APPLE® IIe

Programmer's Reference Guide

APPLE IIE PROGRAMMER'S REFERENCE GUIDE

by

David L. Heiserman

Howard W. Sams & Co., Inc.

4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 462688

Preface

The Apple IIe is a wonderfully dynamic microcomputer that fits into a lot of different environments. It is a home computer-a personal computer-that is easy to set up and operate. Yet it fits equally well into the more demanding environments of business, education, and engineering.

It would be unrealistic to suppose that any single book could venture into all of the features of the Apple IIe system in great detail. There is simply too much to learn about and use. A more realistic approach-the one used in this book-is to introduce the main features, describe some important variations, and suggest some techniques for exploring the finer details at your own leisure.

A beginner can benefit from the discussions of the system's main features: how to set up a system (Chapter 1), some general rules and procedures for using Applesoft and Integer BASIC (Chapter 2), and a survey of the BASIC statements, commands and functions (Chapter 3).

Readers who already have some experience with BASIC programming can learn about some of the finer points of text formatting (Chapter 4), color graphics (Chapters 5 and 6), and more advanced disk operations (Chapter 7).

Programmers who are prepared to deal with the Apple IIe in more subtle ways can benefit from descriptions of the memory maps (Chapter 8) and 6502 machine-language programming techniques (Chapter 9).

The foregoing description suggests that the material in this book is organized according to the level of know-how and skill required of the reader. That is indeed the case. But the main topics are organized and titled in such a fashion that should make it relatively easy to locate a topic of special interest; there is no need to read the book from the first page to the last in order to find what you want to know.

I would like to acknowledge the invaluable assistance offered by my personal secretary, Robin M. Yates. She prepared many of the tables and drawings, proofread the drafts of the manuscript, double-checked the operation of the sample programs, and assembled the finished manuscript.

DAVID L. HEISERMAN

Contents

CHAPTER 6

CHAPTER 7

CHAPTER 8

CHAPTER 9

APPENDIX A

APPENDIX B

APPENDIX C

APPENDIX D

CHAPTER 1
General Start-Up and Operating Procedures

The fundamental Apple IIe system consists of the Apple IIe console, a tv set or monitor that serves as a display screen, and some mechanism for saving and loading programs from magnetic tape or disk. Beyond that are dozens of optional features and variations.

The discussions and examples cited in this book assume that the reader has at least a minimum system: the Apple IIe console, a suitable display system (preferably a color display), and either a cassette recorder/player or a single disk drive. Details regarding the optional features are included in the manuals that accompany them.

STARTING UP THE SYSTEM

Applesoft BASIC is permanently built into the Apple IIe system, so it is rather easy to get the system started and running in BASIC. The exact start-up procedure that the microcomputer operator is required to use depends on whether or not a disk drive is connected to the Apple IIe system; but even then, the difference is almost trivial.

Starting Up an Apple IIe With No Disk Drives

If there are no disk drives connected to the system, simply turn on the tv receiver or monitor, and switch on the Apple IIe POWER switch located on the back of the unit, near the left-hand side. You will immediately hear a single beep from the console loud speaker, and then you will see the BASIC prompt symbol-a right bracket (])-and the blinking, checkerboard cursor symbol. That particular prompt symbol indicates that the system is indeed running in BASIC, and the blinking cursor symbol indicates where the next-typed keyboard character will appear on the screen.

Starting Up an Apple IIe With One or More Disk Drives

If there is one or more disk-drive units connected to the system, you must insert a properly formatted diskette into drive 1 before turning on the power. So turn on the tv set or monitor, insert a formatted disk into drive 1 (one of the system disks that are supplied with the Apple IIe does the job quite nicely), and then turn on the POWER switch.

You will hear a single beep from the console loudspeaker and notice the disk drive going to work-the red IN USE lamp goes on and the drive mechanism makes some buzzing and clicking sounds. During that time, the system is loading and *booting the* Disk Operating System (DOS) programming.

What happens after that depends on the kind of programming that is included on the disk. If you are using a newly formatted disk (a procedure described later in this chapter) or the DOS 3.3 System Master disk, the start-up sequence soon concludes by showing Applesoft's prompt and cursor symbols-symbols that indicate that the system is ready to run under BASIC.

Incidentally, should you fail to insert a disk into drive 1 prior to turning on the POWER switch, the drive unit will run endlessly. The best procedure in that case is to turn off the POWER switch on the console, insert a properly formatted disk, and turn the POWER on again.

In those instances where you do not want DOS booted up in the system, make sure there is no disk in the disk-drive unit, turn on the power and then strike the RESET key while holding down the CONTROL key. The disk drive will run for a few seconds; and after

that, you can work with the system without the DOS programming in memory.

Optional Self-Diagnosis

If instructed to do so, the Apple IIe will carry out a series of internal hardware tests, Once the system is powered up, whether you are using DOS or not, you can invoke this 20-second internal testing procedure by holding down the OPEN-APPLE key (located at the left side of the SPACE BAR), depressing the CONTROL key, and striking the RESET key.

That keyboard operation begins the testing procedure. As long as the Apple is passing its tests, you will see patterns of highresolution light moving on the screen and a closing message, KERNAL OK. Any other message or response indicates an internal hardware program.' You can then start programming after holding down the CONTROL key while striking the RESET key.

GETTING FAMILIAR WITH IIE KEYBOARD ASSEMBLY

After the Apple IIe is properly started, or initialized, you will see the right-bracket prompt symbol and the blinking, checkerboard cursor symbol on the screen. In essence, that means that the system is ready to accept BASIC commands from the keyboard.

Much of the material that remains in this book deals with the nature of those commands. The computer is virtually useless without some commands; and unless those commands are loaded in the form of prescribed programs from cassette tape or disk, they must come directly from your work at the keyboard.

Most of the keys on the Apple IIe keyboard are identical to those of a conventional typewriter; and, indeed, they can serve the same general purpose. However, there are some additional keys that would be irrelevant for ordinary typing operations, but are quite important for operating a computer.

The RETURN key is perhaps the most-used control key. You must strike that key whenever you want the computer to execute a command that you have given it in a typewritten form. When you are ready to execute a BASIC program, for example, you should type RUN on the keyboard and then strike the RETURN key to get the computer to read and execute that program.

Of course the keyboard includes 26 keys for the letters of the alphabet. As in the case of a normal typewriter, you at least have the option of typing them to the screen as upper- or lower-case characters. Applesoft BASIC, however, accepts commands and program statements only in the upper-case format. The lowercase letters are reserved for literal messages that are to be printed to the screen.

The two SHIFT keys serve much the same function as the shift keys on an ordinary typewriter. If the CAPS LOCK key is locked in its down position, the letters of the alphabet always appear in an upper-case form, whether you are holding down one of the SHIFT keys or not. It is the typing mode that is most useful for entering BASIC commands and programs, because those characters must be entered as upper-case characters.

HINT: Make a habit of locking down the CAPS LOCK key every time you start up the APPLE IIe system.

Unlock the CAPS LOCK key, however, and the letters will be printed in an upper- or lower-case form, depending on whether or not you are holding down one of the SHIFT keys. That mode is the one that most closely resembles the operation of an ordinary typewriter keyboard: hold down one of the SHIFT keys to print upper-case letters, and release the SHIFT keys to print lower-case letters.

That CAPS LOCK feature appIIes only to the letters of the alphabet. No matter what CAPS LOCK mode you are using, you can print the symbols located on the numeric keys only by holding down one of the SHIFT keys. Striking the #/3 key for example, prints a 3 if you are not holding down a SHIFT key at the same time. And striking the #/3 key while holding down one of the SHIFT keys always prints a # to the screen. It makes no difference whether the CAPS LOCK key is locked down or up.

The ESC and CONTROL keys, in effect, multiply the number of key functions that are available. Striking the A key without using it in conjunction with the ESC or CONTROL key simply prints that character onto the screen. That same keystroke takes on different meanings when it is used with either an ESC or CONTROL function.

Normally, an ESC (spoken as "escape") function is performed

by first striking the ESC key and then striking another key-the A key, for instance. Throughout this book, an escape sequence is shown as:

ESC/*key*

where *key* is some other keystroke. For example, ESC/A literally means: strike the ESC key, release it, and then strike the A key.

By way of a working example, you can clear the screen and home the cursor by doing an ESC/@ sequence.

A CONTROL function is performed differently. Instead of using a sequence of keystrokes, a CONTROL function is enabled by holding down the CONTROL key while striking some other key. That sort of function is shown in this book as:

CONTROL-key where *key is* some other designated keystroke. An operation that is described as CONTROL-A literally means: hold down the CONTROL key while striking the A key.

Test your understanding of CONTROL operations by doing a CONTROL-G-that beeps the loudspeaker that is built into the console unit.

NOTE: Some text and programmers prefer the following notation for CONTROL-key operations:

^key

The carat (^) means the same as CONTROL. So the equivalent of a CONTROL-A operation is ^A

The TAB key serves the same function as the corresponding key on a typewriter. Striking that key jumps the cursor to the next tab stop-stops that are normally set at intervals of eight columns.

There are four arrow keys located in the lower right-hand corner of the keyboard. The purpose of these keys is to allow you to move the cursor in their designated directions on the screen. You will find, however, that the UP-ARROW key does not move the cursor upward unless it is preceded by an ESC keystroke. The arrow keys have special significance for on-screen editing as described later in this chapter.

The Apple IIe features two highly unique keys that are marked

with images of an apple. The one at the left side of the SPACE BAR is not shaded, and it is often called the OPEN-APPLE key. The second apple key, located at the right side of the SPACE BAR is colored white, and is known as the SOLID-APPLE key.

Depressing the apple keys alone has the same effect as depressing the pushbuttons on the game paddles. The OPENAPPLE key simulates the pushbutton on game-paddle #0, and the SOLID-APPLE key simulates the pushbutton on game-paddle #1. The apple keys serve a different function, however, when they are used in conjunction with a CONTROL-RESET operation.

Holding down the OPEN-APPLE and CONTROL keys while striking the RESET key restarts the entire Apple IIe system. The overall action is identical to starting up the system from scratch by turning on the POWER switch. If a disk drive is installed and a properly formatted disk is inserted into it, doing the OPEN-APPLE-CONTROL-RESET operation reboots DOS (and consequently wipes out any BASIC programming that might be stored in memory at the time).

A SOLID-APPLE-CONTROL-RESET operation invokes the Apple's self-testing sequence as described earIIer in this chapter. When the KERNEL OK message appears (hopefully), executing a CONTROL-RESET usually restarts the system for you. If that fails, perhaps because of the nature of the initialization program on the current disk, execute the OPEN -APPLE-CONTROL-R ES ET operation.

In spite of the fact that the DELETE key carries a name that seems to be self-explanatory, it serves no real purpose outside programs that are specifically written to use it.

CURSOR MOVEMENT AND EDITING FEATURES

Cursor motion and program editing go hand-in-hand. One is virtually useless without the other. The following discussions first describe some ways to move the cursor to any desired place on the screen. The discussion then deals with actual editing procedures.

Moving the Cursor

There are three different ways to go about moving the cursor on the display screen of an Apple IIe. Two are carryovers from the

earlier Apple II and Apple II+ systems, and the other is new to the Apple IIe. In any event, all are ESCape functions. That is, an ESC keystroke must precede some other keystrokes in some fashion.

The A-B-C-D Method - With the Apple in its command mode (prompt and cursor symbols showing), do an ESC/A keyboard operation: strike the ESC key, immediately followed by striking the A key. That moves the cursor one column location to the right. By the same token, ESC/B, ESC/C and ESC/D operations move the cursor one position to the left, down and up, respectively.

ESC/A moves the cursor one column to the right. ESC/B moves the cursor one column to the left. ESC/C moves the cursor one line downward. ESC/D moves the cursor one line upward.

The scheme is handy for moving the cursor just a couple of locations, but it is terribly awkward for moving the cursor much farther than that.

The I-J-K-M Method - The I-J-K-M method is easier to use in two respects. First, you only have to strike the ESC key one time in order to enter the cursor-motion mode. Getting out of that cursor-motion mode is a simple matter of striking the SPACE BAR one time.

The second advantage is that these cursor-motion keys are selected so that their physical locations on the keyboard match the direction of motion. Look at the 1, J, K, and M keys on the keyboard, and you will see that the I key is at the top of the group, the J key is at the bottom, the K key is to the right of the others, and the J key is to the left. There is a direct correspondence between these relative locations and the direction of cursor motion.

After striking the ESC key:

K moves the cursor to the right.
J moves the cursor to the left.
M moves the cursor downward.
I moves the cursor upward.

The auto-repeat feature of the Apple IIe comes in handy here. Strike the ESC key to initiate this cursor-motion mode, then hold

down one cursor-motion key or the other until the cursor increments to the desired screen position.

Remember to strike the SPACE BAR to end this operating mode and allow I-J-K-M keys to print their respective characters once again.

The Arrow-Key Method - Moving the cursor by operating the four arrow keys is technically identical to the I-J-K-M method, but it is easier to use-the arrows point to the direction of cursor movement.

You can actually move the cursor to the left, right and downward by striking or holding down the corresponding arrow key at any time. But if you want to apply the UP-ARROW key in this fashion, you must first strike the ESC key. Technically, moving the cursor by working the four arrow keys is regarded as an ESCape operation.

Editing Entire Lines of Programming

The Apple IIe uses some BASIC-programming editing features that are common to virtually all personal computers: re-entering a whole line of existing programming, adding a whole new line of programming, and deleting an entire line of programming.

Re-entering a Whole Line of Programming - Suppose that you have already entered a program that looks like this:

```
10 FOR N=O TO 9

20 PRINT "GOOD MORNING, MR. PHELPS"

30 NEXT N
```

But you now decide that you want the message in line 20 to be extensively revised to read: GOOD AFTERNOON, MY DEAR. Because the revision involves most of the characters in the line, there is little point in attempting to use live-screen editing techniques. It is much easier, and equally effective, in this instance to retype the entire line.

Either execute a HOME command to clear the screen, LIST the program or use the cursor-motion keys to locate the cursor at a clear place on the screen, and simply retype the line:

```
20 PRINT "GOOD AFTERNOON, MY DEAR"
```

That sort of operation, in effect, tells the computer to type over anything listed as program line 20. If you want to confirm that the change has indeed taken place, LIST the program again. You should now see this version:

```
10 FOR N=O TO 9
20 PRINT “GOOD AFTERNOON, MY DEAR”
30 NEXT N
```

Retyping any given line of BASIC programming replaces the older version with the new version.

Adding New Lines of Programming - Adding a new line of BASIC programming is a matter of typing in the new line, using a line number that will ultimately fix it into the desired place in the programming. The successful application of this technique assumes, of course, that your original program was written such that there are line numbers available between successive lines of programming.

Suppose that a portion of an existing program looks like this:

```
100 INPUT “ENTER TWO NUMBERS: “;X,Y
110 PRINT X+Y
```

After testing the routine, however, you decide that it is important to insert an additional PRINT statement between lines 100 and 110 that offers a somewhat more meaningful presentation of the results. Perhaps you want the program to look like this:

```
100 INPUT “ENTER TWO NUMBERS: “; X,Y
102 HOME
104 PRINT “THE SUM OF "X" AND "Y" IS: “;
110 PRINT X+Y
```

Inserting line 102 and 104 is a simple matter of entering them as new lines of programming.

If you happen to be in the midst of entering a new line and you change your mind, doing a CONTROL-X key operation aborts the revision. Alternatively, you can use the left-arrow key to move the cursor to the first digit in the line number, and then strike the RETURN key. Or simply strike the RETURN key in the midst of the entry, and then type the line number and strike the RETURN key.

Deleting an Entire Line - Deleting an entire line of BASIC programming is the simplest editing procedure of all: just type the line number and strike the RETURN key.

In instances where you might want to delete two or more lines in succession, BASIC's DEL command can be quite helpful. The general syntax of that command is:

DEL *startno, lastno*

where *startno is* the first line number in the group to be deleted, and *lastno is* the last line in that group. Thus a command such as:

DEL 120,300

deletes all BASIC programming between lines 120 and 300, inclusively. The two arguments, *startno* and *lastno,* must be line numbers that actually exist in the current program.

Editing Characters Before Striking RETURN

Most programmers make typing errors or other kinds of entry errors while entering a line of programming. Correcting such errors before committing the line to RAM by striking the RETURN key is a rather straightforward procedure.

Suppose that you are in the process of entering a new line of BASIC programming and, before striking the RETURN key, you find that you've typed one character incorrectly. All you have to do in that instance is use the arrow keys to position the cursor over the character that is to be changed, and then overstrike it with the correct character.

You can change any number of characters in that fashion; but when you are done, it is absolutely necessary that you move the cursor one column beyond the end of the program line before striking the RETURN key. Otherwise, all text to the right of the cursor will be lost.

Unfortunately, it is not possible to insert or delete text within a line of programming prior to striking the RETURN key. Such operations must be handled after the line-errors and all-has been committed to the system's memory.

Whenever a new line requires extensive insertions or deletions, it is usually simpler to re-enter it from scratch. The Apple

IIe, however, allows you to make after-the-fact insertions and deletions by using the ESC-editing features.

ESCape-Editing Techniques

Striking the ESC (ESCape) key allows you to use the arrow keys to position the cursor without copying any text to program memory. For the purposes of the present discussion, I am calling this situation the *no-copy* mode. Once you have moved the cursor under the *no-copy* mode, you can return to the normal copy mode by striking the **SPACE BAR.** Thus the ESC key, the four arrow keys, and the SPACE BAR can work together to perform some powerful editing tasks for program lines that have already been entered into memory.

Although the following instructions might seem rather complicated at first, just bear in mind that they are all based on the principles just cited. If you understand the principles, you don't have to memorize the procedures: enter the *no-copy* mode by striking the ESC key, and return to the normal copy mode by striking the **SPACE BAR.** And as long as the system is in the copy mode, characters falling under the cursor are copied to the program memory; and in the *no-copy* mode, characters passing under the cursor are ignored.

Incidentally, one good way to determine whether or not you are operating in the *no-copy* mode is by striking the UP-ARROW key; the cursor responds by moving upward only when the system is in the *no-copy* mode.

The general procedure is to LIST the line that is to be edited, enter the *no-copy* mode to position the cursor at the beginning of the line (over the first digit in the line number), then use the copy and *no-copy* modes to change single characters, insert new programming, delete some programming, and even copy entire lines of programming with different line numbers.

Changing Characters - Changing characters within a program line is a rather straightforward procedure if no insertions or deletions are involved.

• LIST the line to be edited.

• Strike ESC to enter the *no-copy* mode, and use the arrow keys to position the cursor over the first digit in the line number.

• Strike the SPACE BAR to return to the normal copy mode.

A. Use the RIGHT-ARROW key to move the cursor to the right and copy text to memory as it goes along. (If the program line happens to be so long that it wraps around to the beginning of another line on the screen, you must use the *no-copy* mode again as described later.)

B. Position the cursor over the character to be changed, and overstrike it with the desired character.

C. Use the RIGHT-ARROW key to run the cursor just past the end of the program line (to make sure it is all copied into memory).

• Strike the RETURN key to mark the end of the line and, indeed, the editing operation.

You can use steps A and B for overstriking any number of characters in a program line, moving the cursor to the left or right as necessary. Just remember to run the cursor all the way past the end of the line before striking RETURN.

If it happens that a program line is so long that it is continued on a lower line on the screen, you must enter the *no-copy* mode before moving the cursor to the beginning of the next screen line. Failing to do that, the spaces that the Apple IIe inserts between the end of one line and the beginning of the next are copied quite literally into the edited version of the line.

So when you are editing in the copy mode and reach the end of a screen line, move one space past the last character, strike the ESC key to enter the *no-copy* mode, hold down the RIGHTARROW key until the cursor is positioned over the beginning of that next screen line, and then return to the normal copy mode by striking the SPACE BAR.

Deleting Segments of a Line - Being able to switch between the copy and the *no-copy* modes makes it rather easy to delete sections of a given line of BASIC programming. The general idea is to use the copy mode and arrow keys to save sections of the line, and go into the *no-copy* mode while moving the cursor through segments that are to be deleted.

• LIST the line to be edited.

• Strike ESC to enter the *no-copy* mode, and use the arrow keys to position the cursor over the first digit in the line number.

- Strike the SPACE BAR to return to the normal copy mode.
  A. Use the RIGHT-ARROW key to move the cursor to the right and copy text to memory as it goes along. Be sure to do an ESC/SPACE BAR sequence when you must drop from the end of one screen line to the beginning of the next.

  B. Position the cursor over the first character to be deleted.
- Strike the ESC key to get into the *no-copy* mode, and use the RIGHT-ARROW key to move over the text to be deleted; stop when the cursor is one column past the text to be deleted.

- Strike the SPACE BAR to return to the copy mode, and use the RIGHT-ARROW key to move and copy the remainder of the line. Again, be sure to enter the *no-copy* mode whenever it is necessary to move to the beginning of a lower line on the screen.

- Strike the RETURN key to mark the end of the program line and the editing operation for it.

Inserting New Text into a Line - Bearing in mind that you have access to copy and *no-copy* modes, the procedure for inserting new characters into a line of BASIC programming can be mastered with a bit of practice.

- LIST the line to be edited.

- Strike ESC to enter the *no-copy* mode, and use the arrow keys to position the cursor over the first digit in the line number.

- Strike the SPACE BAR to return to the normal copy mode.

  A. Use the RIGHT-ARROW key to move the cursor to the right and copy text to memory as it goes along. (Use an ESC/SPACE BAR sequence whenever it is necessary to drop from the end of one screen line to the beginning of the next.
  B. Position the cursor over the place where the insertion is to begin.

- Strike the ESC key to get into the *no-copy* mode, and use the arrow keys to position the cursor at a clean section of the screen, or to the beginning of some text that is to be inserted.

- Strike the SPACE BAR to return to the copy mode, and type the text to be inserted, or use the RIGHT-ARROW key to move and copy text that might already be printed on the screen.

- Strike the ESC key to get into the *no-copy* mode, and use the arrow keys to return the cursor to the place where the material is to be inserted.

• Strike the SPACE BAR to return to the copy mode, and use the RIGHT-ARROW key to move and copy the remainder of the line. And as usual, be sure to enter the *no-copy* mode whenever it is necessary to move to the beginning of a lower line on the screen.

• Strike the RETURN key to mark the end of the program line and the editing operation for it.

Copying Entire Lines of Programming - The ability to copy text material is not limited to the actual content of BASIC program lines; it applies to the program line numbers as well. Here is an example of a programming routine that does not have to be entered in a tedious, character-by-character fashion.

Listing 1-1

```
10   HOME:PRINT
20   PRINT "SELECT ONE:"
30   PRINT TAB(5) “1 -- START FROM THE BEGINNING
40   PRINT TAB(5) “2 -- START FROM NEAR THE BEGINNING
50   PRINT TAB(5) “3 -- START FROM THE MIDDLE
60   PRINT TAB(5) “4 -- START FROM NEAR THE END
70   PRINT TAB(5) “5 -- FORGET THE WHOLE THING
80   PRINT:INTPUT X
```

With the help of the *copy* and *no-copy* editing feature, you can save a lot of typing for lines 30 through 70.

As far as this example is concerned, begin by entering line 30. Then strike the ESC key to get into the *no-copy* mode, and use the arrow keys to place the cursor over the 3 in the line number. Strike the SPACE BAR to get into the copy mode, and overstrike the 3 with a 4. Then copy material to the right, overstriking the characters that are to be changed in order to compose line 40 of the program. Strike the RETURN key when you reach the end of the line, and then try doing a LIST to prove that lines 30 and 40 both appear in the program. Repeat that general procedure for the remaining lines that have that same PRINT format.

You can multiply similar lines of BASIC programming rather quickly with that technique.

SAVING AND LOADING PROGRAMS FROM CASSETTE TAPE

An ordinary cassette tape recorder/player that is equipped with a tape-counter mechanism offers the most economical means for saving and loading BASIC programs and other kinds of computer information. When saving programs or data by means of a cassette tape recorder, the Apple IIe converts the information into audio tones that can be easily recorded on magnetic tape; specifically on the narrow tape in common audio tape cassettes. And when loading previously saved information, the tape player reproduces the audio tones, and the computer converts them back into meaningful computer data for the system's RAM (Random Access Memory).

It is possible, and certainly economically attractive, to save more than one program on a single cassette. The only problem is being able to find the segment of tape that contains the desired program. That is the reason for specifying a cassette tape recorder/player that has a tape-counter on it.

Connecting the Cassette Recorder/Player to the Apple

Two conductor-and-plug assemblies are required to interconnect the Apple IIe with a standard cassette recorder/player. On the rear, right-hand side of the console are two subminiature jacks that are labeled with little white arrows. The upwardpointing arrow indicates the CASSETTE-OUT connection-a connection that is to go to the RECORD IN jack on the cassette recorder. The downward-pointing arrow indicates the CASSETTE-IN connection, and it should be connected through a cable to the EARPHONE jack on the recorder.

If the appropriate cable-and-plug assembly is not supplied with your cassette machine, equivalent versions are available through most electronics hobby and audio stores.

Adjusting the Cassette Volume Level

Because there are so many different makes and models of cassette recorder/player machines, it is impossible to specify exactly what volume level is appropriate for recording and, especially, loading programs from cassette tape. The adjustment has

to be made by trial-and-error, and the best place to start is with a commercially prepared Apple program tape.

Obtain a commercially prepared Apple program tape (tapes made for the Apple 11 and Apple II+ work quite well, too), insert it into the tape player and make sure that it is rewound to the beginning. Set the tape counter to zero, temporarily remove the plug to the EARPHONE jack, and PLAY the tape until you hear the beginning of a steady whistling sound. Stop the tape immediately, make a note of the tape-counter reading for future reference, and re-insert the plug into the EARPHONE jack.

Set the volume control on the cassette machine to a rather low reading, type LOAD at the keyboard, begin PLAYing the tape and then strike the computer's RETURN key.

If things are going well, you will hear two beeps from the console loudspeaker. The first indicates that data is being loaded into the computer, and the second signals the end of the loading procedure. That is the time to turn off the cassette machine. If, indeed, things happen that way, it means that you have found the correct volume-level setting. Make a note of that level for future reference.

Changes are quite good, however, that you won't select the correct volume level the first time. In that case, you might not hear one or either of the beep tones, or perhaps you will see an error message printed on the screen. If it appears that nothing is happening after a reasonable length of time, turn off the cassette machine and reset the Apple by executing a CTRL-RESET operation, followed by a CTRL-B and RETURN.

Set the volume control a bit higher, rewind the tape to the beginning of the program (as indicated by your earIIer notation of the tape-counter reading), and try the LOAD command again.

Repeat that procedure, setting the volume control a bit higher each time, until you either get a successful load as described earIIer or this sort of message:

///SYNTAX ERR(OR)

That message indicates that the computer is receiving data from the tape, but that it is garbled. In that case, make sure that you are actually starting the tape from the beginning of the program's leader tone.

Once you can load programs successfully from cassette tape, you can be confident that the volume level is correct for recording programs.

Saving Programs on Cassette Tape

Saving your own BASIC programs for future use is a vital function for any serious programmer. And if the storage medium happens to be cassette tape, the procedure is quite simple (assuming that the volume level on the cassette recorder/player is properly adjusted as described in the previous discussion). The BASIC command for saving programs on tape is SAVE.

First, find the end of the last-recorded program on the cassette tape. If you have been making a note of the tape-counter readings, you should have no difficulty finding that place. Confirm your selection by temporarily removing the plug from the EARPHONE jack and listening directly to the tape. If you hear tones, whistles and chirps, you are not yet at the end of a previously recorded program. A faint hissing sound marks the blank area.

Assuming that a BASIC program is present in the Apple's memory, note the tape-counter reading, type SAVE at the keyboard, start the cassette in its record mode, and then strike the RETURN key to execute the SAVE command.

About 15 seconds later, you should hear a single beep tone from the console loudspeaker that is followed by an interval of time that is required for loading the programming, itself. A second beep indicates that the recording task is done. Turn off the recorder and write down the tape-counter reading for future reference-so that you will know where to begin recording the next program you want to save.

Loading Programs From Cassette Tape

Assuming that you have already adjusted the cassette recorder/player volume level as described earIIer, the BASIC command for loading a program from tape is LOAD. In fact, the recommended procedure for adjusting the volume level incorporates the program-loading operations.

First, find the beginning of the leader tone for the program you want to load into the Apple IIe. That is done by consulting your written records of tape-counter readings and confirming the mat-

ter by temporarily removing the plug from the EARPHONE jack and listening for the beginning of the steady leader tone.

Once you've found the beginning of the leader tone, reinsert the EARPHONE jack, type LOAD at the keyboard, start the cassette machine in its playback mode, and then strike the RETURN key. The first beep from the console loudspeaker signals that the loading operation is beginning. After some time--when the loading operation is done-you will hear a second beep tone and see BASIC's prompt and cursor symbols reappear on the screen. That is the time to turn off the tape recorder and execute a RUN command to begin running the program.

SAVING AND LOADING PROGRAMS FROM A DISK

The disk operating system is far easier to use than a cassettemachine system. Not only do the saving and loading operations take place faster, but it is much easier to access and keep track of the programs,

Formatting a Disk

As mentioned several times earlier in this chapter, the Disk Operating System (DOS) cannot operate at all without access to a properly formatted disk. The DOS 3.3 diskette supplied with the Apple IIe product is properly formatted, and it represents the only means for getting DOS booted and working in instances where a user has no previously formatted disks available. The purpose of this section is to describe how to format your own disks for the sake of saving and loading custom programs.

With DOS properly booted, insert a new diskette into disk drive #1. Execute a NEW command to make sure that no other BASIC programming is in memory, and then enter this short BASIC program:

```
10      HOME
20      PRINT " MY DISK #1"
30      PRINT:PRINT "(MAKE CERTAIN THAT CAPS LOCK IS DOWN)"
```

The exact nature of the program isn't relevant. The only

requirements are that it be written in BASIC and include at least one valid line of programming.

The next step is the important one. Enter this command from the keyboard:

```
INIT HELLO
```

The moment you strike the RETURN key, the disk drive begins running; and it runs for quite some time. DOS is copying itself to the new disk and setting up the proper sector and directory format. When the task is completed, BASIC's prompt and cursor symbols reappear on the screen.

The important command is INIT; the HELLO is simply a name for the DOS-initialization program that is used almost universally by Apple users. You can name it anything you wish, however.

Saving Programs on Disk

After you have typed a BASIC program into the Apple IIe, you can save that program on disk for future use by executing this general command:

SAVE *filename*

where *filename is* some name that you've chosen to give the program.

Filenames can be up to 30 characters long. They must begin with a letter of the alphabet, however, and they must not include commas. It is a good practice to make up filenames that have some meaning as far as the purpose of the program is concerned, yet it is wise to keep those names as short as possible. Here are some valid filenames:

GOODSTUFF V. 1

GAME 41

BETAGEN 10/26/84

So if you want to save a program under the filename such as XODD-1, then the appropriate command is:

```
SAVE XODD-1
```

Saving a program under a filename that already exists on the

disk will cause the new version to write over the old one. There is no warning from the computer, so make sure that you do not use duplicate filenames; (unless, of course, you really do want to replace the old version with the new one).

Viewing the Disk Files

Any disk that is properly formatted will contain at least one BASIC program-the initialization program that Apple users normally name HELLO. Whenever you want to see that filename and any others that happen to represent programs saved on a disk, type this simple command:

CATALOG

Having done that, the computer will respond by printing out all of the filenames on the disk.

Here is a sample of a typical disk CATALOG printout:

```
DISK VOLUME 254

 A 002 HELLO
 A 006 FUNSTUFF
*A 014 SLICK.1
 I 024 SETUP
 B 004 PRINTIT
 T 024 NAMES 8-8-63
```

That example shows that there are six different programs, or files, on the current disk. A single letter, possibly an asterisk, and a 3-digit number precedes each of the file names. The computer inserts that information; it is not part of the filename.

The letters at the beginning of each catalog listing indicate the type of program or file:

A - Applesoft BASIC program

I - Apple Integer BASIC program

B - Binary, or machine-language, program

T - Data file

An asterisk preceding any of those file-type letters indicates that the file has been locked so that it cannot be accidentally

erased or changed in any manner. Chapter 7 describes how to lock and unlock files.

The 3-digit numbers indicate the number of disk sectors that are devoted to the program or file; and, generally speaking, summing those numbers can lead to a figure that represents the amount of room remaining on the disk. The maximum number of sectors is 496, so subtracting the sum of the number of sectors for the program from 496 leads to the number of unused sectors.

NOTE: The maximum capacity of a disk is 496 sectors.

There is one catch to this notion of calculating the amount of space remaining on a disk, however: whenever the number of sectors for a program or file exceeds 255, the numbering starts over from 0. A very long program or file, then, might show a sector figure of 10; when, in reality, it occupies 265 sectors.

In any event, the system prevents you from overfilling a disk. Whenever you attempt to save a new program and there is insufficient disk space, the computer displays a DISK FULL error message. In that case, you have the option of deleting some files or programs that you no longer need, or inserting another disk that has sufficient space remaining on it.

The DISK VOLUME message at the beginning of a CATALOG listing is the volume number of that particular diskette. Unless you specify otherwise when initializing the disk, the volume number takes on the largest-allowable value-254.

If you wish to assign a different volume number to a disk, it must be done when that disk is initialized. The general procedure in that case is to execute a command of this form:

INIT filename,Vvol

where *filename is* the name of the initialization program usually HELLO-and vol is the desired disk-volume number. So executing the following command will initialize a disk with HELLO, and give it a volume number of 1:

INIT HELLO,V1

Executing the CATALOG command then shows this sort of display:

```
DISK VOLUME 001
A 002 HELLO
```

Loading Programs From Disk

Loading a BASIC program into the computer from a disk is a simple matter of executing this command:

LOAD *filename*

where *filename is* the name of a program, either Applesoft or Integer BASIC, that exists on the current disk. Assuming that the file is found, the computer loads the program and replaces any other programming that might have been in the system at the time.

Alternatively, it is sometimes convenient to load programs with this command:

RUN *filename*

That command not only loads a program called *filename,* but begins executing it as well.

If the specified filename is not on the current disk, the computer responds by printing a FILE NOT FOUND message. And if the program was originally written and saved in Integer BASIC, and you have not loaded Integer BASIC into the computer, the system responds by printing LANGUAGE NOT AVAILABLE.

Deleting Disk Files

Erasing a program or file that already exists on a disk is a matter of executing this command:

DELETE *filename*

where *filename is* the name of the program or file to be erased.

If you have previously locked the file, the computer cannot erase the file; instead, it shows a FILE LOCKED message. The file must be unlocked (Chapter 7) before it can be deleted.

Summary of Disk Error Messages

Automatically generated disk error messages can be annoying at times, but they exist to protect the user from taking some potentially devastating mistakes. The following summary of disk error messages deals only with those that are relevant to the discussions in this chapter. Chapter 7 extends the list to include errors that can occur when working with DOS at a more sophisticated level.

LANGUAGE NOT AVAILABLE - This error message appears whenever you attempt to load a disk program when its language is not available. Applesoft BASIC is always available, because it is built into the Apple IIe system. Integer BASIC, on the other hand, must be loaded into the system as a program, itself. So this error does not occur when loading programs that were originally written and saved in Applesoft BASIC.

Integer BASIC, on the other hand, must be loaded into the system before it can be used. And the LANGUAGE NOT AVAILABLE error occurs when attempting to load an Integer BASIC program without having first loaded that language into the system.

WRITE PROTECTED - That error message appears on the screen whenever you attempt to initialize, write on, delete or otherwise alter the content of a disk that is mechanically writeprotected.

Most diskettes have a small rectangular slot cut into the righthand side. If that slot does not exist, or if it is covered with an adhesive tab, the disk is mechanically wr ite-protected. Removing the tab to expose the slot lets you carry out normal writing operations.

FILE NOT FOUND - This error occurs when you attempt to load or delete a file and your specified *filename is* not found on the disk.

I/0 ERROR - Seeing this error message tells you that the system cannot work with the current disk at all. Typically, it occurs when the disk-drive door is not properly closed, and when the current disk is not properly initialized.

DISK FULL - This error message appears on the screen whenever you attempt to save a program or file, and there is insufficient space remaining on the disk. The options are to make room by deleting some unnecessary files or to insert a different disk that does have sufficient space remaining.

FILE LOCKED - This error condition is similar to the WRITE PROTECTED error, but it applies to selected files rather than the entire disk. A CATALOG listing of the disk's contents will show an asterisk for files that are locked in this fashion. See Chapter 7 for further information about locking and unlocking disk files.

FILE TYPE MISMATCH - The basic Apple IIe system works with four kinds of files: Applesoft programs (A), Integer BASIC programs (1), machine-language programs (B), and text/data files (T). A file-type mismatch occurs when you attempt to load a machinelanguage program or text/data file as though it were a BASIC program. Or, conversely, the error occurs when attempting to read a BASIC program as though it were machine-lan