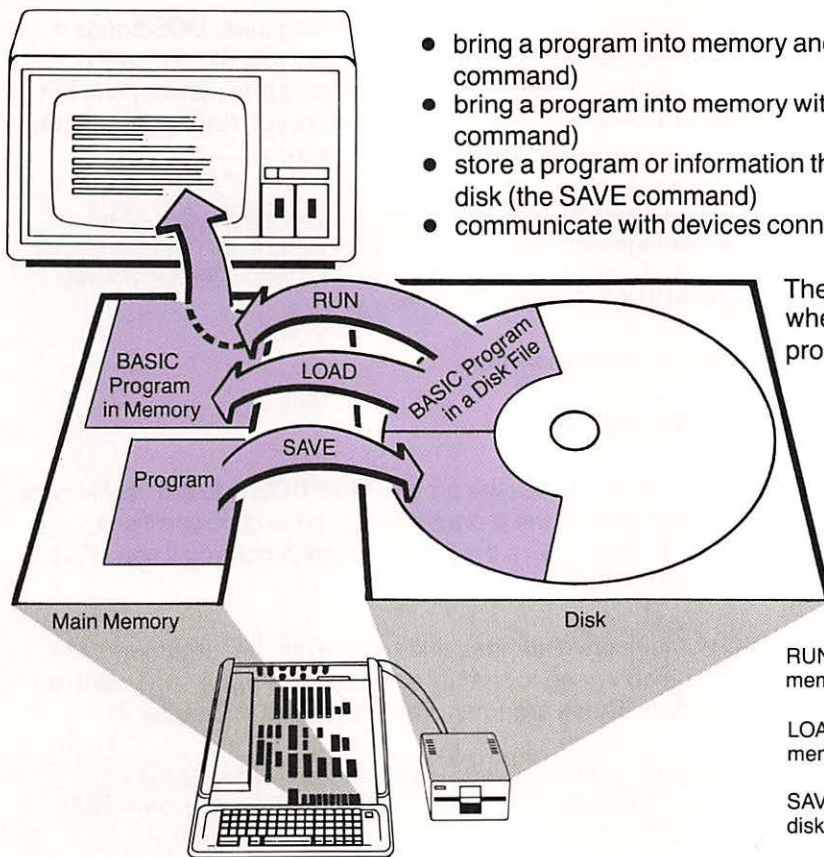
102	The RUN Command
103	Example
104	The LOAD Command
104	Example
105	The SAVE Command
106	Example
107	Talking to Other Devices
108	The PR# Command
109	Example
110	The IN# Command
110	Example
111	Ways to Restart DOS
111	Conclusion
112	Chapter 5 Summary



Using Programs Already on Disks

This chapter describes the DOS commands that let you use the BASIC programs on your disks. If all you want to do is run programs that are already on disks, pay special attention to the RUN command; it moves any BASIC program from a disk file into memory and starts it running.

Figure 5-1. Using DOS Commands With BASIC Programs



The commands discussed in this chapter let you

- bring a program into memory and start it running (the RUN command)
- bring a program into memory without running it (the LOAD command)
- store a program or information that is currently in memory on a disk (the SAVE command)
- communicate with devices connected to your computer

These commands are most relevant when you are running or modifying programs that already exist.

RUN copies a program from the disk into memory and executes it.

LOAD copies a program from a disk into memory.

SAVE copies a program in memory to a disk.

You use the **RUN** command to start a program running.

A **peripheral card** is a removable printed-circuit board that plugs into an expansion slot and expands or modifies the computer's capabilities.

An explanation of the syntax notation in this manual can be found in Chapter 2, "DOS Commands."

The RUN Command

The RUN command executes an Applesoft or Integer BASIC program that is stored on a disk. The command loads the program for you. Starting a program up this way is called *running the program*.

RUN combines several tasks into one command. When DOS sees the RUN command, first it finds the named program. Then, before bringing the program into memory, DOS checks the program's file type. If the language is not being used, DOS switches to the appropriate one, if possible.

When the file's type is A and Applesoft BASIC is not active, DOS switches to Applesoft. When the file's type is I and Integer BASIC is not active, DOS switches to Integer.

Note: DOS is unable to switch automatically if your Apple computer has only one language available. Apple II and Apple II Plus computers require a language card or some other **peripheral card** to use both Applesoft and Integer BASIC programs.

Once DOS has switched to the program's language, DOS brings a copy of the program into memory and starts it running. Since the RUN command automatically loads a program into memory, it is not necessary to use the LOAD command before you run a program (the LOAD command is explained in the next section).

By the Way: The RUN command erases main memory before it puts a copy of the program there. If you have something in memory that you don't want to lose, you must store it on a disk with the SAVE command before you issue a RUN command.

The command has this syntax:

RUN fn [,Sn] [,Dn] [,Vn]

The filename, fn, indicates the program that DOS is to run. Its file type must be either A or I. If the program is not on the disk, you'll see **FILE NOT FOUND**. If the file type is not A or I, you'll see **FILE TYPE MISMATCH**.

The slot number, drive number, and volume number arguments are optional. You only need to specify a number when you don't want to use the default. These arguments are described in Chapter 2.

By the Way: Once your program is in memory, you can run it again by issuing the RUN command without a filename. Without a filename, RUN is a BASIC command.

Example

The following example will show how the RUN command works with Integer BASIC and Applesoft BASIC programs.

If You Have Only One BASIC Language: If you try to run a program whose language you don't have in your computer, you'll see LANGUAGE NOT AVAILABLE. Try the RUN command on the program that has the file type for your BASIC.

1. Insert the SAMPLE PROGRAMS disk into drive 1. Use the CATALOG command to look at the disk's contents and to set the slot and drive number. Type

C A T A L O G SPACE D 1 , S 6

You'll see the programs APPLEVISION and COLOR TEST listed in the catalog.

2. Now try running the Integer BASIC program. Type

R U N SPACE A P P L E V I S I O N

and press RETURN. You'll see the most amazing display. When you want to stop the show, press ESC. You'll see Integer BASIC's prompt, >.

3. The Applesoft program is a spectacular demonstration for color screens. But even if you don't have color, you'll be able to watch a fascinating two-tone display. Type

R U N SPACE C O L O R SPACE T E S T

When you've had enough, press RETURN to stop the program and to return to the menu. Press 5 to exit from the COLOR TEST program. You'll see Applesoft's prompt, 1.

Language Switching: If your Apple computer has both Applesoft and Integer BASIC available, DOS changed languages automatically; you didn't have to issue any special command.

The **LOAD** command puts a copy of a program on a disk into memory.

The **LOAD** Command

The **LOAD** command transfers a program from a disk into memory. Before bringing the program into memory, DOS checks the program's file type and tries to switch to the appropriate BASIC. When you use the **LOAD** command to bring a copy of the contents of a file into memory, the file on the disk remains unchanged.

It is not necessary to load a program before you use the **RUN** command; **RUN** automatically loads the program into memory.

By the Way: The **LOAD** command erases main memory before it puts a copy of the program there. If you have something in memory that you don't want to lose, you must store it on a disk with the **SAVE** command before you issue a **LOAD** command.

The command has this syntax:

LOAD fn [,Sn] [,Dn] [,Vn]

The filename, fn, indicates the program that DOS is to load. Its file type must be either A or I. If the program is not on the disk, you'll see **FILE NOT FOUND**. If the file type is not A or I, you'll see **FILE TYPE MISMATCH**.

The slot number, drive number, and volume number arguments are optional. You only need to specify a number when you don't want to use the default. These arguments are described in Chapter 2.

Example

1. With the **SAMPLE PROGRAMS** disk in drive 1, bring a copy of the **GET TEXT** program into memory by typing

```
LOAD SPACEGET SPACETEXT
```

You'll hear the disk drive working. Then you'll see BASIC's prompt again.

2. To look at the program, use the **LIST** command. Type

```
LIST
```

You'll be looking at the BASIC statements that make **GET TEXT** work. If you're curious about the programming involved, this program is discussed in the *DOS Programmer's Manual*.

When you load a program, DOS transfers a copy from the disk to the computer's memory. The original version remains unchanged on the disk. You can use the **CATALOG** command to see that a file is still on the disk after you load a copy of its contents into memory.

LIST is a BASIC command.

The **SAVE** command puts a copy of the program in memory onto a disk.

The **SAVE** Command

The **SAVE** command transfers the BASIC program that is currently in memory to a file on a disk.

DOS keeps track of the language you are in and writes the program to the disk with the proper file type. When you are using Applesoft BASIC, DOS saves the program with file type A. When you are using Integer BASIC, DOS saves the program with file type I.

DOS also determines the length of the program and where to put it on the disk.

The **LOCK** and **UNLOCK** commands are discussed in Chapter 4.

Once you have saved a program, you can use the **LOCK** command to protect it from accidental erasure. The **UNLOCK** command removes that protection.

You can use the **SAVE** command to copy files and to make backup copies of a program. To do this, use the **LOAD** command to bring a copy of the file's contents into memory, then use the **SAVE** command to save the program to a different disk or on the same disk with a different name.

The command has this syntax:

SAVE fn [,Sn] [,Dn] [,Vn]

The **fn** argument specifies the filename for the program. If a disk file by that name already exists, DOS will copy the program in memory over the program on the disk if the file is unlocked and it is the same type (A or I).

You'll see **I/O ERROR** when you use the **SAVE** command if the specified disk is bad, if the disk is not initialized, if there is no disk in the disk drive, or if the drive door is open. Put an initialized disk in the disk drive, close the door properly, and issue the **SAVE** command again.

The slot number, drive number, and volume number arguments are optional. You only need to specify a number when you don't want to use the default. These arguments are described in Chapter 2.

Warning

If you save a program in memory to a file on a disk that already has a file with the same name and file type, the contents of the original disk file are lost. DOS does not display any warning.

NEW is a BASIC command that erases memory.

Example

1. Put an initialized disk into disk drive 1. Be sure that it is not write-protected—that the write-enable notch is uncovered.
2. Issue a NEW command to clear memory for a new program. Type

NEW

and press **RETURN**.

3. Now, type a short program, such as:

```
10 PRINT "A SHORT PROGRAM"  
20 END
```

Even if you aren't too familiar with writing programs, you might guess that this program does only one thing: it prints A SHORT PROGRAM on the display.

Without a filename, RUN is a BASIC command.

4. To test your program, use the BASIC command RUN. Type

RUN

and press **RETURN**.

5. Then, to save the program on your disk, use the SAVE command with a descriptive filename. For instance, type

SAVE SPACE ONE SPACE LINER

and press **RETURN**.

The disk drive whirs as the SAVE command puts your program on the disk and names it ONE LINER.

Check It Out: After you use the SAVE command, you can see that a program has found its new home on the disk by using the CATALOG command. Notice that a copy of a program remains in memory (it will stay there until you issue a NEW command or turn off the computer). You can use the RUN or LIST commands to check that the program is still in memory.

Talking to Other Devices

An **expansion slot** is a connector inside the computer in which a peripheral card can be installed. A **peripheral card** is a removable printed-circuit board that plugs into one of the expansion slots. It is used to connect a peripheral device.

A **peripheral device** is used in conjunction with the computer. It is usually connected with a peripheral card and cable. A video monitor, disk drive, printer, and modem are all examples of peripheral devices.

I/O refers to the transfer of information into and out of a computer.

A **controller card** is a peripheral card that connects a device, such as a printer or disk drive, to the computer and controls the operation of the device.

The commands **PR#** and **IN#** enable your Apple computer to communicate with devices like printers that are connected through **expansion slots**.

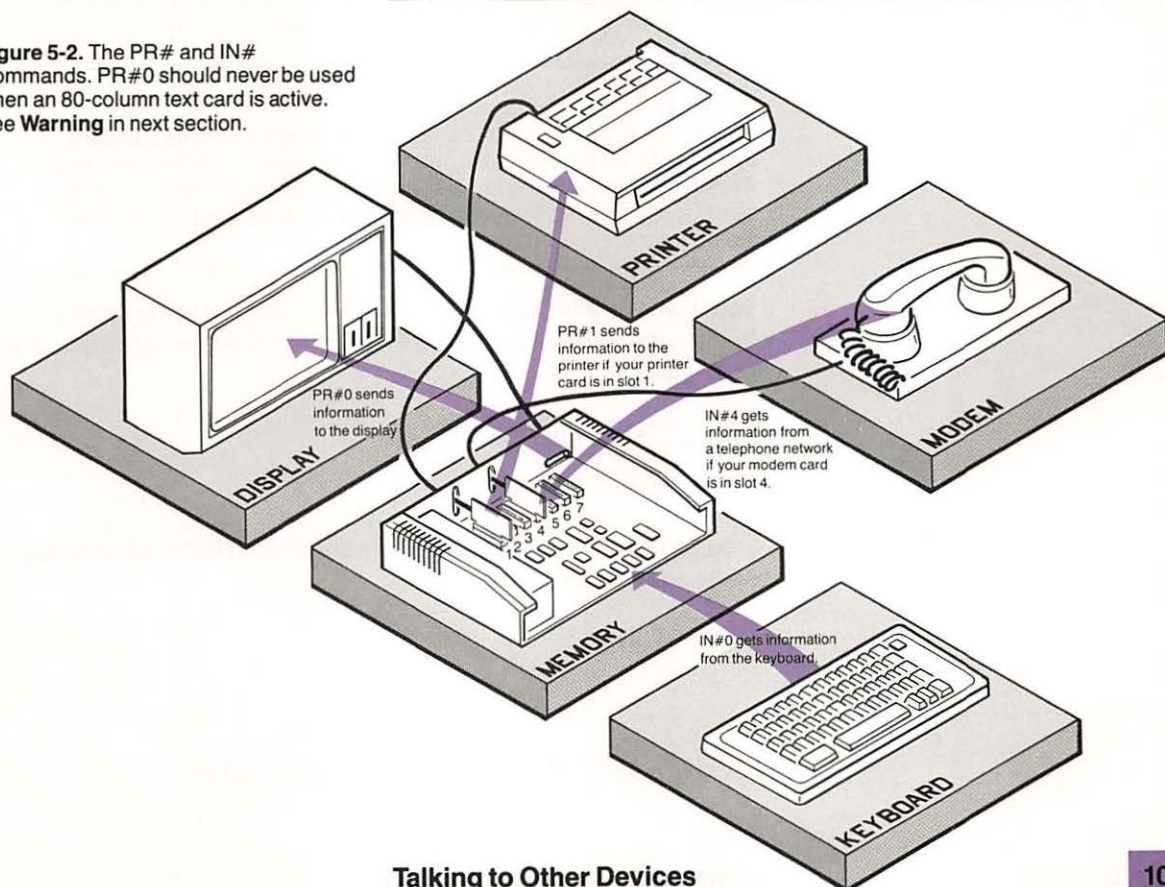
Your Apple computer usually sends characters to the screen and reads characters from the keyboard. The **PR#** and **IN#** commands allow your computer to talk to additional devices, called **peripheral devices**. Such communication is referred to as **I/O**, the input and the output of information.

In the **PR#** and **IN#** commands, you must type the **#** character.

Warning

If you do not have a **controller card** in the slot that you have specified with the **PR#** or **IN#** commands, random events may occur. For example, if your controller card is in slot 7 and you issue a **PR#6** command, the disk will boot on some systems and not on others. You won't see an error message, but DOS may look like it's disconnected. You have to restart DOS if it doesn't restart itself. See the section at the end of this chapter, "Ways to Restart DOS."

Figure 5-2. The **PR#** and **IN#** Commands. **PR#0** should never be used when an 80-column text card is active. See **Warning** in next section.



The **PR#** command directs information to any device.

The PR# Command

The **PR#** command tells your computer where to send information; that is, it specifies a device connected through one of the computer's slots. The **PR#** command will send *all* information—including prompts, error messages, commands, program listings, and catalogs—to the device specified.

A Bit of Speculation: Back in the dark ages of computers, a printer was the one and only output device. The name of the **PR#** command probably is short for "printer." In addition, printers used to be assigned numbers, which may account for the number symbol (#) in this command.

The command takes this form:

PR# n

The number, n, is the slot that connects the device to which you want to send information. The number can be between 0 and 7.

When you want to send information to the display again, use **PR#0**, unless you are using an 80-column text card. This command works on the Apple IIe computer even though the IIe does not have a slot 0.

Warning

When your Apple IIe computer is operating the 80-column text card, using the **PR#0** command is *not* recommended. Use **PR#3** (instead of **PR#0**) to deactivate the printer and reconnect the 80-column text card. Use **ESC CONTROL-Q** to deactivate both the 80-column text card and the printer.

If you have a printer, you can send a disk's catalog to the printer. When your printer is connected to a printer controller card installed in slot 1, type

P R SHIFT # 1 RETURN

C A T A L O G RETURN

When you are through printing and want to send information to the display again, type

P R SHIFT - # 0

or

P R SHIFT - # 3 if your 80-column text card is active.

Starting Up a Disk Using the PR# Command: Even though the PR# command is used primarily to direct outgoing information, you can also use PR# to start disk drives. Using the command this way is easier on the equipment than turning the power off and then on again.

Note: If you have an Apple II with the old Monitor ROM, you must start DOS with **(S) (CONTROL)-(P)** and be in BASIC before you can use the PR# command to start disks.

Example

1. Put the SAMPLE PROGRAMS disk in drive 1, close the door, and type

(P)(R)(SHIFT)-#6

and press **(RETURN)**. If your disk controller card is connected via another slot, use that number. Drive 1 will whirl as DOS starts up the disk drive.

2. If you have a printer connected via slot 1, try printing a program listing. First read HELLO into memory by typing

(L)(O)(A)(D)(SPACE)(H)(E)(L)(L)(O)

3. Then, to direct the computer's output to the printer, type

(P)(R)(SHIFT)-#1

and press **(RETURN)**. If your printer controller card is in another slot, specify the number of that slot.

4. Now, to print a listing of HELLO, type

(L)(I)(S)(T)

and press **(RETURN)**. Your printer will print a listing of the HELLO program.

5. Try issuing other commands. Everything that is normally printed on the display goes to the printer, including prompts and error messages.
6. To send information back to the display screen, use the PR# command with slot number 0.

The **IN#** command tells your computer where to get information

The IN# Command

The **IN#** command tells your computer where to get information; that is, it specifies a device connected through one of the computer's slots. **IN** stands for "input."

The command takes this form:

IN# n

The number, *n*, is the slot of the device from which to get information. The number can be from 0 to 7. When you use the **IN#** command, DOS will get information, or *read*, from the device connected through the specified slot.

Use **IN#0** to read information from the keyboard again.

This command works the same on the Apple IIe computer even though the IIe does not have a slot 0.

Example

1. If your Apple computer has a telephone modem that connects to a telephone network and the modem is connected to slot 4, typing

IN **SHIFT**-**#** **4**

causes DOS to read information from the network.

2. To read information from the keyboard again, use the **IN#** command with slot number 0.

Incidentally: You can start a disk in drive 1 by using the **IN#** command with the slot number of the disk controller card—just as you did with **PR#** command.

Ways to Restart DOS

There will be times when you will want to restart DOS. For instance, if your keyboard quits responding or the cursor disappears—referred to as a “hung” system—it is necessary to restart DOS. You may also want to restart DOS to make sure a new disk will start the system.

When you have started up your computer with DOS, there are several methods you can use to restart DOS. They are listed in increasing order of severity.

1. Type **CONTROL-RESET**. This method does not affect the contents of main memory. This is the first method to try if your system hangs.
2. Type **CALL SPACE 1002**. This is a BASIC command that reconnects DOS but does not affect the contents of main memory.
3. Type **PR SHIFT-#6** to restart DOS from drive 1, slot 6 on all models of the Apple II or type **3-CONTROL-RESET** on an Apple IIe. These two methods destroy whatever is in main memory. This is the method you should use to test a new system disk.

If you see the Monitor program's asterisk (*), type

6 CONTROL-P, or

6 CONTROL-K to start DOS.

Or type

3 D O G (that's a zero), or

C 6 0 0 G (these are zeros, too) to restart DOS.

If your disk controller card is installed in a slot other than slot 6, replace 6 in these commands with the number of that slot.

Conclusion

This is the end of the introduction to DOS. With what you have read so far, you now know enough to deal with reasonably complex disk operations. Once you are comfortable with the fundamental commands and concepts, you may want to try the advanced ones presented in the *DOS Programmer's Manual*.

Chapter 5 Summary

RUN fn [,Sn] [,Dn] [,Vn]	copies a program (file type A or I) from a disk file into memory and then executes the program.
LOAD fn [,Sn] [,Dn] [,Vn]	copies a program (file type A or I) from a disk file into memory. Once the program is in memory, you can run it, modify it, or save it.
SAVE fn [,Sn] [,Dn] [,Vn]	writes a copy of the BASIC program currently in memory to a disk file. An Applesoft program is saved as type A. An Integer BASIC program is saved as type I.
PR# n	specifies where your computer should send information; that is, it designates a device, connected through one of the computer's slots, for output.
IN# n	specifies where your computer should get information; that is, it designates a device, connected through one of the computer's slots, for input.

Appendixes

117	Appendix A: Dealing With 13-Sector Disks
118	Converting a 13-Sector Disk: the CONVERT13 Program
118	Example
122	Duplicate Filenames
122	The Wildcard Character
123	Running 13-Sector Disks Without Conversion
123	Using the START13 Program
124	Example
124	Using the BASICS Disk
127	Appendix B: Warnings and Error Messages
128	DOS Messages
128	DISK FULL
128	END OF DATA
129	FILE LOCKED
129	FILE NOT FOUND
130	FILE TYPE MISMATCH
130	I/O ERROR
131	LANGUAGE NOT AVAILABLE
132	NO BUFFERS AVAILABLE
132	NOT DIRECT COMMAND
133	PROGRAM TOO LARGE
133	RANGE ERROR
134	SYNTAX ERROR
134	VOLUME MISMATCH
134	WRITE PROTECTED
135	Stopping a Program

137	Appendix C: Programs on the DOS Disks
137	Programs on the SYSTEM MASTER Disk
139	Programs on the SAMPLE PROGRAMS Disk
143	Appendix D: Summary of DOS Operating Concepts and Commands
143	Operating Concepts
143	Starting Up
144	Restarting DOS
144	Capacity
145	Notation
145	Syntax
146	Arguments
147	Command Summary
147	CATALOG
147	DELETE
148	IN#
148	INIT
149	LOAD
149	LOCK
150	PR#
150	RENAME
150	RUN
151	SAVE
151	UNLOCK
151	VERIFY

Dealing With 13-Sector Disks

An **application program** accomplishes some specific purpose or task, such as word processing or data-base management.

A **sector** is a fixed fraction of a track. When you initialize a disk with DOS 3.3, DOS prepares the disk for information by dividing its surface into 35 tracks, each containing 16 sectors.

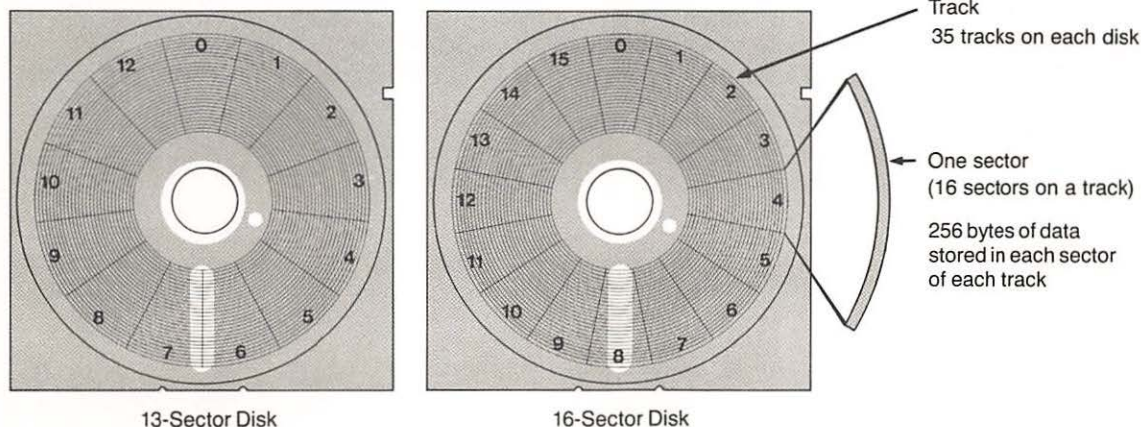
This appendix is for people who have games or other **application programs** that don't run on their present system. If the disk drive keeps whirring when you attempt to start up a disk, and you have already checked to see that the disk is seated properly, you might have a 13-sector disk.

Earlier versions of DOS (3.1 and 3.2) organized the surface of each track on a disk into 13 **sectors**. The version you have (3.3) uses a format that divides each track into 16 sectors (see Figure A-1). Changing from 13 sectors to 16 sectors gives you about 20 percent more space for information on each disk.

However, 13-sector disks cannot be used with 16-sector DOS until they are converted. This appendix tells you how to:

- convert 13-sector disks with the CONVERT13 program
- use 13-sector disks with the START13 program
- use the BASICS disk if you can't convert 13-sector disks because they're copy-protected

Figure A-1. 13-Sector Versus 16-Sector Disks



Converting a 13-Sector Disk: the CONVERT13 Program

The CONVERT13 program on the SYSTEM MASTER disk converts a 13-sector disk and its programs to a 16-sector format. CONVERT13 copies the contents of your 13-sector disk to an initialized 16-sector disk.

CONVERT13 reads the information from a 13-sector disk and writes the information to a newly prepared 16-sector disk. The information on the 13-sector source disk is not changed in any way.

Incidentally: You may be familiar with the MUFFIN program from a previous version of DOS. MUFFIN performed the same function as CONVERT13 does. In fact, CONVERT13 runs MUFFIN for you.

Example

This example uses CONVERT13 to transfer information on a 13-sector disk to a 16-sector disk. The 16-sector disk must already be **initialized**.

If your 13-sector disk is protected against copying (as are some commercially available disks), you won't be able to convert it, but you can still run the programs on it by using START13 and the BASICS disk (see the section "Running 13-Sector Disks Without Conversion.")

This example shows you how to use CONVERT13 with *one disk drive*. You may need to swap disks several times to convert an entire disk.

Two Drives: When you are converting files and you have more than one disk drive, place the 13-sector disk in drive 1 and the newly prepared destination disk in drive 2 before conversion begins. When CONVERT13 asks for slot and drive numbers, specify 6 for both slot numbers (if that's the disk controller card's slot) and 1 and 2, respectively, for the source and destination disk drive numbers.

If you need to know how to **initialize** a disk, see "Preparing a Blank Disk: the INIT Command," Chapter 1.

Figure A-2. The CONVERT13 Menu. You see MUFFIN VERSION D because CONVERT13 runs the binary MUFFIN program.

Cursor

What You Do

1. Put the SYSTEM MASTER disk in drive 1 and type

`R U N SPACE C O N V E R T 1 3`

and press `RETURN`.

What You Get

You'll see EXECUTING MUFFIN briefly, then you'll see CONVERT13's menu (Figure A-2).

```
*****
*  APPLE II DOS 3.2 TO 3.3 CONVERTER  *
*                                     *
*          MUFFIN VERSION D          *
*                                     *
*  COPYRIGHT APPLE COMPUTER, INC. 1979 *
*****

CHOOSE ONE OF THE FOLLOWING OPTIONS

      <1> CONVERT FILES
      <2> QUIT

WHICH WOULD YOU LIKE? 
```

2. To convert files, type

`1`

and press `RETURN`.

The program starts up, and you'll see SOURCE SLOT? The program wants to know where to find the source (13-sector) disk. First it asks for the **expansion slot** number and then the disk drive number.

An **expansion slot** is a long connector inside the computer for connecting a peripheral card.

A **peripheral card** is a removable printed-circuit board that expands or modifies the computer's capabilities. A disk controller card is a peripheral card.

A **disk controller card** is a peripheral card that connects one or two disk drives to the computer and controls their operation.

3. To tell the program to look in the disk drive connected through slot 6, type

6

and press **(RETURN)**. If your **disk controller card** is in a different slot, type the number of that slot.

The next prompt will appear on the display, **DRIVE?**

4. To indicate the number of your disk drive that will contain the 13-sector disk, type

1

for drive 1 and press **(RETURN)**.

Then you'll see **DESTINATION SLOT?** as the program asks where to put the converted files. The destination disk is the 16-sector disk.

5. Type

6

for slot 6 and press **(RETURN)**. If your disk controller card is in a different slot, type the number of that slot.

You'll see **DRIVE?**

6. Type

1

for drive 1 and press **(RETURN)**. You have now told **CONVERT13** to use drive 1 for both the source and destination disks.

Now the program asks for the name of the file with **FILENAME?**

7. Type

=

and press **(RETURN)**. The equal sign is a character that stands for all the files on the disk.

You'll see **DO YOU WANT PROMPTING?** The program is giving you the chance to double-check that you want each file converted before **CONVERT13** starts the process.

If you do not want to convert all the files on the 13-sector disk, type **(Y)** instead. Then when each file's name is displayed, you must tell the program whether to convert the file or not by typing **(Y)** or **(N)**. (Or **(Q)** if you want to end the conversions and return to the menu.)

8. Type

(N)

for "no" prompting and press **(RETURN)**. All the files will be converted automatically.

You'll see **INSERT DISK(S), THEN PRESS <ESC> TO RETURN TO MAIN MENU OR ANY OTHER KEY TO BEGIN.**

This message gives you a chance to change your mind about converting the disk. If you wanted to do this (don't try it now), you'd simply press **(ESC)**.

9. Remove the SYSTEM MASTER disk from drive 1.

From here on, you'll be working with two disks: your 13-sector disk and your newly prepared 16-sector disk.

10. Put the 13-sector disk into the disk drive. Press **(RETURN)** to begin.

CONVERT13 finds the first file on your 13-sector (source) disk and displays its name: **FILE HELLO**. Then you'll see **INSERT DESTINATION DISK AND PRESS ANY KEY**.

11. Insert the 16-sector disk that you prepared with the **INIT** command; press **(RETURN)**.

CONVERT13 puts a 16-sector version of the file on the new disk and reports **DONE**.

12. If there are more files on the source disk, **CONVERT13** instructs you to insert the source disk again.

Repeat steps 10 and 11 as the program alternately reads from the source disk and writes to the destination disk until each 13-sector file has been converted and moved to the new disk.

Eventually, you'll see **PRESS ANY KEY TO CONTINUE**. This means **CONVERT13** is done converting all your files.

13. Press **RETURN**.

You'll see CONVERT13's menu again.

14. To return to the prompt, type

2

to "quit" and press **RETURN**.

You can now use the 16-sector disk, with the converted 13-sector files, as described in the DOS 3.3 manuals.

Duplicate Filenames

If you try to convert a 13-sector file with the same name as a file already on the destination disk, you'll see:

```
FILE [filename]  
ALREADY EXISTS.  
TYPE IN A NEW FILE NAME FOR THE COPY  
OR <RETURN> TO REPLACE EXISTING FILE  
OR <CONTROL-C><RETURN> TO CANCEL COPY  
:
```

Once you have received this message, you may: type a new name for the file to be converted; press **RETURN** and have the converted 13-sector file replace the file currently on the 16-sector disk; or type **CONTROL-C** and **RETURN** to stop the conversion.

The Wildcard Character

As you saw in the CONVERT13 example, you may use the **wildcard** character, an equal sign, to mean all files on the disk. You may also use it to stand for any character or group of characters within a filename. For example, if you respond to **FILENAME?** by typing **F=LE**, CONVERT13 converts all files whose names begin with **F** and end with **LE**.

When you use the wildcard, CONVERT13 asks **DO YOU WANT PROMPTING?** To have CONVERT13 stop after finding each file on the 13-sector disk and ask you to confirm that you want to convert it, type **Y** and press **RETURN**. To convert all the files in the wildcard group without checking each one, type **N** and press **RETURN**. Once the prompting has begun, if you want to end the conversions and return to the menu, type **Q** and press **RETURN**.

The **wildcard** can stand for any character or sequence of characters in a filename. See Chapter 4 for more information.

Running 13-Sector Disks Without Conversion

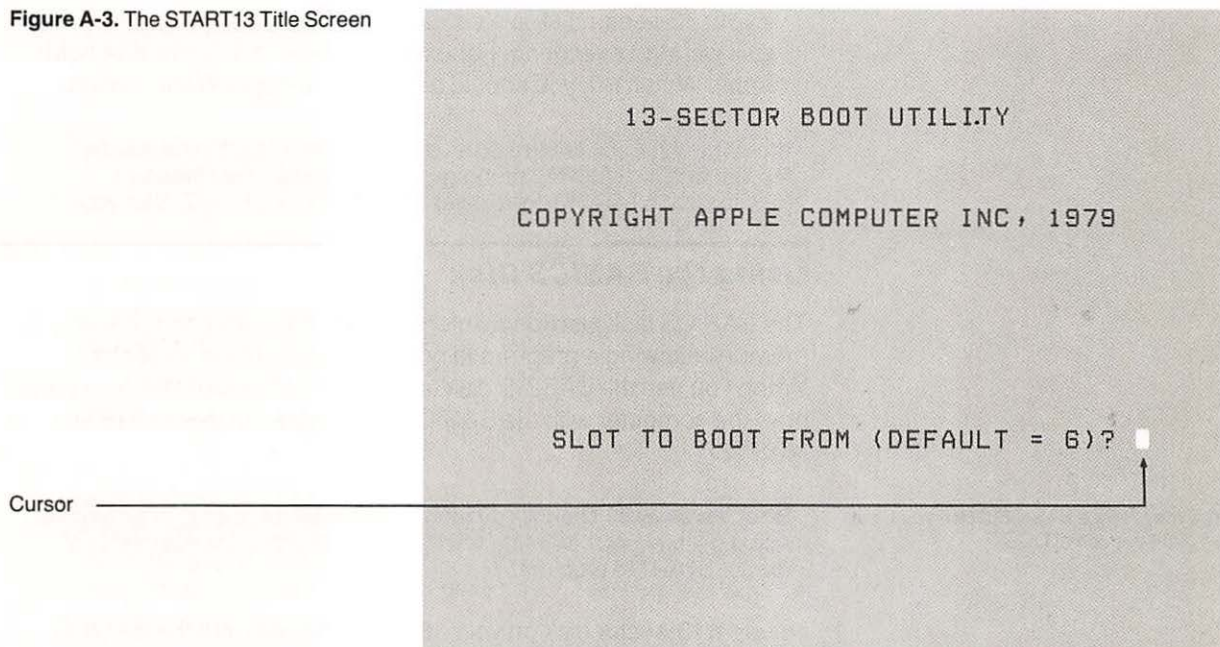
Both the START13 program and the BASICS disk can be used to run 13-sector disks on your 16-sector system. The START13 program and the BASICS disk are designed to allow you to use a 13-sector disk when that disk is protected against copying.

The BASICS disk is no longer supplied with DOS because the START13 program is easier to use. Programs you have purchased may tell you to "boot the BASICS disk." You can use START13 instead.

Using the START13 Program

The START13 program loads information from a 13-sector disk into memory. Essentially, START13 is a special loader that allows your system to start up 13-sector disks.

Figure A-3. The START13 Title Screen



Example

This example shows how to use START13 to read information from a 13-sector disk.

1. Put the SYSTEM MASTER disk in drive 1 and execute the START13 program by typing

`R U N SPACE S T A R T 1 3`

Press `RETURN`. The message EXECUTING BOOT13 appears briefly. Then you'll see the title screen illustrated in Figure A-3.

2. When you see 13-SECTOR BOOT UTILITY, followed by SLOT TO BOOT FROM (DEFAULT=6)?, remove the SYSTEM MASTER and insert your 13-sector disk in drive 1.
3. Press `RETURN` if your disk controller card is connected through slot 6. If your card is in another slot, enter the number of that slot before you press `RETURN`.

If your 13-sector disk is a startup disk, the startup program on that disk will start executing. For example, if your 13-sector disk holds Apple Writer 1.0, you should be looking at Apple Writer's menu.

Incidentally: If you have used an earlier version of DOS, you may be familiar with the BOOT13 program. BOOT13 performed the same function that START13 does. In fact, START13 runs BOOT13 for you.

Using the BASICS Disk

The BASICS disk also loads information from a 13-sector disk into memory, allowing a program to both read and write in 13 sectors. When you use the BASICS disk to work with 13-sector files, you must start the computer with the BASICS disk instead of the SYSTEM MASTER.

You may have a BASICS disk if you have an older version of DOS.

Boot the BASICS Disk: If you read "boot the BASICS disk" in some older Apple manual, you can either start the BASICS disk (described next) or run the START13 program.

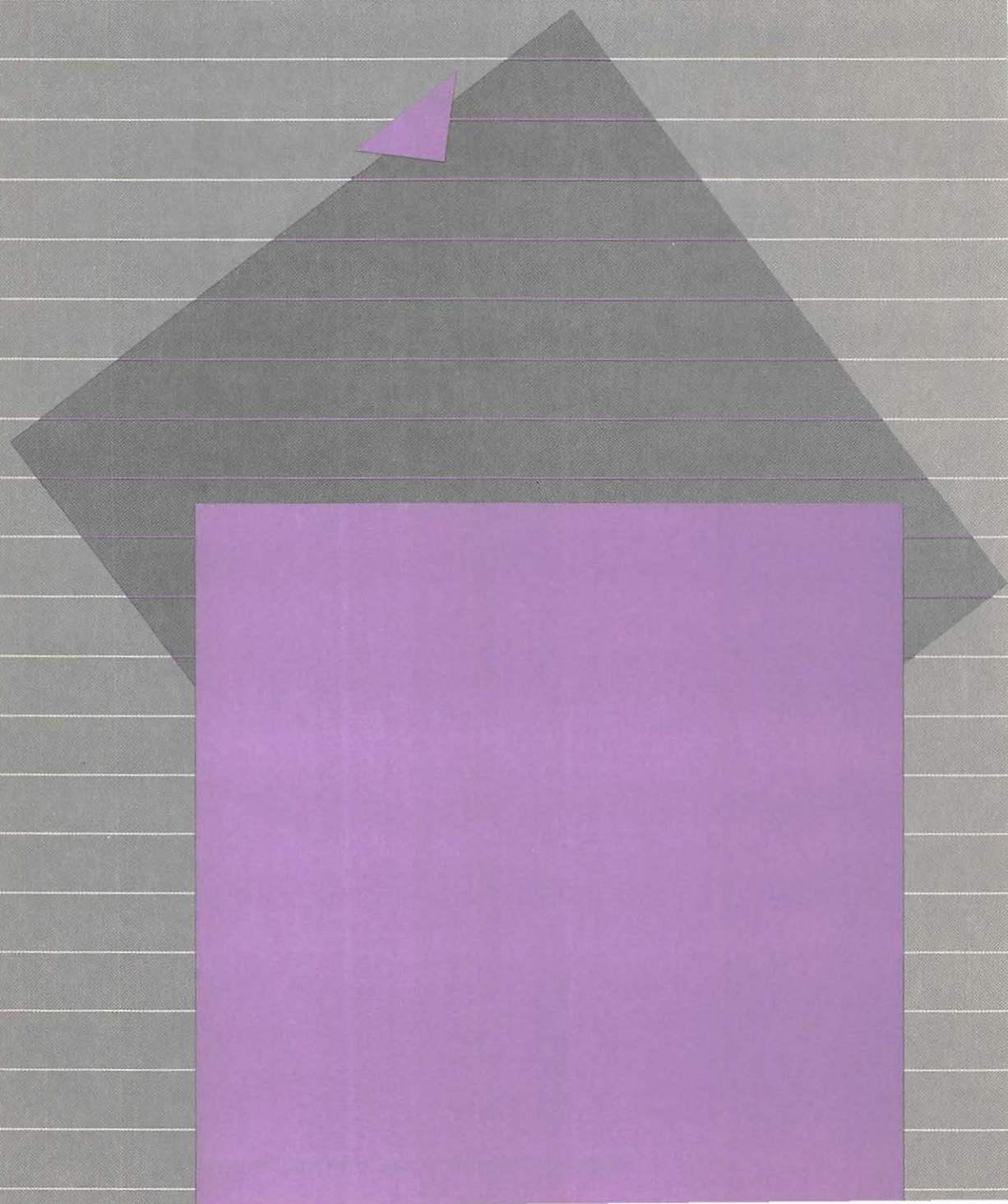
To use a 13-sector disk on your 16-sector system, put the BASICS disk in drive 1 (connected via slot 6) and turn the computer on. When you see this on your display

INSERT YOUR 13-SECTOR DISKETTE
AND PRESS RETURN

insert any 13-sector disk and press `RETURN`.

Note that the BASICS disk itself does not contain DOS; trying to use DOS when you have started up the computer with the BASICS disk will produce garbage on your display.

If a program automatically runs, it means that your 13-sector disk contains a turnkey program, and you're all set. If only a cursor appears on your display, it means that your 13-sector disk does not contain a turnkey program; type CATALOG to see a list of the files on the disk, and use the RUN command to execute the program you want.



Warnings and Error Messages

An error message alerts you to a problem and indicates that a command you have issued is incompatible with DOS's rules about that command. The form of an error message tells you whether it comes from DOS or from one of the BASIC languages.

Form	Message Sent By:
SYNTAX ERROR	DOS
?SYNTAX ERROR	Applesoft BASIC
*** SYNTAX ERR	Integer BASIC

This appendix discusses only DOS errors. Applesoft and Integer BASIC error messages are documented in their respective manuals.

This appendix explains the possible causes of each error message and suggests what to do to correct the problem.

DOS Messages

DOS messages are summarized in Table B-1. A discussion of each error message, including a description of causes and cures, follows the table. The error messages are discussed in alphabetic order. The error messages are discussed in alphabetic order.

Table B-1. DOS Error Messages

Message	Most Common Cause
DISK FULL	Too many files in the catalog or all sectors on disk used
END OF DATA	Reading beyond end of text file
FILE LOCKED	Attempt to write to, delete, or rename a locked file
FILE NOT FOUND	Filename misspelled or file not on disk
FILE TYPE MISMATCH	Command doesn't match file type
I/O ERROR	Door open, disk not initialized, bad disk drive, or bad disk
LANGUAGE NOT AVAILABLE	Language required is not present in computer
NO BUFFERS AVAILABLE	Too many text files are open
NOT DIRECT COMMAND	Command must be in a program
PROGRAM TOO LARGE	Insufficient memory available
RANGE ERROR	Command value is too large or too small
SYNTAX ERROR	Bad filename or value
VOLUME MISMATCH	Wrong volume number
WRITE PROTECTED	Write-enable notch not present or covered by write-protect tab

DISK FULL

DISK FULL means that DOS tried to store information on a disk when no more space was available. In this situation, DOS closes all files and saves all the information that it can.

When you get a **DISK FULL** message, you can delete a file or two on the current disk before trying to save the information in memory. Or you can change disks and save the information in memory on a disk that has more room.

If you receive this message and try again to save a file on the full disk, the sector length of one of the existing files in that disk's catalog will be set to zero. Despite the odd appearance of that catalog entry, the existing file itself will not be damaged in any way.

END OF DATA

END OF DATA means that your program tried to retrieve information from an area of the text file where there is no information.

Any byte beyond the last field in a sequential-access text file or beyond the last field of each record in a random-access text file may

ASCII stands for the "American Standard Code for Information Interchange."

contain the value 0. The zero character is the **ASCII** code for a null character. Any text file command that tries to read this character produces the **END OF DATA** message.

You may also see **END OF DATA** after an **INPUT** or a **GET** command if they were used incorrectly in your program.

To keep any data from being lost when you see the **END OF DATA** message, issue the **CLOSE** command. See the *DOS Programmer's Manual* for more information on how to fix your program.

FILE LOCKED

FILE LOCKED means you tried to save, write to, change, append, delete, or rename a file that is locked. Check the catalog: the name of a locked file is preceded by an asterisk (*). To unlock the file, use the **UNLOCK** command.

FILE NOT FOUND

FILE NOT FOUND means that you specified the name of a file that is not on the disk that you are currently using.

These are the most likely causes:

- You may have misspelled the filename by typing the filename incorrectly or by omitting the comma that separates the filename from another argument. Check your typing carefully. Check the catalog for the exact spelling of the filename.
- The file is not on the disk you specified. Check the catalog.
- The file has been accidentally deleted from the disk. Check the catalog.
- If you see **FILE NOT FOUND** each time you start a disk, you must tell DOS the name of a greeting program on that disk. If you have no files on the disk that you want to keep, you can initialize the disk again using the **INIT** command—remember, the **INIT** command erases anything that was on the disk. If you cannot remember the name of the greeting program that is already on the disk, run the **MASTER** program to rename the greeting program.

The **MASTER** program is explained in the *DOS Programmer's Manual*.

FILE TYPE MISMATCH

FILE TYPE MISMATCH means that a DOS command specified a filename that is already assigned to a file whose file type is inappropriate to the present command.

Use the **CATALOG** command to check the type of the file. Then look at Table B-2 to make sure that the command you are using is legal with that file type.

If you are sure that the command is correct, use a filename that is not on the disk, use a different disk, rename the existing file, or delete the existing file.

Table B-2. Types of Files According to Command

Command	Legal File Type
LOAD, RUN, SAVE	Applesoft or Integer BASIC
CHAIN	Integer BASIC
OPEN, READ, WRITE, APPEND, POSITION, EXEC	Text
BLOAD, BRUN, BSAVE	Binary

I/O ERROR

I/O ERROR means that DOS is unable to store information on a disk or to retrieve information from a disk. DOS tries 96 times before it gives you this message.

This message can occur because:

- The door to the disk drive is open. Close it.
- No disk is in the selected or default disk drive. Put a disk into the drive and close the drive door.
- The disk in the drive has not been initialized. Prepare the disk using the **INIT** command.
- The disk is inserted incorrectly. Gently remove the disk from the disk drive and insert it again.
- The **VERIFY** command is indicating that a file is not stored correctly on the disk.
- A 16-sector version of DOS is trying to use a 13-sector disk. See Appendix A.

The **INIT** command is discussed in Chapter 3, "Preparing Disks."

Restart procedures are explained in Chapter 5, "Ways to Restart DOS."

- The drive number (Dn) argument specified a disk drive that does not exist in your system. A non-existent drive is now the default drive. Specify the correct drive number with the next DOS command.
- The slot number (Sn) argument specified a slot that does not contain a disk controller card. An erroneous slot is now the default slot, and DOS assumes that the disk that isn't connected to the slot is still running. Even if the next DOS command specifies the right slot, DOS waits in limbo forever for the non-existent disk to respond to the last command.

If you have no program in memory that you want to save, simply restart DOS. To recover with your program intact, do this:

1. Press **[RESET]**.
2. Type **CATALOG SPACE S_n**, where *n* is the correct slot number.
If this leads to a **SYNTAX ERROR**, it means that DOS has been disconnected. To reconnect DOS, type
CALL SPACE 1002
if you have a BASIC prompt, or type
3D0G (that's a zero)
if you have the Monitor's asterisk.
3. Then issue the catalog command again—**CATALOG, S_n**—and everything should be fine.

In any case, issue a **VERIFY** command to make sure the file you are working with is stored correctly on the disk. If the information is not stored correctly and is still in memory, try storing it again.

LANGUAGE NOT AVAILABLE

You will see **LANGUAGE NOT AVAILABLE** when DOS cannot find Integer or Applesoft BASIC. The commands **FP**, **INT**, **LOAD**, or **RUN** all initiate a language search.

You will see this message after you issue a command that requests Applesoft from the disk in the current drive and that disk does not contain the **APPLESOFT** or **FPBASIC** programs. Replace the disk with a disk that contains these programs (for example, the **SYSTEM MASTER** disk) and issue the command again.

The contents of **read-only memory**, or **ROM**, can only be read. Information is written into ROM once, during manufacture, and it stays there permanently, even when the computer is turned off.

The contents of **random-access memory**, or **RAM**, can be read from or written to. When you turn off the computer, the contents of RAM are erased.

You also will see this message after you issue a command that requests Integer BASIC and Integer BASIC is not in your computer. As long as your computer has enough memory to use both Applesoft and Integer, you can load Integer BASIC into memory from the SYSTEM MASTER disk.

When you request Applesoft, DOS looks for the language in ROM, **read-only memory**, or on a language card. If Applesoft is not there, DOS looks in RAM, **random-access memory**. When Applesoft is not in RAM, DOS looks on the disk in the current disk drive (the most recent values of the slot and drive number arguments).

When you request Integer BASIC, DOS looks for that language in ROM or RAM.

If you started DOS from the SYSTEM MASTER and get the LANGUAGE NOT AVAILABLE message, it means that you don't have enough space in your computer (64K or more) for both languages.

NO BUFFERS AVAILABLE

Each open file and each DOS command (except PR#, IN#, and MAXFILES) requires a file buffer. NO BUFFERS AVAILABLE means that you or your program tried to open one more file or issue one more DOS command than there were buffers available in memory.

Issue the CLOSE command to release the file buffers, or issue the MAXFILES command to increase the number of file buffers. To find out how to increase or decrease the number of file buffers, see the MAXFILES command in the *DOS Programmer's Manual*.

NOT DIRECT COMMAND

NOT DIRECT COMMAND means that you tried to use an APPEND, OPEN, POSITION, READ, or WRITE command in immediate execution. These commands can be used only within PRINT statements in a program; they are described in the *DOS Programmer's Manual*.

This message also can occur when a program is stopped and restarted. Using a POKE command will repair the error. Type

POKE SPACE 51,0:CONT

The **FP** and **INT** commands are used to switch between the two BASIC languages. See your BASIC manual.

PROGRAM TOO LARGE

PROGRAM TOO LARGE means that a DOS command tried to load a disk file and found insufficient space in main memory for the contents of that file.

If you are in immediate execution, try issuing an **FP** or **INT** command, whichever is appropriate to your program. When DOS executes an **FP** or **INT** command, it examines memory and consolidates all the available space. This will remove the gaps in memory that some programs cause.

You may also get this message if there is a problem with the program you are running. It also may indicate that a previous program set **HIMEM:** or **LOMEM:** to values that will not allow your present program to be loaded.

You can use the **MAXFILES** command to decrease the number of file buffers that are available to your BASIC program, or you can use **POKE** to place the correct values into the **HIMEM:** or **LOMEM:** locations. To find out more about these programming fixes, see the *DOS Programmer's Manual*.

RANGE ERROR

RANGE ERROR means that the value of a command's argument is too large or too small. Table B-3 shows which values can be used with which arguments.

Table B-3. Minimum and Maximum Argument Values. **Note:** Using **PR#** and **IN#** with arguments of 8 to 16 may give unpredictable results.

		Argument	Minimum	Maximum
All Files:	Slot	S	1	7
	Drive	D	1	2
	Volume	V	0	254
Sequential Text Files:	Byte	B	0	32767
	Relative Field	R	0	32767
	Absolute Field (EXEC)	R	0	32767
Random-Access Text Files:	Record Length	L	1	32767
	Record Number	R	0	32767
Binary Files:	Starting Address	A	0	65535
	Number of Bytes	L	1	32767
DOS Commands:	PR# n	n	0	16
	IN# n	n	0	16
	MAXFILES n	n	1	16

Using values beyond the range specified in Table B-3 does not always cause the **RANGE ERROR** message. Any DOS command with a value specified that is less than 0 or greater than 65535 may return a **SYNTAX ERROR**.

SYNTAX ERROR

SYNTAX ERROR means that you or your program issued a DOS command with an incorrect value or incorrect separator (comma or space). You will also see this message when the command lacks a required argument. Check the command's syntax and then try again.

The most common cause for this error message is a simple typing mistake. Check your spelling and then try again.

VOLUME MISMATCH

VOLUME MISMATCH means that the volume number (Vn) used in a DOS command differs from the volume number assigned to the disk when it was initialized. Use the **CATALOG** command to check the volume number of the disk.

WRITE PROTECTED

WRITE PROTECTED means that DOS is unable to save, write, or delete information on a specified disk. Write-protected means that the disk has no write-enable notch or that the notch is covered with a write-protect tab.

These are the most likely causes:

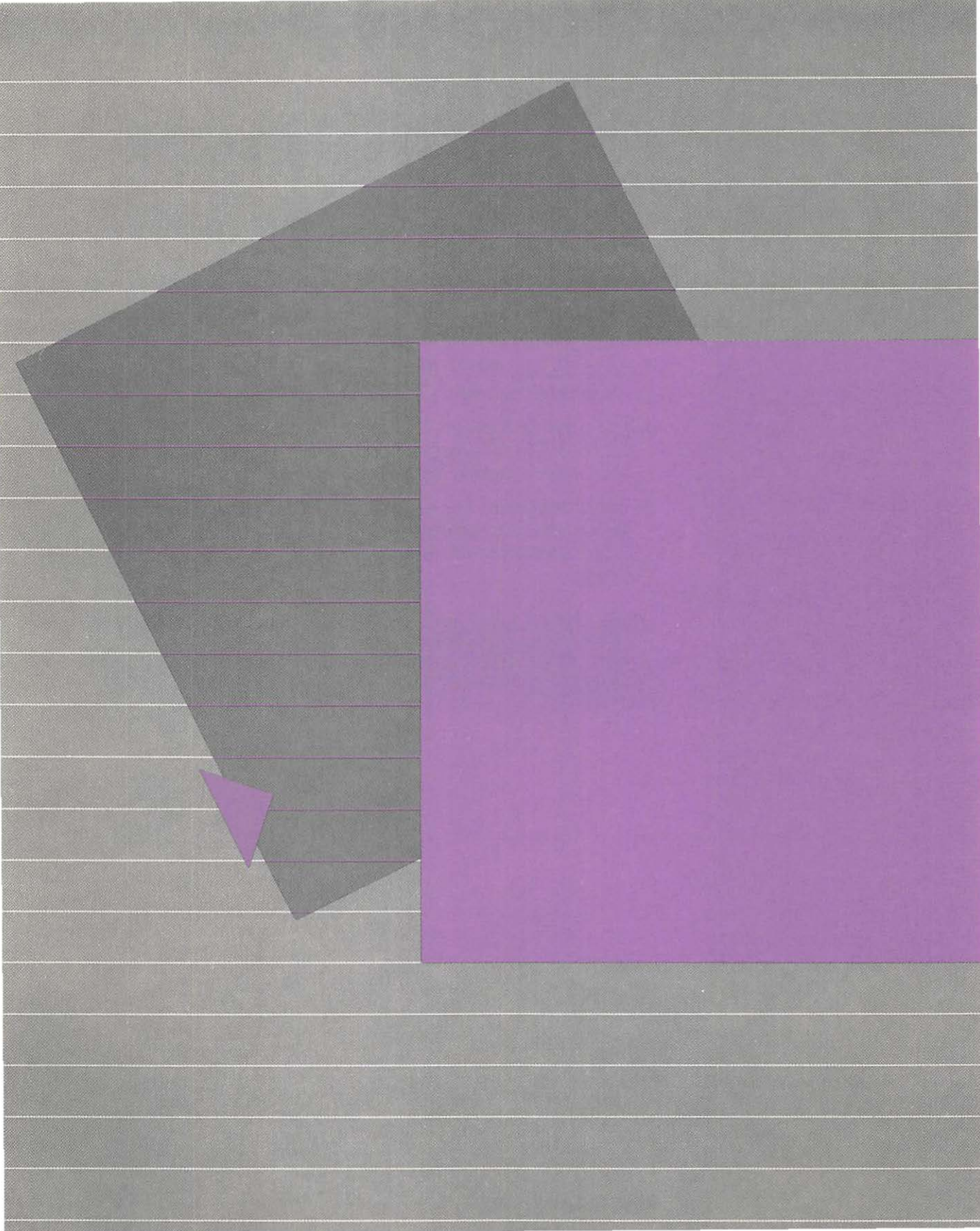
- There is an adhesive tab (write-protect tab) over the disk's write-enable notch. You may remove the tab and issue the command again.
- If you receive this message while running the copy program, you may have inserted the disk into the drive incorrectly. Check the disk's position in the drive.
- There may be no write-enable notch on the disk. Choose another disk to save your file on.

Stopping a Program

Occasionally, you will need to interrupt a program. Here are several things to try. They are listed in order of increasing severity. Eventually, your program will stop.

1. Press **(ESC)**. It's in the upper-left corner of your keyboard; it helps you "escape" from the program.
2. Press **(CONTROL)-(C)**. This cancels a program.
3. Press **(CONTROL)-(C)** and then **(RETURN)**. This cancels a program more emphatically.
4. Press **(CONTROL)-(RESET)**. This will stop your program and may return you to its opening menu. The program usually stays in memory.
5. If you have an Apple IIe, hold down **(⌘)** and then press **(CONTROL)-(RESET)**. Release **(⌘)** last. This is a very drastic measure, and it erases anything that is in main memory.
6. Turn off the power. You will rarely have to go this far to exit from a program. But it is a powerful way to show the computer who's boss. It does erase whatever programs and information you had in memory.

Note: When you see a hyphen joining two keys, it means to press the keys simultaneously. For instance, **(CONTROL)-(RESET)** means you should press **(CONTROL)** and **(RESET)** at the same time. In actual practice, you probably will press **(CONTROL)** first and then, while still holding down **(CONTROL)**, press **(RESET)**.



Programs on the DOS Disks

This appendix lists all of the programs that come on the SYSTEM MASTER and SAMPLE PROGRAMS disks.

Programs on the SYSTEM MASTER Disk

This section briefly describes the programs on the SYSTEM MASTER disk. When the program is used in this manual, it is marked with a cross (+). To find where the program is discussed, check the Index. When an entry does not have a cross (+), the program is covered in the *DOS Programmer's Manual*.

HELLO	is an Applesoft greeting program that DOS runs automatically, if the Applesoft language is available, when DOS is started. (+)
APPLESOFT	is an Integer BASIC greeting program that DOS runs automatically, if the Applesoft language is not available, when DOS is started.
BOOT13	is a binary program that starts up 13-sector disks, which were created under earlier versions of DOS.
CHAIN	is a binary program that loads and runs a second program without erasing from memory the variables and arrays of the first. Both chained programs must be Applesoft BASIC.
CONVERT13	is an Applesoft BASIC program that runs the MUFFIN program, which converts 13-sector disks to 16-sector disks. (+)

COPY	is the copy program to use when you are running Integer BASIC. (+)
COPY.Y.OBJ0	is a machine-language program used by COPY and COPY.A.
COPY.A	is the copy program to use when you are running Applesoft BASIC. (+)
FID	is a binary program that performs several support functions for DOS.
FILEM	is an Applesoft program that runs FID, a binary program, so that you can perform disk functions with groups of files. (+)
FPBASIC	is the Applesoft BASIC language on disk; it is in binary code.
INTBASIC	is the Integer BASIC language on disk; it is in binary code.
LOADER.OBJ0	is a machine-language program loaded by HELLO on the SYSTEM MASTER disk. LOADER.OBJ0 loads the alternate language into the language card or main memory. When the alternate language already exists in the computer, no action is taken.
MASTER	is an Applesoft program that runs the MASTER CREATE program, a binary program, so that you can convert an initialized disk to a master disk.
MASTER CREATE	is a binary program that converts an initialized disk into a master disk.
MUFFIN	is a binary program that converts your 13-sector disks to 16-sector format.
RENUMBER	is an Applesoft program (a programming tool) that can renumber the statements of your BASIC program and merge two programs.

SLOT#	is an Applesoft program that returns the current default values for slot and drive. (+)
START13	is an Applesoft program that runs BOOT13, which starts up 13-sector disks that were created under earlier versions of DOS. (+)

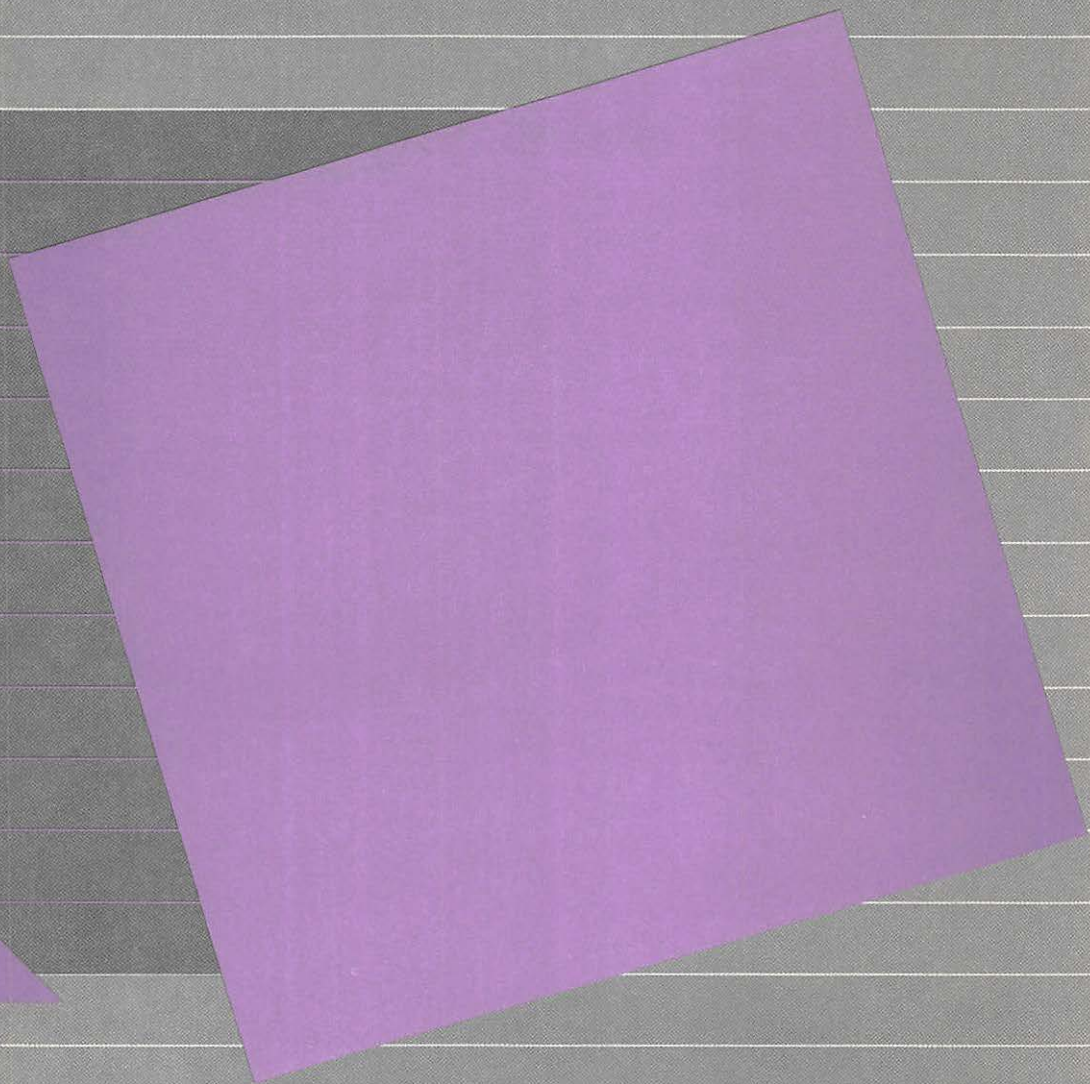
Programs on the *SAMPLE PROGRAMS* Disk

This section briefly describes the programs on the *SAMPLE PROGRAMS* disk. Many of these programs were used in this manual to demonstrate DOS commands. If so, the program is marked with a cross (+). To find where the program is discussed, check the Index. When an entry does not have a cross (+), the program is covered in the *DOS Programmer's Manual*.

HELLO	is an Applesoft greeting program. (+)
ADDRESS	is a sample program written in Applesoft that illustrates reading and writing of random-access text files. It uses BLACK BOOK to hold the data.
ANIMALS	is a game written in Integer BASIC for which you build a data file.
APPLE PROMS	is a data file for the RANDOM program. It contains data for a parts list.
APPLESOFT	is an Integer BASIC greeting program that DOS runs automatically, if the Applesoft language is not available, when DOS is started.
APPLEVISION	is a demonstration program, written in machine language, with an interface to Integer BASIC. (+)
BLACK BOOK	is a data file that stores the records of the ADDRESS program.
BRICK OUT	is a game program, written in Applesoft, that uses game paddles or the arrow keys on the keyboard. (+)

COLOR TEST	is a demonstration program, written in Applesoft, that helps you adjust a color TV set. (+)
DELETE.ME.1	is a demonstration program, written in Applesoft, that lets you practice deleting a file. (+)
DELETE.ME.2	is a demonstration program, written in Applesoft, that lets you practice deleting a file. (+)
DELETE.ME.3	is a demonstration program, written in Applesoft, that lets you practice deleting a file. (+)
EXEC DEMO	is a program, written in Applesoft, that demonstrates how an EXEC program is created and used.
FPBASIC	is the Applesoft BASIC language on disk; it is in binary code.
GET TEXT	is a sample Applesoft program that reads text files.
INTBASIC	is the Integer BASIC language on disk; it is in binary code.
LOADER.OBJ0	is a machine-language program loaded by HELLO on the SAMPLE PROGRAMS disk. LOADER.OBJ0 loads the alternate BASIC into the language card or main memory. If the alternate language already exists in the computer, no action is taken.
LOCK.ME.1	is a demonstration Applesoft program that lets you practice locking an unlocked file. (+)
LOCKED.UP.1	is a demonstration Applesoft program that lets you practice unlocking a locked file. (+)
LOCKED.UP.2	is a demonstration Applesoft file that lets you practice unlocking a locked file. (+)
MAKE TEXT	is a sample Applesoft program that illustrates how to create sequential text files.

ONERR DEMO	is a sample program that illustrates error recovery. It checks to see if a file is locked and, if so, lets you unlock it.
POKER	is a sample program, written in Applesoft, that translates machine-language into a text file.
PHONE LIST	is a demonstration Applesoft program. (+)
RANDOM	is a sample program, written in Applesoft, that illustrates reading and writing with a random-access text file. It uses APPLE PROMS to hold the data.
VERIFY.ME	is a demonstration Applesoft program that lets you practice verifying a file. (+)



Summary of DOS Operating Concepts and Commands

This appendix contains only the concepts, commands, and programs covered in the *DOS User's Manual*. For a more complete summary, refer to the *DOS Programmer's Manual*. That manual also has a summary card that you can tear out for easy reference.



Operating Concepts

DOS operates on all models of Apple II computers: the Apple II, the Apple II Plus, and the Apple IIe. Information and procedures that apply only to a specific model are indicated explicitly.

Starting Up

Starting up is the process of turning on the power to your computer (or simulating this same sequence) so that DOS is loaded into main memory. (See Chapter 1)

1. Insert a DOS disk into disk drive 1.
2. Turn on your display device, either a video monitor or a television set.
3. Find the power switch on the computer and turn it on.

On an Apple IIe computer, you can simulate a startup once the power is on. Hold down , and then press **CONTROL**-**RESET**. Release **RESET**, **CONTROL**, and then .

Restarting DOS

Restarting is the process of reconnecting DOS when the power is already on. Some restart methods do not affect the contents of main memory. Others destroy what is in main memory. The methods are listed in order of increasing severity.

1. Press **CONTROL-RESET**. This method does not affect the contents of main memory. This is the first method to try if your system is stuck.
2. Type **CALL SPACE 1002**. This is a BASIC command that reconnects DOS but does not affect the contents of main memory.
3. Type **PR SHIFT-#6** to restart DOS from drive 1, slot 6 on all models of the Apple II or press **⌘-CONTROL-RESET** on an Apple IIe. These two methods destroy whatever is in main memory. This is the method you should use to test a new system disk.
4. Type one of these:

6 CONTROL-P

6 CONTROL-K

3D0G (that's a zero)

C600G (these are zeros, too)

if you see the Monitor program's asterisk (*). If your disk controller card is installed in a slot other than slot 6, replace 6 in these commands with the number of that slot.

Capacity

A maximum of 496 disk sectors is available to the DOS user. Each disk sector can store up to 256 bytes of information.

Minimum file length is one sector for an empty text file (occupied by the track/sector list for the file) and two sectors for empty Applesoft BASIC, Integer BASIC, and binary files (one for the track/sector list and one for the first program sector, which contains the program's length.)

Notation

Here is an explanation of the notation used in this manual to indicate command syntax:

UPPERCASE	indicates the actual name of something, like a DOS command. Type it exactly as indicated.
lowercase	indicates something you supply, like the name of a program.
fn	indicates a filename that you supply.
[]	indicates an optional argument in a command. You may include the option or not, as you choose. In any case, <i>don't type the brackets</i> .
n	indicates a number you supply.

Syntax

A filename usually is the first argument after a command word; remaining arguments may appear in any order. Use a comma to separate the filename from an argument that follows. (See Chapter 2)

For example,

INIT fn [,Vn] [,Sn] [,Dn]

indicates the INIT command's syntax and should be interpreted this way:

- The command word INIT is uppercase and should be typed exactly as shown.
- The argument fn stands for "filename," and you should type an actual name in place of fn when you issue the command.
- The argument Vn is an optional argument. It stands for "volume number." When you use this argument, you should type the V (it's in uppercase) and then the number you want.
- The argument Sn is an optional argument. It stands for "slot number." When you use this argument, you should type the S (it's in uppercase) and then the number you want.
- The argument Dn is an optional argument. It stands for "drive number." When you use this argument, you should type the D (it's in uppercase) and then the number you want.

Arguments

Here's a quick run down of the arguments used in DOS commands. For more detailed information, see Chapter 2.

- **fn** stands for "filename."

A file's name may be from one to 30 characters. It must begin with a letter. Any typeable character except the comma may appear in a filename.

- **n** stands for "number."

You may use integer or hexadecimal numbers.

- **D** stands for "drive."

This argument must be used with a number, either 1 or 2. It specifies the disk drive you want to use. The drive number initially defaults to 1. Subsequently it defaults to the last drive number specified.

- **S** stands for "slot."

This argument must be used with a number, 1 through 7. It specifies the expansion slot you want to use. It initially defaults to the number of the slot from which DOS was started. Subsequently it defaults to the last slot number specified.

- **V** stands for "volume."

This argument must be used with a number, 0 through 254. It specifies the volume number you want to use. It initially defaults to the volume number of the disk from which DOS was started. Subsequently it defaults to the last volume number specified or read from a disk. When omitted in the INIT command, the volume number defaults to 254. This argument is rarely used.

Command Summary

Only the DOS commands covered in this manual are included in this appendix. The command summaries are listed in alphabetical order.

The chapter the command is discussed in is indicated.

CATALOG

Syntax: CATALOG [,Sn] [,Dn]

Example: CATALOG

This command displays a listing of the contents of the specified disk. It displays the volume number and the names of all the files on your disk, as well as some information about each file: whether the file is locked, what its file type is, and its size in sectors.

In the catalog, an asterisk indicates that the file is locked.

The file types are

- | | |
|---|---|
| A | Applesoft BASIC program file, created by SAVE. |
| B | Binary memory-image file, created by BSAVE. |
| I | Integer BASIC program file, created by SAVE. |
| R | Relocatable, assembly-language file (see the <i>6502 Assembler/DOS Tool Kit</i>) |
| S | For future use. |
| T | Text file, created by OPEN and filled by WRITE. |

When an individual file exceeds 255 sectors, the CATALOG display of that file's length starts over at 000. This gives an erroneous impression of the size of the file and the remaining space on the disk.

See Chapter 3.

DELETE

Syntax: DELETE fn [,Sn] [,Dn] [,Vn]

Example: DELETE TEST

This command removes a file from a disk. The file must be unlocked.

When the specified file does not exist on the disk, DOS displays the message `FILE NOT FOUND`.

See Chapter 4.

IN#

Syntax: IN# n

Example: IN# 6

This command reads the information, normally input from the keyboard, from the device that is connected through slot n. The # character is part of the command and must be typed.

Use IN#0 to direct DOS to read input from the keyboard once again.

When no controller card is in the slot you specified, DOS appears to be disconnected, and you have to restart the computer. (See the first section of this appendix.)

See Chapter 5.

INIT

Syntax: INIT fn [,Sn] [,Dn] [,Vn]

Example: INIT HELLO, V18

This command initializes a disk. In other words, it prepares a disk to receive information. The command divides the surface of the disk into tracks and sectors and places a copy of DOS and a greeting program on the disk.

See Chapter 3.



Warning

The INIT command wipes a disk clean. Be sure there is nothing you want to keep on the disk before you use this command.

LOAD

Syntax: LOAD fn [,Sn] [,Dn] [,Vn]

Example: LOAD DOW JONES, S6, D1

This command gets a copy of a program (file type A or I) from a disk file and puts it into main memory. Once the program is in memory, you can run it, modify it, or save it.

Before loading the new program into memory, DOS closes any files that are open, erases the contents of main memory, and changes to the BASIC that corresponds with the named file's type.

If the file type is A and Applesoft is neither in memory nor available from firmware, DOS looks for Applesoft on a disk. When Applesoft is not on the disk or your computer is not large enough to have room for both BASIC languages, DOS displays `LANGUAGE NOT AVAILABLE`.

If the file type is I, DOS looks for Integer BASIC in RAM (the language card) and then ROM. When DOS cannot find or load the language you need, it displays `LANGUAGE NOT AVAILABLE`.

See Chapter 5.

LOCK

Syntax: LOCK fn [,Sn] [,Dn] [,Vn]

Example: LOCK LOVE LETTERS, D2

This command protects a file from being accidentally altered. The entry for a locked file is preceded by an asterisk in the catalog. You cannot rename, delete, or change a locked file until it is unlocked.

See Chapter 4.

PR#

Syntax: PR# n

Example: PR# 6

This command sends the characters normally printed on the display to another device, such as a printer, that is connected through slot n. The # character is part of the command and must be typed.

Use PR#0 to direct output back to the display. If you have an Apple IIe computer with an 80-column text card, use PR#3 to direct output back to the display.

When no controller card is in the slot you specified, DOS appears to be disconnected, and you have to restart the computer. (See the first section of this appendix.)

See Chapter 5.

RENAME

Syntax: RENAME fn1,fn2 [,Sn] [,Dn] [,Vn]

Example: RENAME CURRENT, NEW, S5, D1

This command changes the name of a file from fn1, the current filename, to fn2, the new filename. The contents of the file are not affected. RENAME does not check to see whether the new name, fn2, already exists.

See Chapter 4.

RUN

Syntax: RUN fn [,Sn] [,Dn] [,Vn]

Example: RUN ANNUITY, D2

This command takes a copy of a BASIC program (file type A or I) from a disk file, puts it into memory, and executes the program.

See Chapter 5.

SAVE

Syntax: SAVE fn [,Sn] [,Dn] [,Vn]

Example: SAVE COLOR DEMOS, D2

This command writes the BASIC program currently in main memory to a disk file. An Applesoft BASIC program is saved as file type A. An Integer BASIC program is saved as file type I.

When a file with that name does not exist on the disk, DOS creates a new file and stores the program currently in memory in that file. If the disk already contains a file with the same filename and the same file type, DOS writes the program currently in memory over the contents of the existing file. When the disk contains a file with the same filename and a different file type, DOS displays `FILE TYPE MISMATCH`.

See Chapter 5.

UNLOCK

Syntax: UNLOCK fn [,Sn] [,Dn] [,Vn]

Example: UNLOCK RECIPES, D2

This command removes protection from a locked file, making it possible to rename, delete, or change it.

See Chapter 4.

VERIFY

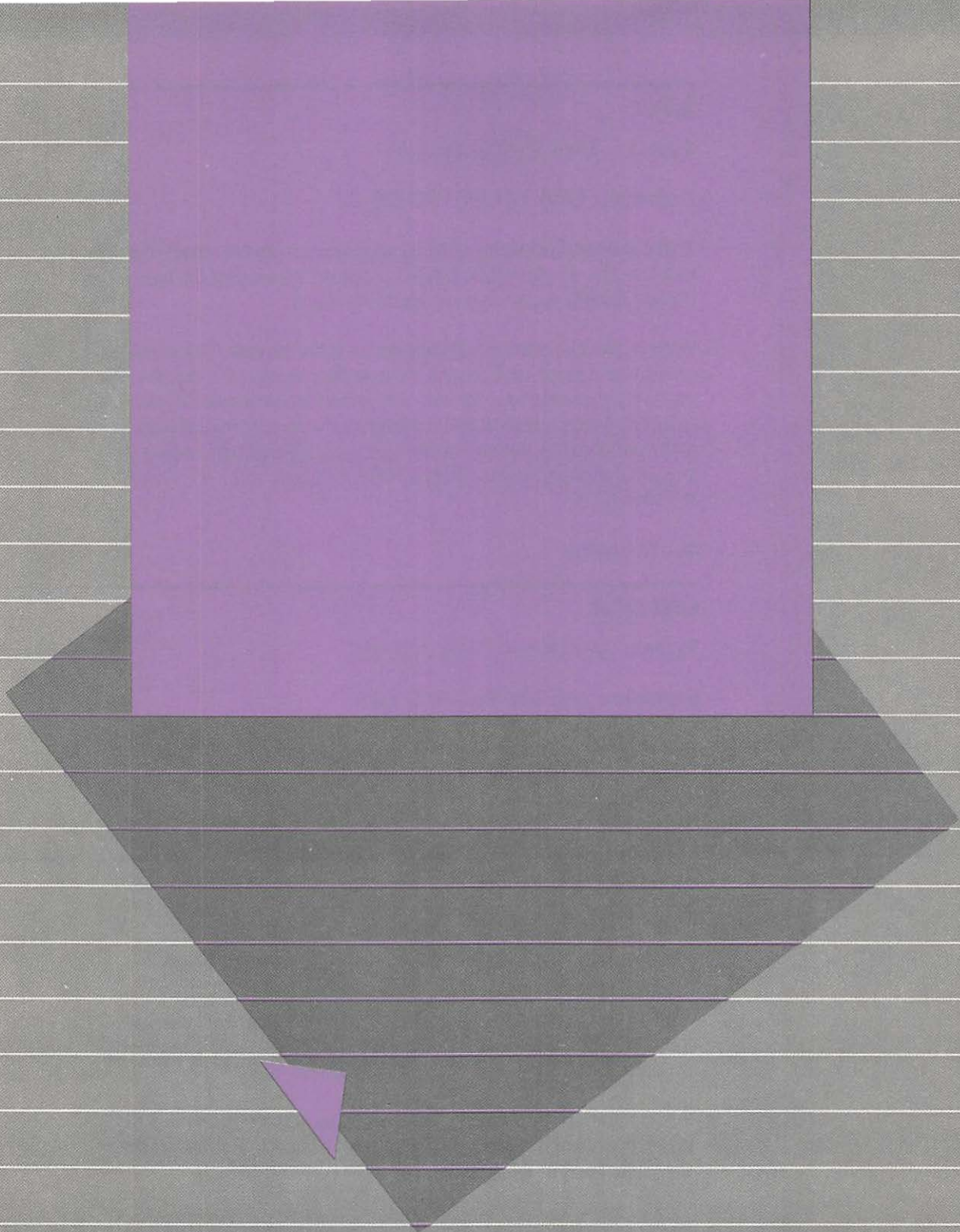
Syntax: VERIFY fn [,Sn] [,Dn] [,Vn]

Example: VERIFY SAM

This command makes sure that a file is stored properly on a disk. It tests whether DOS is able to read the file from the disk into the computer's memory.

When DOS can read the file, DOS displays no confirming message. When DOS cannot read the file, it displays `I/O ERROR`. You may verify any type of file.

See Chapter 4.



Glossary

This glossary includes computer terms that appear in this manual as well as some you may encounter in other Apple Computer, Inc. publications.

Applesoft BASIC: An extended version of the BASIC programming language, capable of processing numbers in floating-point form, used with the Apple II computer. An interpreter for creating and executing programs in Applesoft is built into the firmware of the Apple II Plus and Apple IIe systems. Compare **Integer BASIC**.

application program: A program that puts the resources and capabilities of the computer to use for some specific purpose or task, such as word processing, data-base management, graphics, or telecommunications.

application software: The component of a computer system consisting of application programs.

argument: An argument stands for a value that should be used with a command. In DOS commands, the common arguments are filename (fn), slot number (Sn), drive number (Dn), and volume number (Vn). In the last three arguments, *n* stands for the number.

ASCII: American Standard Code for Information Interchange; a code in which the numbers from 0 to 127 stand for text characters. It is used for representing text inside a computer and for transmitting text between computers or between a computer and a peripheral device.

assembly language: A low-level programming language in which individual machine-language instructions are written in a symbolic form more easily understood by a human programmer than machine language itself.

auxiliary slot: A special expansion slot inside the Apple IIe, used for an 80-column text card.

backing up: The process of making a copy of the information in a file or on a disk and putting that copy in another file or on another disk.

BASIC: Beginners All-purpose Symbolic Instruction Code; a high-level programming language designed to be easy to learn and use. Two versions of BASIC are available from Apple Computer, Inc. for use with the Apple II: Applesoft (built into the firmware of the Apple II Plus and Apple IIe computers) and Integer (built into the Apple II computer).

binary: The representation of numbers in terms of powers of two, using the two digits 0 and 1. Commonly used in computers because the values 0 and 1 can easily be represented in physical form in a variety of ways, such as the presence or absence of current, positive or negative voltage, or a white or black dot on the display screen.

bit: A binary digit (0 or 1); the smallest possible unit of information, consisting of a simple two-way choice, such as yes or no, on or off, positive or negative, something or nothing.

boot: To start up a computer by loading a program into memory from an external storage medium such as a disk. Often done by first loading a small program whose purpose is to read the larger program into memory. The program is said to "pull itself up by its own bootstraps"; hence the term *bootstrapping* or *booting*.

buffer: An area of the computer's memory reserved for a specific purpose, such as to hold graphic information or text characters. Often used as an intermediary holding area for transferring information between devices operating at different speeds, such as the computer's processor and a printer or disk drive. Information can be stored into the buffer by one device and then read out by the other at a different speed.

byte: A unit of information consisting of a fixed number of bits; in an Apple II, one byte consists of eight bits and can hold any value from 0 to 255.

catalog: A list of all files stored on a disk; sometimes called a *directory*.

character: A letter, digit, punctuation mark, or other written symbol used in printing or displaying information.

chip: The small piece of semiconducting material (usually silicon) on which an integrated circuit is fabricated. The word *chip* properly refers only to the piece of silicon itself, but is often used for an integrated circuit and its package; see **integrated circuit**.

code: (1) A number or symbol used to represent some piece of information in a compact or easily processed form. (2) The statements or instructions making up a program.

command: A communication from the user to a computer system (usually typed from the keyboard) directing it to perform some immediate action.

compiler: A language translator that converts a program written in a high-level programming language into an equivalent program in some lower-level language (such as machine language) for later execution. Compare **interpreter**.

computer: An electronic device for performing predefined (programmed) computations at high speed and with great accuracy.

copy-protect: To prevent the copying of information recorded on a storage medium, such as a disk containing software sold as a commercial product.

connector: A physical device, such as a plug, socket, or jack, used to connect one hardware component of a system to another.

control character: A character that controls or modifies the way information is printed or displayed. Control characters have ASCII codes between 0 and 31 and are typed from the keyboard by holding down **CONTROL** while typing some other character.

controller card: A peripheral card that connects a device, such as a printer or disk drive, to the Apple II computer and controls the operation of the device.

cursor: A marker or symbol displayed on the screen that marks where the user's next action will take effect or where the next character typed from the keyboard will appear.

data: Information; especially information used or operated on by a program.

data base: A collection of information organized in a form that can be processed by a computer system.

decimal: The common form of number representation used in everyday life in which numbers are expressed in terms of powers of ten, using the ten digits 0 to 9. Compare **binary**, **hexadecimal**.

default: A value, action, or setting that is automatically used by a computer system when no other explicit information has been given. For example, if a command to run a program from a disk does not identify which disk drive to use, DOS automatically uses the same drive that was used in the previous operation.

deferred execution: The saving of a program line for execution at a later time as part of a complete program; occurs when the line is typed with a line number. Compare **immediate execution**.

device: (1) A physical apparatus for performing a particular task or achieving a particular purpose. (2) In particular, a hardware component of a computer system.

digit: (1) One of the characters 0 to 9, used to express numbers in decimal form. (2) One of the characters used to express numbers in some other form, such as 0 and 1 in binary or 0 to 9 and A to F in hexadecimal.

disk: An information-storage medium consisting of a flat, circular, magnetic surface on which information can be recorded in the form of small magnetized spots, similarly to the way sounds are recorded on tape.

disk controller card: A peripheral card that connects one or two disk drives to the Apple II and controls their operation.

disk drive: A peripheral device that writes and reads information on the surface of a magnetic disk.

disk envelope: A removable protective paper sleeve used when handling or storing a disk; must be removed before inserting the disk in a disk drive. Compare **disk jacket**.

diskette: A term also used for the small (5¼ inch) flexible disks.

disk jacket: A permanent protective covering for a disk, usually made of black paper or plastic; the disk is never removed from the jacket, even when inserted in a disk drive.

Disk Operating System (DOS): A software system for the Apple II that enables the computer to control and communicate with one or more disk drives.

disk-resident: Stored or held on a disk.

display: (1) Information exhibited visually, especially on the screen of a display device. (2) To exhibit information visually. (3) A display device.

display device: A device that exhibits information visually, such as a television set or video monitor.

display screen: The glass or plastic panel on the front of a display device on which images are displayed.

DOS: See **Disk Operating System**.

drive: See **disk drive**.

80-column text card: A peripheral card that plugs into an Apple IIe's auxiliary slot and converts the computer's display of text from 40 to 80 columns.

envelope, disk: See **disk envelope**.

error message: A message displayed or printed that indicates an error or problem in the execution of a program.

execute: To perform or carry out a specified action or sequence of actions, such as those described by a program.

expansion slot: A connector inside the Apple II computer in which a peripheral card can be installed; also called *peripheral slot*.

file: A collection of information stored as a named unit on a peripheral storage medium such as a disk.

filename: The name under which a file is stored.

file type: The one-letter code that characterizes the contents of a file and indicates how the file may be used.

firmware: Those components of a computer system consisting of programs stored permanently in read-only memory. Such programs (for example, the Applesoft interpreter and the Monitor program) are built into the computer at the factory; they can be executed at any time but cannot be modified or erased from main memory. Compare **hardware**, **software**.

fixed-point: A method of representing numbers inside the computer in which the decimal point (more correctly, the binary point) is considered to occur at a fixed position within the number. Typically, the point is considered to lie to the right end of the number, so that the number is interpreted as an integer. Fixed-point numbers of a given length cover a narrower range than floating-point numbers of the same length, but with greater precision.

flexible disk: A disk made of flexible plastic; often called a *floppy* disk. Compare **rigid disk**.

floating-point: A method of representing numbers inside the computer in which the decimal point (more correctly, the binary point) is permitted to float to different positions within the number. Some of the bits within the number itself are used to keep track of the point's position. Floating-point numbers of a given length cover a wider range than fixed-point numbers of the same length, but with less precision. Compare **fixed-point**.

floppy disk: See **flexible disk**.

format: (1) The form in which information is organized or presented. (2) To specify or control the format of information. (3) To prepare a blank disk to receive information by dividing its surface into tracks and sectors; also known as *initializing*.

hardware: Those components of a computer system consisting of physical (electronic or mechanical) devices. Compare **software**, **firmware**.

hexadecimal: The representation of numbers in terms of powers of sixteen, using the sixteen digits 0 to 9 and A to F. Hexadecimal numbers are easier for humans to read and understand than binary numbers, but can be converted easily and directly to binary form: each hexadecimal digit corresponds to a sequence of four binary digits, or bits.

high-level language: A programming language that is relatively easy for humans to understand. A single statement in a high-level language typically corresponds to several instructions of machine language. BASIC is a high-level language.

immediate execution: The execution of a program line as soon as it is typed; occurs when the line is typed without a line number. Compare **deferred execution**.

information: Facts, concepts, or instructions represented in an organized form.

initialize: (1) To set to an initial state or value in preparation for some computation. (2) To prepare a blank disk to receive information by dividing its surface into tracks and sectors; also called *formatting*.

input: (1) Information transferred into a computer from some external source, such as the keyboard or a disk drive. (2) The act or process of transferring such information.

integer: A whole number, with no fractional part; represented inside the computer in fixed-point form.

Integer BASIC: A version of the BASIC programming language that is resident in the Apple II and available with the Apple II Plus and Apple IIe computers. It is older than Applesoft and processes numbers in integer form only. An interpreter for creating and executing programs in Integer BASIC is included on the DOS 3.3 SYSTEM MASTER disk and is automatically loaded into the computer's memory when the computer is started up with that disk if the computer has enough memory. Compare **Applesoft BASIC**.

integrated circuit: An electronic component consisting of many circuit elements fabricated on a single piece of semiconducting material, such as silicon; see **chip**.

interface: The devices, rules, or conventions by which one component of a system communicates with another.

interface card: A peripheral card that implements a particular interface by which the computer can communicate with a peripheral device, such as a printer.

interpreter: A language translator that reads a program written in a particular programming language and immediately carries out the actions that the program describes. Compare **compiler**.

I/O: The abbreviation for input/output; a general term for the equipment and process used to communicate with a computer, and the information involved in the communication.

I/O device: Input/output device; a device that transfers information into or out of a computer. See **input**, **output**, **peripheral device**.

K: Two to the tenth power, or 1024 (from the Greek root *kilo*, meaning one thousand); for example, 64K bytes equals 64 times 1024, or 65,536 bytes.

keyboard: The set of keys built into the Apple II computer, similar to a typewriter keyboard, for typing information to the computer.

kilobyte: A unit of information consisting of 1K (1024) bytes, or 8K (8192) bits; see **K**.

language: See **programming language**.

language translator: A system program that reads a program written in a particular programming language and either executes it directly or converts it into some other language (such as machine language) for later execution.

load: To transfer information from a peripheral storage medium (such as a disk) into main memory for use; for example, to transfer a program into memory for execution.

lock: To restrict the use of file so that it cannot be changed, deleted, or renamed. A locked file has an asterisk preceding its entry in a DOS catalog. Compare **unlock**.

low-level language: A programming language that is relatively close to the form that the computer's processor can execute directly.

machine language: The form in which instructions to a computer are stored in memory for direct execution by the computer's processor. Each model of computer processor (such as the 6502 microprocessor used in the Apple II computer) has its own form of machine language.

main memory: The memory component of a computer system that is built into the computer itself and whose contents are directly accessible to the processor.

memory: A hardware component of a computer system that can store information for later retrieval; see **main memory**, **random-access memory**, **read-only memory**.

memory location: A unit of main memory that is identified by an address and can hold a single item of information of a fixed size. In the Apple II, a memory location holds one character of information.

memory-resident: (1) Stored permanently in main memory, as firmware. (2) Held continually in main memory even while not in use, as the Disk Operating System.

menu: A list of choices presented by a program, usually on the display screen, from which the user can select.

microcomputer: A computer, such as the Apple II, whose processor is a microprocessor.

microprocessor: A computer processor contained in a single integrated circuit, such as the 6502 microprocessor used in the Apple II.

mode: A state of a computer or system that determines its behavior.

modem: Modulator/demodulator; a peripheral device that enables the computer to transmit and receive information over a telephone line.

monitor: See **video monitor**.

Monitor program: A system program built into the Apple II computer that is used for directly inspecting or changing the contents of memory and operating the computer at the machine-language level.

nibble: A unit of information equal to half a byte, or four bits. Can hold any value from 0 to 15. Sometimes spelled *nybble*.

operating system: A software system that organizes the computer's resources and capabilities and makes them available to the user or to application programs running on the computer. See **Disk Operating System**.

option: An argument that is optional.

output: (1) Information transferred from a computer to an external destination, such as display screen, disk drive, or printer. (2) The process of transferring such information.

peripheral: At or outside the boundaries of the computer itself, either physically (as a peripheral device) or in a logical sense (as a peripheral card).

peripheral card: A removable printed-circuit board that plugs into one of the Apple II's expansion slots and expands or modifies the computer's capabilities by connecting a peripheral device or performing some subsidiary or peripheral function.

peripheral device: An auxiliary piece of equipment that is under the control of the computer. Examples of this are printers, cassette recorders, and disk drives.

peripheral slot: See **expansion slot**.

printed-circuit board: A hardware component of a computer or other electronic device, consisting of a flat, rectangular piece of rigid material, commonly fiberglass, to which integrated circuits and other electronic components are connected.

printer: A peripheral device that writes information on paper.

processor: The hardware component of a computer that performs the actual computations by directly executing instructions represented in machine language and stored in main memory.

program: A set of instructions, conforming to the rules and conventions of a particular programming language, describing actions for a computer to perform to accomplish some task.

programming language: A set of rules or conventions for writing programs.

prompt: To remind or signal the user that some action is expected, typically by displaying a distinctive symbol, a reminder message, or a menu of choices on the display screen.

prompt character: A text character displayed on the screen to prompt the user for some action. Often also identifies the program or component of the system that is doing the prompting; for example, the prompt character `[` is used by the Applesoft BASIC interpreter, `>` by Integer BASIC, and `*` by the system Monitor program. Also called *prompting character*.

radio-frequency modulator: A device for converting the video signals produced by a computer to a form that can be accepted by a television set.

RAM: See **random-access memory**.

random-access memory: Memory in which the contents of individual locations can be referenced in an arbitrary or random order. This term is often used to refer to read-write memory, but strictly speaking both read-only and read-write memory can be accessed in random order. Compare **read-only memory**, **read-write memory**.

read: To transfer information into the computer's memory from a source external to the computer (such as a disk drive) or into the computer's processor from a source external to the processor (such as the keyboard or main memory).

read-only memory: Memory whose contents can only be read. Information is written to read-only memory once, during manufacture; it then remains there permanently, even when the computer's power is turned off; it can never be erased or changed. Compare **read-write memory**, **random-access memory**.

read-write memory: Memory whose contents can be both read and written; often misleadingly called random-access memory, or RAM. The information contained in read-write memory is erased when the computer's power is turned off and is permanently lost unless it has been saved on a more permanent storage medium, such as a disk. Compare **read-only memory**, **random-access memory**.

resident: See **memory-resident**, **disk-resident**.

rigid disk: A disk made of a hard, nonflexible material. Also called a *hard disk*. Compare **flexible disk**.

run: (1) To execute a program. (2) To load a program into main memory from a peripheral storage medium, such as a disk, and execute it.

ROM: See **read-only memory**.

save: To transfer information from main memory to a peripheral storage medium for later use.

screen: See **display screen**.

scroll: To change the contents of all or part of the display screen by shifting information out at one end (most often the top) to make room for new information appearing at the other end (most often the bottom), producing an effect like that of moving a scroll of paper past a fixed viewing window.

sector: A portion of the recording surface of a disk consisting of a fixed fraction of a track. Under DOS 3.3, there are 16 sectors per track.

silicon: A nonmetallic, semiconducting chemical element from which integrated circuits are made. Not to be confused with silica—that is, silicon dioxide, such as quartz, opal, or sand—or with silicone, any of a group of organic compounds containing silicon.

slot: See **expansion slot**.

software: Those components of a computer system consisting of programs that determine or control the behavior of the computer. Compare **hardware**, **firmware**.

space character: A text character whose printed representation is a blank space, typed from the keyboard by pressing the **(SPACE)** bar.

startup disk: A disk containing software recorded in the proper form to be loaded into the Apple II's memory to set the system into operation. Sometimes called a *boot disk*.

statement: A unit of a program in a high-level language specifying an action for the computer to perform, typically corresponding to several instructions of machine language.

syntax: The rules governing the structure of statements or instructions in a programming language.

system: A coordinated collection of interrelated and interacting parts organized to perform some function or achieve some purpose.

system program: A program that makes the resources and capabilities of the computer available for general purposes, such as an operating system or a language translator. Compare **application program**.

system software: The component of a computer system consisting of system programs.

telecommunications: The transmission of information across long distances, such as over telephone lines.

television set: A display device capable of receiving broadcast video signals (such as commercial television) by means of an antenna. Can be used in combination with a radio-frequency modulator as a display device for the Apple II computer. Compare **video monitor**.

turnkey program: A program, such as a game or application, that runs automatically when the disk it is on is started.

track: A portion of the recording surface of a disk consisting of a single circular band at a fixed distance from the center of the disk. Under DOS 3.3, there are 35 tracks on a disk.

unlock: To remove the restriction on the use of a file so that it can once again be changed, deleted, or renamed. Compare **lock**.

video monitor: A display device that is capable of receiving video signals by direct connection only and which cannot receive broadcast signals such as commercial television. Can be connected directly to the Apple II computer as a display device.

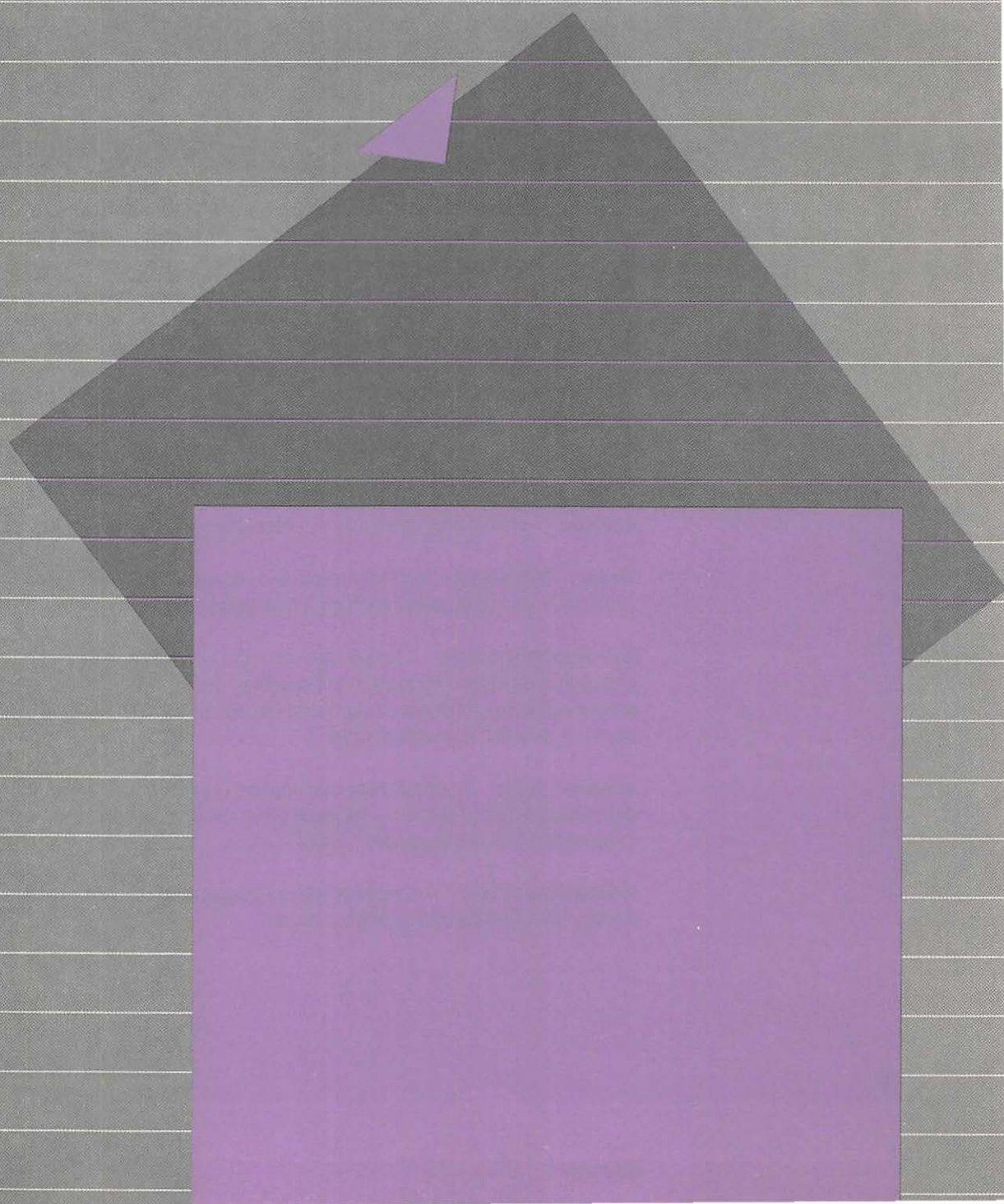
volume: The collection of all the information placed on a disk.

write: To transfer information from the computer to a destination external to the computer (such as a disk drive or printer).

write-enable notch: The square cutout in one edge of a disk's jacket that permits information to be written on the disk. If there is no write-enable notch or if the notch is covered, information can be read from the disk but not written onto it.

write-protect: To protect the information on a disk by covering the write-enable notch with a write-protect tab, preventing any new information from being written onto the disk.

write-protect tab: A small adhesive sticker used to write-protect a disk by covering the write-enable notch.



Index

A

adding lines 37
 ADDRESS program 139
 alternate BASIC 48,50
 ANIMALS program 139
 APPEND command 132
 Appendix A, overview of xiii,117
 Appendix B, overview of xiii,127
 Appendix B, overview of xiii
 Appendix C, overview of xiii
 Appendix D, overview of xiii
 Apple II 29,47,50,102,109
 Apple II Plus 29,50,102
 Apple IIe 29,50,110,150
 APPLE PROMS program 139,141
 Applesoft BASIC 13,16,18,66,
 102,105
 APPLESOFT program 137,139
 APPLEVSION program 41,
 103,139
 application programs xi
 that don't run 117
 argument(s) 53,55,102,145
 drive number (Dn) 55,57,131
 when used 57
 default 55
 filename (fn) 55,56
 optional 53,55,56,57
 required 55
 slot number (Sn) 55,56-57,131
 when used 56
 summary of 146
 volume number (Vn) 55,57-58
 when used 58
 ASCII code for null character 129
 autostart ROM 12,49

B

backing up 17
 backup copy 66
 BASIC 46
 Applesoft 102,105
 Integer 102,105,132
 disk 123,124
 languages 49-50
 binary 16
 BLACK BOOK program 139
 blank disk, preparation 17
 blank screen 12
 BOOT13 program 137
 booting 49
 brackets ([]) 53
 BRICK OUT program 139

C

CANCELLED message 86
 CAPS LOCK 11,15,64
 caring for disks 6-7
 catalog(s) 51
 parts of 65
 CATALOG command 14-17, 33,
 38,63-65,73,90,106,147
 causes of error messages,
 Table B-1 128
 CHAIN program 137
 changing a file 151
 changing contents of memory,
 steps 36-37
 changing defaults 55
 changing from 13-sector to 16-sector
 disks 117
 Chapter 1, overview of xiii,3
 Chapter 1, summary 42

- Chapter 2, overview of xiii,45
 - Chapter 2, summary 59
 - Chapter 3, overview xiii,63
 - Chapter 3, summary 73
 - Chapter 4, overview xiii,77
 - Chapter 4, summary 97
 - Chapter 5, overview xiii,101
 - Chapter 5, summary 112
 - character, null, ASCII code for 129
 - CLOSE command 129,132
 - COLOR TEST program 40,103,140
 - command,
 - APPEND 132
 - CATALOG 14-17,33,38,63-65,73,106,147
 - CLOSE 129,132
 - DELETE 82,147
 - END 32
 - FP 131,133
 - GET 129
 - IN# 107,110,112,148
 - INIT 31-34, 49,52,69,70,71-72,73,129,130,148
 - INPUT 129
 - INT 131,133
 - legal file type 130
 - LIST 35,104
 - LOAD 34,35,101,104,112,131,149
 - LOCK 80,105,149
 - MAXFILES 132,133
 - NEW 32,39,106
 - OPEN 132
 - POKE 132,133
 - POSITION 132
 - PR# 107,108-109,112,150
 - PRINT 32
 - READ 132
 - RENAME 77-79,150
 - RUN 32,40-41,101,102-103,112,131,150
 - SAVE 38-40,101,105-106,112,151
 - summary 147
 - syntax 53-55,63
 - UNLOCK 80,81,105,151
 - VERIFY 83,130,131,151
 - WRITE 132
 - commands, DOS 3,53-58
 - contents of disk erased with initialization, warning 66,71
 - CONTROL-C 135
 - and RETURN 135
 - CONTROL-RESET xiv,9,13,14,111,135
 - controller card 56,107
 - warning 107
 - CONVERT13 program 118,138
 - converting disks 117
 - COPY program 17,18,19,25,66,67,73,138
 - COPYA program 18,19,25,66-67,73,138
 - with one disk drive 67
 - with two disk drives 67
 - copying
 - checklist 28
 - locked files 90
 - problems 27-28
 - the SYSTEM MASTER disk 17-30
 - with FILEM, steps 87-90
 - with one disk drive 18-24
 - with two disk drives 24-27,88
 - copying, steps 18-27
 - COPY.OBJO program 138
- ## D
- default(s) 19,20,25,52,54,55
 - changing 55
 - number,254 52
 - slot number 56
 - DEFAULT message 18,19
 - default number, 254 52
 - default slot number 56
 - DELETE command 82,147
 - DELETE.ME.1 file 82,140
 - DELETE.ME.2 file 140
 - DELETE.ME.3 file 140
 - deleting a file 84,94,140,147
 - demonstration programs xii
 - on SAMPLE PROGRAMS disk 4
 - devices, peripheral 46
 - disk
 - BASICS 123,124
 - blank, preparation 17
 - capacity 144
 - caring for 6-7
 - changing from 13-sector to 16-sector 117
 - contents of 14-17
 - controller card 19,25,49,120
 - converting 117
 - DOS xii
 - DOS 3.3 SAMPLE PROGRAMS xii,4,24,27,40,64
 - DOS 3.3 SYSTEM MASTER xii,4,10,17
 - drive xi,4
 - duplicate 25,66
 - envelope 33
 - flexible 4
 - how to care for 4
 - initializing 118

- jacket 5
- master 69
- operating system xi
- original 25,66
- preparing 69-72
- problems 115-123
- reading a 34
- removing file from 82
- running a copy-protected 115-123
- 16-sector xi,117
- system 17
- slave 69
- startup 49
- stopping a runaway 10
- testing a copied 28-30
- that won't run 115-123
- 13-sector xi,117,137
- running 123
- what it is 4
- DISK FULL message 128
- display device 10
- display, opening 46
- DO YOU WANT PROMPTING?
 - message 85,89,91
- DO YOU WISH TO MAKE ANOTHER COPY? message 22,27,28,68
- DOS xi,4,45,117
 - commands 3,53-58,147-151
 - disks xii
 - how it works 45-50
 - in memory 10
 - introduction to xi
 - manuals xii
 - overview of 45-59
 - Programmer's Manual xii
 - reading 10
 - restarting 111
 - starting 10
 - summary of operating concepts 143-151
 - 3.1 117
 - 3.2 117
 - 3.3 SAMPLE PROGRAMS disk
 - xii,4,24,27,40,64
 - programs on the 139-141
 - 3.3 SYSTEM MASTER disk
 - xii,4,10,17
 - copying 17-30
 - programs on the 137-139
 - using 10
- drive 1 19,49,54
- drive number (Dn) argument 55,57,131,145
- default 57
- when used 57
- duplicate
 - disk 25,66
 - filenames 122
- DUPLICATE message 18,19
- E**
- editing 34
- 80-column text card 108,150
 - warning 108
- END command 32
- END OF DATA message 128
- equal sign (=) 93,120
- error messages 58,68
 - and warnings 127-135
 - causes of, Table B-1 128
 - from bad arguments 58
- error recovery 141
 - ESC** 135
 - ESC CONTROL-O** 108
- EXEC DEMO program 140
- expansion slot(s) 25,49,107,119
- F**
- FID 84,95,138
- figures and tables, list of vii
- file 36
 - size 17,65
 - type 16,51,65
- FILE LOCKED message 78, 80,82,129
- FILE NOT FOUND message 35,78,82,83,94,102,104,129,147
- FILE TYPE MISMATCH message 102,104,130,151
- file,
 - changing a 151
 - DELETE.ME.1 82
 - deleting a 140
 - LOCK.ME.1 80
 - locked 16
 - LOCKED.UP.1 81
 - locking a 140
 - PHONE LIST 79
 - protecting a 149
 - putting into memory 149
 - removing from disk 82
 - storing a 151
 - unlocked 16
 - unlocking a 140
 - VERIFY.ME 83
 - verifying a 141
- FILEM
 - messages 95-96
 - options 87-95
 - CATALOG 90
 - COPY FILES 87-90

- DELETE FILES 94
- LOCK FILES 92-93
- QUIT 95
- RESET SLOT & DRIVE 94
- SPACE ON DISK 90-91
- UNLOCK FILES 91-92
- VERIFY FILES 94
 - program 68,84-96,138
 - specifying filenames with the wildcard character 85-86
 - specifying slot and drive numbers 85
 - wildcard 91
- filename(s) 17,39,51,52,65,105
 - argument (fn) 55,56,145
 - changing 77
 - duplicate 122
 - legal 52
 - length 52
 - rules 52
- files,
 - copying 87
 - copying locked 90
 - sequential text 140
- firmware 49
- flexible disk 4
- floppy disk See flexible disk
- fn See filename
- formatting 31,69
- FP command 131,133
- FPBASIC program 138,140

G

- games on SAMPLE PROGRAMS
 - disk 4
- GET command 129
- GET TEXT program 104,140
- getting started 4
- Glossary, overview of xiii
- gray box xiv
- greater than (>) prompt 13,18, 19,25,67,103
- greeting program 31,69,70

H

- hardware 49
- HELLO program 32,33,48,70, 137,139
- hexadecimal number 55
- HIMEM: program 133
- how DOS works 45-50
- hung system 111
- hyphen joining two keys xiv

I

- I/O ERROR message 27,28, 33,58,83,94,105,107,130,151
- I/O ERROR STOPPED
 - AT (line number) message 68
- IN#
 - command 107,110,148
 - warning 107,110
- INIT
 - command 31-34,49,52,69, 70,71-72,73,129,130,148
 - warning 148
- initializing 31,69
 - disk 118
 - steps 31-34
- INITIALIZING message 22,26
- INPUT command 129
- inserting a disk into a disk drive 7-9
- INT command 131,133
- INTBASIC program 138,140
- Integer BASIC 13,16,18,66,102,105
- integer number 55

J

K

- K xi
- keeping a file 105-106
- kilobytes xi

L

- language
 - card 50,102,132
 - switching 103
- LANGUAGE NOT AVAILABLE
 - message 41,103,131-132,149
- language, resident 50
- LEFT ARROW key 15,32
- legal filenames 52
- LIST command 35,104
- list of figures and tables vii
- LOAD
 - command 34,35,101,104,112, 131,149
 - example 104
- LOADER.OBJO program 138,140
- loading 34
 - steps 35
- LOADING APPLESOFT
 - BASIC INTO MEMORY message 50

LOADING INTEGER BASIC
 INTO MEMORY message 50
 LOCK command 80,105,149
 lock status 65
 LOCK.ME.1 file 80,140
 locked file 16
 LOCKED.UP.1 file 81,140
 LOCKED.UP.2 file 140
 locking a file 140
 with FILEM, steps 92-93
 LOMEM: program 133

M

magnetic storage device See disk
 main memory 10,46
 MAKE TEXT program 140
 marginal notes xiv
 MASTER CREATE program 138
 master disk 69
 MASTER
 file 89
 program 70,128,138
 MAXFILES command 132,133
 memory 35,45,50
 changing contents of, steps
 36-37
 main 10,46
 random-access xi,50,132
 read-only 50,132
 size 34,69
 menu 40
 merging programs 138
 message,
 CANCELLED 86
 DEFAULT 18,19
 DISK FULL 128
 DO YOU WANT
 PROMPTING? 85,89,91
 DO YOU WISH TO MAKE
 ANOTHER COPY? 22,27,
 28,68
 DUPLICATE 18,19
 END OF DATA 128
 FILE LOCKED 78,80,82,129
 FILE NOT FOUND 35,78,
 82,83,94,102,104,129,147
 FILE TYPE MISMATCH
 102,104,130,151
 I/O ERROR 27,28,33,58,
 83,94,105,107,130,151
 I/O ERROR STOPPED AT
 (line number) 68
 INITIALIZING 22,26
 LANGUAGE NOT AVAIL-
 ABLE 41,103,131-132,149

LOADING APPLESOFT
 BASIC INTO MEMORY 50
 LOADING INTEGER
 BASIC INTO MEMORY 50
 NO BUFFERS AVAILABLE
 132
 NOT DIRECT COMMAND
 132
 ORIGINAL 18,19
 OUT OF MEMORY 69
 PROGRAM TOO LARGE 133
 RANGE ERROR 58,133
 READING 21,26
 SYNTAX ERROR 38,52,134
 UNABLE TO READ 27,68
 UNABLE TO WRITE 27,68
 VOLUME MISMATCH 134
 WRITE-PROTECTED 134
 WRITING 26

messages,
 FILEM's 95-96
 mysterious 12
 models of Apple II xiv
 modem 107,110
 Monitor
 program 46,48
 ROM 109
 MUFFIN program 118,137,138
 mysterious messages 12

N

n (number) 53,55
 NEW command 32,39,106
 new computer user xii
 NO BUFFERS AVAILABLE
 message 132
 NOT DIRECT COMMAND
 message 132
 notation 145
 notes in the margin xiv
 null character, ASCII code for 129
 number (n) argument 145
 number,
 hexadecimal 55
 integer 55
 number (n) 53,55

O

ONERR DEMO program 141
 ⌘ and (CONTROL)-(RESET) 135
 ⌘-(CONTROL) 29
 OPEN command 132
 opening display 46
 operating concepts 143
 option 54

- optional argument 55,56,57
- original disk 25,66
- ORIGINAL message 18,19
- OUT OF MEMORY message 69
- overview,
 - Appendix A xiii,117
 - Appendix B xiii,127
 - Appendix C xiii
 - Appendix D xiii
 - Chapter 1 xiii,3
 - Chapter 2 xiii,45
 - Chapter 3 xiii,63
 - Chapter 4 xiii,77
 - Chapter 5 xiii,101
 - Glossary xiii

P

- peripheral
 - card 102
 - devices 46,107
- PHONE LIST
 - file 79
 - program 141
- POKE command 132,133
- POKER program 141
- POSITION command 132
- pound sign (#) character 107
- PR# command 107,108-109,150
 - starting up a disk using 109
 - example 109
 - warning 107,108
- PR#0 107,108,150
- PR#1 107
- PR#3 108,150
- PR#n command 112
- preparing a blank disk 31-34,69-72
- PRINT command 32
- printer 107
 - deactivating 108
 - sending information to 109
- problems, startup 12
- PROGRAM TOO LARGE
 - message 133
- program, 4
 - ADDRESS 139
 - ANIMALS 139
 - APPLE PROMS 139,141
 - APPLESOFT 137,139
 - APPLEVISION 41,103,139
 - application xi
 - BLACK BOOK 139
 - BOOT13 137
 - BRICK OUT 139
 - CHAIN 137
 - COLOR TEST 40,103,140

- CONVERT13 118,138
- COPY 17,18,19,25,66,67,73,138
- COPYA 18,19,25,66-67,73,138
 - with one disk drive 67
 - with two disk drives 67
- COPY.OBJO 138
- DELETE.ME.1 140
- DELETE.ME.2 140
- DELETE.ME.3 140
- EXEC DEMO 140
- FID 138
- FILEM 68,84-96,138
- FPBASIC 138,140
- GET TEXT 104,140
- greeting 31,69,70
- HELLO 32,33,48,70,137,139
- HIMEM: 133
- INTBASIC 138,140
- LOADER.OBJO 138,140
- LOCK.ME.1 140
- LOCKED.UP.1 81,140
- LOCKED.UP.2 140
- LOMEM: 133
- MAKE TEXT 140
- MASTER 70,129,138
- MASTER CREATE 138
- Monitor 46,48
- MUFFIN 118,137,138
- ONERR DEMO 141
- PHONE LIST 141
- POKER 141
- RANDOM 141
- RENUMBER 138
- running a 102
- SLOT# 66,72,73,138
- START13 123,138
- stopping a 135
- turnkey 71
- VERIFY.ME 141
- programming xii
 - tool 138
- programs
 - application xi
 - demonstration xii
 - games xii
 - merging 138
 - on the DOS 3.3 SAMPLE
 - PROGRAMS disk 139-141
 - on the DOS 3.3 SYSTEM
 - MASTER disk 137-139
 - ready-made xii
 - that don't run 117
 - using 34-41
- prompt character 13,47

- prompt,
 - greater than (>) 13,18,25,67,103
 - square bracket (]) 13,18,19, 25,103
- protecting a file 149
- putting a file into memory 149
- Q**
- R**
- RAM 50,132
- RANDOM program 141
- random-access memory xi,50,132
- RANGE ERROR message 58,133
- READ command 132
- read-only memory 50,132
- READING message 21,26
- removing a disk from a disk drive 9
 - while drive's light is on, warning 9,13
- RENAME
 - command 77-79,150
 - warning 78
- RENUMBER program 138
- RESET 29,131
- resident language 50
- restarting 30,58,107,150
 - Apple IIe 29
 - DOS 14,58,109,111,144
 - methods 29
- ROM 50,132
- ROM, autostart 12,49
- RUN
 - command 32,40-41,101, 102-103,112,131,150
 - example 103
- running
 - a copy-protected disk 115-123
 - a disk program, steps 40-41
 - a program 102
 - 13-sector disks 123-125
- S**
- SAVE
 - command 38-40,101,105-106, 112,151
 - example 106
 - warning 105
- saving 34
 - information on a disk 38-40
- screen blank 12
- sectors 17,33,65,69
- sending information to printer 109
- sequential text files 140
- (SHIFT) key xiv,15
- (6) (CONTROL)-(P) 12,47
- 16-sector disks xi,117
- 64K 50,132
- slave disk 69
- slot 6 25,49,55
- slot number (Sn) argument 55, 56-57,131,145
 - when used 56
- slot number default 56
- SLOT# program 66,72,73,138
- solving startup problems 12-13
- (SPACE) bar xiv,14,15
- space character 14,38
- SPACE ON DISK 90-91
- special notes xiv
 - marginal xiv
- square bracket (]) prompt 13,18, 19,25,103
- square bracket notation, warning 54
- START13 program 123,138
- starting up 143
 - a disk using PR# command 109
 - what it is 13-14
- startup
 - disk 49
 - drive 49
 - problems,
 - asterisk 12
 - blank screen 12
 - drive whirring 12
 - lights 12
 - messages 12
 - power indicators 12
 - solving 12
 - strange sounds 13
 - process 46-50
- steps,
 - copying SYSTEM MASTER disk 17-30
 - copying with FILEM 88-90
 - copying with one disk drive 18-24
 - disks, converting 13-sector to 16-sector 119-121
 - initializing 31-34
 - inserting a disk into a disk drive 7-9
 - locking files with FILEM 92-93
 - running a disk program 40-41
 - startup 46-50
 - stopping a program 135
 - unlocking files with FILEM 91-92
- stopping a program 135
- storing a file 151
- strange sounds 13
- summary of arguments 146
- summary of commands 147

- summary,
 - Chapter 1 42
 - Chapter 2 59
 - Chapter 3 73
 - Chapter 4 97
 - Chapter 5 112
 - syntax 145
- syntax 71
 - command 53-55,63
 - summary 145
- SYNTAX ERROR message
 - 15,38,52,134
- system
 - disks 17
 - hung 111
 - program xi

T

- tables and figures, list of vii
- talking to other devices 107-110
- testing a copied disk 28-30
- text files, sequential 140
- 13-sector disks 117-125,137
 - running with 16-sector DOS xi,123
- 32K system 34
- tracks 69
- turning on the computer 41
- turnkey 125
 - program 71
- tutorial 3

U

- UNABLE TO READ message
 - 27,68
- UNABLE TO WRITE message
 - 27,68
- UNLOCK command 80,81,105,151
- UNLOCK FILES 91-92
- unlocked file 16
- unlocking a file 140
 - with FILEM, steps 91-92
- uppercase in commands 15
- using programs 34-41

V

- VERIFY command 35,83,130,
 - 131,151
- VERIFY FILES 94
- VERIFY.ME
 - file 83
 - program 141
- verifying a file 141
- volume 52,55
- VOLUME MISMATCH message
 - 58,134

- volume number 33,63,71
 - (Vn) argument 55,57-58,134,135
 - when used 58
 - default 58

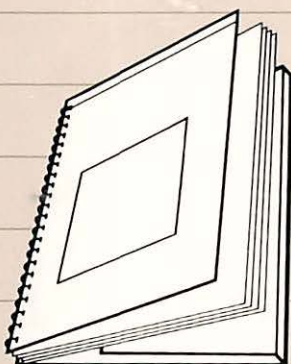
W

- warnings 9,13,54,66,71,78,105,
 - 107,108,110,148
 - and error messages 127-135
- warning boxes, their purpose xiv
- wildcard 85,86,87,89,91,120,122
- WRITE command 132
- write-enable notch 5
- write-protect 66
- write-protect tab 6,24
- write-protected 134
- WRITE - PROTECTED message
 - 134
- WRITING message 26

X

Y

Z



Tuck end flap
inside back cover
when using manual.



Apple II

DOS User's Manual



20525 Mariani Avenue
Cupertino, California 95014
(408) 996-1010
TLX 171-576

030-0407-A