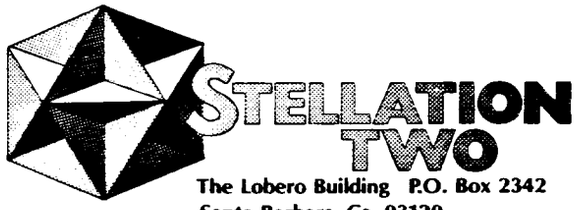
OS/9 BASIC Ø9 DOCUMENTATION

The OS/9 Operating System with Basic Ø9 adapted for the Apple II Personal Computer

INSTALLATION AND OPERATION GUIDE



The Lobero Building P.O. Box 2342
Santa Barbara, Ca. 93120

"OS/9" and "BASIC Ø9" are trademarks of Microware Systems Corp.

STELLATION TWO CONSUMER PRODUCTS REGISTRATION INFORMATION

Please fill out the enclosed registration card and return it to us so that we may provide you with information about updates and about new products. The serial number requested on the card is the number printed on the corner of THE MILL.

[redacted] WARRANTY

Stellation Two/Microware warrants to the original user of this product that the physical media shall be free of defects resulting from faulty manufacture of the product or its components for a period of thirty (30) days from the date of sale. STELLATION TWO/MICROWARE MAKES NO WARRANTIES REGARDING EITHER THE SATISFACTORY PERFORMANCE (e.g., MERCHANTABILITY) OF THE SOFTWARE ENCODED ON THIS PRODUCT OR THE FITNESS OF THE SOFTWARE FOR ANY PARTICULAR PURPOSE. Defects covered by this Warranty shall be corrected either by repair or, at Stellation Two's election, by replacement. In the event of replacement, the replacement unit will be warranted for the remainder of the original thirty (30) day period or 10 days, whichever is longer.

Warranty/Service Information

If your OS/9 BASIC Ø9 requires replacement, please return it to the dealer from whom it was purchased. If it is not possible to return the disk to your dealer, you may send it directly to Stellation Two with proof of purchase.

During warranty, we will replace or repair your OS/9 BASIC Ø9 Software without charge.

To return any Stellation Two product for service, please mail it post-paid. Package the card securely as we cannot be responsible for damage due to shipping. BE SURE TO enclose proof of purchase for warranty work.

Mail post-paid to:

Stellation Two
P.O. Box 2342
Santa Barbara, CA 93120

REGARDING THIS MANUAL

This manual is written in the popular "speak to the reader" fashion rather than the traditional third person. We have attempted to present the material in an easily understood fashion, considering a wide range of readers' skill levels. Your critique will be appreciated.

Conejo Computer Products
3655 Thousand Oaks Boulevard, Suite 255
Westlake Village, CA 91362

The OS/9 User's Guide is your primary source of "how-to" information for the general OS/9 features, whereas this manual is primarily intended to provide supplementary information on features unique to the APPLE II adaptation of OS/9. You will benefit by reading the first few chapters of both before embarking on your first session with OS/9; in particular, it is important that you understand the concepts of the "tree-structured file directories" and the execution and data directories. Please disregard the discussion of the FORMAT command in the User's Guide for creating a backup disk and, instead, refer to the procedure discussed in Chapter 1.

FOR TECHNICAL ASSISTANCE WITH PROBLEMS

Please attempt to resolve the problem with your dealer's help. Should the need arise, contact Stellation Two for further assistance.

OPERATION

WELCOME to the world of OS/9—the most powerful and convenient programming and computing environment ever made available for your APPLE II.

Enclosed with your new software is an OS/9 daughterboard. The daughterboard fits onto your MILL and serves as an address mapping device to reorganize APPLE memory space to conform to OS/9 requirements.

Note: To avoid bending pins or causing personal 'pin-prick' injury, please use enclosed tweezers to remove chips.

DIRECTIONS ON INSTALLATION OF YOUR OS/9 DAUGHTERBOARD

1. Replace Chip #1 on diagram (74LS14) with your new OS/9 chip (74LS04). Make certain that the small notch on the chip is facing upwards in the same direction as all the other chips on the board.

2. Remove Chip #2 (74LS367), located at the bottom right of the MIII.

3. Remove the green backing from the mounting tape adhered to the back of the daughterboard.

4. Insert daughterboard pins into lower right corner socket of the MIII (where Chip #2 was originally in place). Press mounting tape to the MIII so that the daughterboard stays securely in place.

Your OS/9 assemblying is now complete. All that remains for you to do is put the MIII in its slot (any slot except Ø) and activate the switch.

As you face the keyboard of your APPLE II, the daughterboard switch pulled towards you will activate the Pascal Speed-Up Kit and the Assembler APPLE DOS Environment. Pushed away from you, towards the back of your computer, the switch activates the OS/9 mode. When it is necessary for you to flip the switch, we suggest you turn off the power and, as a precautionary measure, use the eraser end of a pencil.

OVERVIEW OF THE APPLE ADAPTATION OF OS/9

Developed by Microware Systems Corp. for 6809 computers, OS/9 was written to exploit the advanced capabilities of the Motorola MC6809 Microprocessor, yielding a truly unique and innovative operating system. The BASIC Ø9 language likewise was developed to provide an extremely powerful and, moreover, convenient language which was previously impractical with the older, less capable microprocessors. The advanced capabilities offered by OS/9 are yielding an increasing set of applications software which is highly modular and extensible.

THE MILL 6809 processor board made possible the marriage of this set of excellent software to the Apple II, resulting in a very large improvement of the capabilities of this popular and inexpensive computer. The adaptation of OS/9 to the Apple II offers several features beyond those intrinsic to OS/9 itself:

- Modern multi-tasking operating system
- Multiple file directories simplify data organization
- 6809 performs disk I/O, freeing the 6502 for I/O work
- 6502 operates as an independent I/O processor, controlling Apple compatible devices with or without interrupt capability
- All I/O is interrupt driven for optimal CPU utilization and operator conveniences such as type-ahead and printer spooling "in the background"
- Time of day clock for multi-progromming and file a creation and modification date logging

- Inter-processor interrupt for obtaining exclusive use of the memory/bus for time-critical operations
- Memory optimally reorganized for OS/9 and for 16K RAM card.
- Modular software with extensive error checking
- Capability for user-written peripheral device drivers

CHAPTER TOPICS

CHAPTER I

GETTING STARTED

RUNNING OS/9 FOR THE FIRST TIME

Please follow these steps to "boot" OS/9 for the first time:

1. Obtain a blank diskette on which a backup copy of the OS/9 system disk will be made.

2. Boot-up APPLE DOS 3.3 and run the "COPYA" program. Answer the prompts from it and copy your OS/9 system disk to the blank disk. Label the new disk "OS/9 v0 System Disk" and store the original in a safe place.

3. Using the Introductory Instructions in the Operations section, install the "remapping ROM daughterboard" on THE MILL board.

   Note: If you purchased THE MILL and OS/9 simultaneously, the daughterboard will already be installed.

4. With the APPLE's power off, install THE MILL in any card slot except zero (OS/9 automatically determines the slot).

5. Verify that the disk controller card for the DISK II drive(s) is in slot six.

6. Place the OS/9 System disk into the first drive and turn on the APPLE's power.

7. The message "OS/9 BOOT V1" should appear on the normal 40 column display. In a few seconds, "48K" or "64K" will appear in the upper left corner of the screen. The latter will appear only if your computer has a Language Card or other 16K RAM card installed.

8. After a few seconds, a flashing cursor will appear. This indicates that OS/9 has been successfully read from the disk into memory and is performing the initialization procedure contained in the file "STARTUP."

9. This procedure displays a brief greeting message enclosed by "*"s, and runs a utility program which displays the memory utilization.

10. Lastly, the message "SHELL" will appear and the command prompt: "OS9:" including cursor.

If all of the above steps were successful, skip the next section of troubleshooting.

WHAT TO DO IF OS/9 WILL NOT BOOT

The causes of a failure to boot are listed below in order of likelihood, most probable first:

1. The daughter board may be incorrectly installed or the I.C. "chips" on it are reversed or backwards. Double check the setting of the switch. If you insert the I.C. chips incorrectly, there is a small chance that the chip will have been permanently damaged and will need to be replaced. Also, check that THE MILL is properly seated in the card slot connector.

2. The disk may not have read properly. Repeat the COPYA program using the original disk, but reverse the drives this time.

3. Perhaps there is an interface board installed which is mecahnically or electrically interfering. Do a visual check for boards touching each other. If this is not the case, remove all boards from the APPLE except THE MILL and the disk controller, then try the boot again. Lastly, remove the RAM or Language Card and restore the memory chip (if any) which was removed when that card was installed (this reverts to a stock 48k machine). Try the boot again.

4. We now may suspect that THE MILL or the daughter board is defective. Do a thorough visual check for loose chips or components which may have been broken in shipment. A good proof of the health of the board is to remove the daughter board, replace the chip in the board whic was relocated to the daughter board, and via APPLE DOS, run one of the demonstration programs for THE MILL which are available at your dealer. If you have purchased the PASCAL SPEED-UP KIT or the Assembler, try running them. If the board runs properly now, we know that the daughter board is defective or was installed incorrectly.

5. If none of the above solves the failure to boot, please contact your dealer for assistance. Stellation Two will gladly help in the event the problem cannot be solved by substitution of known-good components.

GETTING ACQUAINTED WITH OS/9

If you are familiar with APPLE DOS, OS/9 will require a little time for you to get oriented to the philosophy of this new "boss" for your computer hardware, the operating system. OS/9 is quite generalized and, once the initial familiarization is accomplished, you will be delighted with the greater freedom now available. This section contains an overview of OS/9 as adapted for the APPLE II and a short walk-through of your first few commands to OS/9.

The OS/9 system is a modern operating system with capabilities usually found in larger mini-computer systems. The supplied User's Guide manual will give you a good overview of the concepts involved in the multiple file directories and the generalized input-output methods. You will benefit by reading the first few sections of the User's guide before attempting to use OS/9. After finishing this, follow along with the walk-through described below.

After booting OS/9, the "OS9:" prompt appears on the screen. This prompt comes from a program called "SHELL" which responds to your commands to run other programs. You enter commands in response to the OS9: prompt.

REGARDING DISKETTES

This page should be skipped on first reading. The information here is required when you have booted and run OS/9 a few times using a verbatim copy of the OS/9 system disk (via COPA in APPLE DOS 3.3).

The diskettes used by OS/9 are identical to DOS 3.3 disks in terms of sectors per track, however, the directory and file structures are totally different. Do not use OS/9 disks with Apple DOS 3.3 commands except as a "read-only" disk for COPYA. OS/9 will not work with disks or disk controller for the 13 sector diskettes of DOS3.2 and earlier.

The "FORMAT" command, described in the User's guide, is NOT to be used in this APPLE version of OS/9. Because the APPLE disk does not conform to the defacto standards for disks (IBM 3740), you have been supplied an alternate utility program to prepare diskettes for use by OS/9. This program is named "FORMAT11" in the standard command directory. FORMAT11 requires pre-formatted disks. We suggest that you boot APPLE DOS 3.3 and INIT several diskettes for use with OS/9. The steps to create a blank disk are: (requires two drives)

1. Boot up APPLE DOS 3.3

2. Using blank or scratch diskettes, type in an INIT command for several diskettes.

After your first few exploratory sessions with OS/9, you will need to begin to use these diskettes.

3. To make a diskette usable by OS/9, take a diskette prepared in step 2 and, after booting:

   Place the diskette in the second drive and type:

   FORMAT11 /D1<return>

The FORMAT11 program will prompt you to enter the name to be written on the disk (a "volume" name). Enter an appropriate subject title for the purpose of the diskette (be brief). Write this name on the jacket label of the diskette. Try to keep two or three blank, pre-formatted and FORMAT11'd diskettes available for use on a moment's notice. Mark these disks as "formatted for OS9."

Before we begin to enter commands, let's take a minute to learn how to use the special keyboard features.

SPECIAL KEYS AVAILABLE

The special characters you type to stop the screen scrolling, abort a program, etc., are not fixed keys; you may change them to suite your preference. The utility program "TMODE" (see User's Guide) may be used to redefine one or more keys to be used instead of the predefined (default) keys. In addition to these keys, there are several special keys which allow you to use characters not normally available on the APPLE keyboard. These "substitution" keys are:

control-K	yields a left bracket	
shift-M	yields a right bracket	
control-O	yields an underscore	(note 1)
control-P	yields a vertical bar	(note 1)
control-V	yields a left brace	(note 1)
control-W	yields a right brace	(note 1)
right arrow	yields a backslash	

Note 1: This key useful only with an 80 column display.

The keys used for typographical error correction are:

left-arrow	backspace one column
control-X	delete current line
control-A	recall last line typed in

Other special keys:

control-S	"S" tops screen output—press any non-special key to resume (i.e., space)
control-X	deletes the entire line

control-Q	"Q"uit the current program (return to command mode or to BASIC 09's debugger)
escape	signals end-of-file for the keyboard

These keys may be changed as desired via the TMODE command.

The "substitution" keys mentioned earlier cannot be changed with TMODE.

For now, just use left-arrow to backspace and control-X to delete a fouled-up line.

ENTERING YOUR FIRST FEW COMMANDS

Let's try a command now. Type in:

CHX /DØ/CMDS<cr>

(the <cr> means pressing the "return" key)

This means:

"change the execution directory to drive Ø, directory CMDS"

This command tells OS/9 to select drive Ø (known as D1 in APPLE DOS), and directory "CMDS" on that drive and use this as the "home base" for subsequent commands. Each command you type in is actually the name of a small (executable) program to process the command. Thus, the number of commands available is easily expandable. As we will see later, you may tell OS/9 to load a specific command file into memory and leave it there so that it need not be fetched from disk each time it is used. You will want to do this with frequently used commands. Use the CHX command whenever you need to change the current disk directory for executable programs. You'll also need to enter CHX if you change diskettes and wish OS/9 to use the execution directory on the newly inserted diskette.

In addition to the execution directory, OS/9 allows for a separate "data" directory which is used to store the files you work with which contain data rather than executable programs. To set the current data directory, type:

CHD /DØ<cr>

which sets the data directory to drive Ø and to the highest directory for the drive, or the "root" directory. In this directory are the names of all other directories and, perhaps, the names of some files proper.

Now let's enter a command to view the contents of the current data directory:

DIR E<cr>

which yields a length display. Compare the display to that in the User's Guide explanation of the DIR command. Note the effect of the "E" in the command. To show the contents of the execution directory, we must be explicit and override the "default" data directory which DIR showed us:

DIR E /DØ/CMDS<cr>

This directory contains many commands you will be using later. Try the DIR command yourself, substituting the names indicated by the previous DIR commands to be directories (hint: they have a "D" in the "Attributes" column. With multiple directories you can see that organizing data by subject, job, or programming language is quite easy with OS/9. The idea of the separate data and execution directories is to control access to files and to keep the accounts for Company A separate from Company B (or an apple pie recipe separate from oranges in stock).

NOTE: From here on, we will omit the <cr> reminder in the examples.

Thus far, we have used only the DIR command. Here is a utility command which lists text (ASCII) files on the screen:

LIST /DØ/SYS/ERRMSG

which means:

List drive Ø, directory SYS, file ERRMSG to the screen.

In this case, the file (or pathname) is a text file which contains the system error message look-up table. If you LIST a file containing binary data such as an executable file, the screen will "go nuts" and display garbage. Use the DIR E command and if a file's attributes include an "E", you should not LIST it.

Since the last CHD command specified the root (main) directory on drive Ø as the current directory, we could shorten up the last command to:

```
LIST SYS/ERRMSG
```

or even shorter if we change the default data directory:

```
CHD /DØ/SYS
LIST ERRMSG
```

In each of these commands, the LIST utility program was retrieved from the default execution directory which, via the earlier CHX command, we had set to /DØ/CMDS.

Do you recall that frequently used programs may be "loaded" into memory to avoid having to fetch them from disk each time they are needed? Suppose you plan to use DIR often in the current work session so loading DIR would speed up things. Here is a load command to make DIR memory-resident rather than disk-resident:

```
LOAD DIR
```

This goes to the current execution directory <CMDS> and "locks" the DIR program into memory. Now DIR commands will go directly to the copy of DIR in memory.

To view the program "modules" currently loaded in memory, use the "MDIR" (module directory) command:

```
MDIR E
```


The user's guide explains this command. In the display from this command, you will see the modules internal to OS/9 which are always present after booting. Each of these modules is explained in the System Programmer's manual (available from Stellation Two). Your APPLE OS/9 contains the following modules unique to the APPLE computer:

AACIA	controls keyboard and screen input-output as well as miscellaneous peripheral device interfaces.
DISKII	performs all read/write of diskettes via the 6809. The 6502 processor does no disk I/O.
TERM	device descriptor for the keyboard/screen.

If you have done the LOAD DIR command, you will see the DIR program in the list of memory modules displayed by the MDIR command. It is convenient to LOAD MDIR for most sessions.

Ah, but how do we get a module OUT of memory if it is no longer needed? Easy--UNLINK it. As shown in the display from the MDIR E command, a module has a "link" count which shows how many times it has been "claimed" by the system. You'll need to repeat the UNLINK <name> command if the link count is 2 or greater. An easy way to do this is to type in UNLINK <name> once, then hit control A (recalls the last command) and <cr> repeatedly until the "module not found" error happens. You should not UNLINK a module which is an I/O driver or device descriptor—reboot.

You should now use the CHX, CHD, LIST, DIR, and MDIR commands to get the "feel" of OS/9 and the directories. If you get messages such as "ERROR 216" you are typing in non-existant file names or are in the wrong data directory, or have changed the execution directory with the CHX command and OS/9 cannot find the command you are trying to use. To get back to "square one", type:

```
CHD /DØ       and then
CHX /DØ/CMDS
```

If you are terribly confused, RESET the APPLE and reboot. Pressing the RESET key (some Apples require a simultaneous CTRL key) will instantly stop OS/9 and bring up the 6502 monitor program on the 40 column display. From here, type the usual "6 ctrl-P" to reboot.

You will notice that error messages are output as numbers. These numbers are listed in the back of the User's Guide, cross-referenced to the english explanation. If you wish to have this explanation displayed on each error, type the command:

PRINTERR

This tells OS/9 to use the error number to find the appropriate english message in a file named "SYS/ERRMSG" which is on drive Ø. After using OS/9 a while, you will be able to recall the half-dozen or so common error numbers and will not need to wait for the english message conversion. With experience, you will find that YOU may change the ERRMSG file to display your error message instead of the usual one. Ths is useful for "turn-key" systems which need different human factors than those found in a development system. Note: To "undo" the PRINTERR command, you must reboot the system.

Now would be a good time to read about the TMODE command and experiment with it. Try turning on the screen pause which automatically stops every 24 lines (or whatever you define) during screen output. Remember not to use keys in the "substitute" category discussed earlier. Defining a commonly used key (such as <cr>) should be avoided. When entering new key codes, you will have to use the key-code chart on page 7 of the APPLE reference manual. Note that OS/9 prefers to use ASCII codes whose most significant bit is zero rather than those in the APPLE manual which contain a one in that bit. When using TMODE, just subtract hex 80 from all key codes in the APPLE manual. For example, control A in the manual is hex 81, however, you would type in a "1" when using TMODE.

If you have not already, experiment with the type-ahead feature. You may enter a command (or data to a program) at virtually any time since the keyboard is scanned independently by the 6502 processor's I/O package. The current version has about 80 characters of storage within module TERM for type-ahead.

CHAPTER II

SCREEN DISPLAYS

SPECIAL FEATURES OF THE 40 COLUMN DISPLAY

OS/9 sends characters to the 40 column (standard) display via special software in the module AACIA which interacts with input-output support contained in the "6502 I/O package" (more on this later). Unlike APPLE DOS, the characters you type are not sent directly to the display as you type them, but are first interpreted by OS/9. Modules "TERM", "AACIA", and "SCF" are involved with this. Because of this, you may type virtually any character without disrupting the display. Indeed, if you use the TMODE -ECHO option, a program may retrieve keystrokes without them being displayed. This is useful for entering passwords or accepting input which is to be processed without being displayed.

The 40 column screen accepts the following special characters for control purposes:

Hex	Decimal	Function
$ØC	12	erase and home the cursor (upper left)
$ØB	13	set normal (white-on-black) video
$16	22	set inverse video
$17	23	set flashing video
$19	25	home cursor, do not erase screen
$10	28	move cursor one column right, preserving screen
$ØE	15	move cursor one line down, preserving screen

Again, these codes must come from a file or a program rather than the keyboard.

USING AN 80 COLUMN DISPLAY

With power off, install the VIDEX board (other manufacturers' currently being verified) in slot 2 or 3 and prepare to connect the monitor video cable per the manual for the board. Now boot up OS/9. For now, OS/9 will initially "come-up" using the standard 40 column display so you will have to use that video cable for a moment. After obtaining the usual OS/9: prompt, type:

```
TMODE TYPE=3 -UPC        for slot 3, or
TMODE TYPE=2 -UPC        for slot 2.
```

If you now use the video on the 80 column display board's cable for your monitor you should see the usual OS/9: prompt and hitting <cr> should redisplay it on that monitor. Try the DIR E command in 80 columns...much better! As you gain experience, you will find that you may edit the above TMODE command into the STARTUP file so that OS/9 will use the 80 column board automatically upon booting.

Here are a few details on the use of the 80 column board for the technically oriented: OS/9 uses the on-board ROMs for display (output) purposes, however, the keyboard (input) sections of the ROM are not used. (The interrupt-driven type-ahead philosophy prohibits any device service by "spinning in a loop." Those sections of the manual for the display board which discuss the effect of characters TYPED (vs. sent from a program) should be ignored. For the VIDEX unit, the following summarizes the control characters recognized by the output sections of the ROM firmware:

Hex	Decimal	Function
$ØC	12	erase screen and home cursor
$17	11	erase from cursor to end of screen
$ØA	1Ø	line feed, same column
$19	25	home cursor, no erase screen
$1D	29	erase from cursor to end of line

(the following must each be preceded by the "leadin" character, hex 1E = decimal 30:

$ØØ-$Ø7	ØØ-Ø7	low resolution graphics character set
$1Ø-$17	16-23	line-drawing character set
$Ø8-$ØF	Ø8-15	ASCII codes displayed in mnemonics

To position the cursor, send the following three characters:

```
3Ø X+32 Y+32
```

The following example, coded for BASIC Ø9, will illustrate the cursor positioning function:

```
DIM leadin,x,y:integer
PRINT CHR$(12);
leadin=3Ø

FOR x=Ø to 79
  FOR y=23 to Ø STEP -1
    PRINT CHR$(leadin);CHR$(x+32);CHR$9y+32);".";
  NEXT y
NEXT x
```

which print periods in all positions, bottom to top.

A VIDEX demonstration is included in your package, in directory VIDEX, under the file name DEMO. This is a BASIC/Ø9 program.

USING LOWER CASE

(Note: Lower case operation requires an 80 column display board.)

Your OS/9 supports the popular "one-wire" lower case modification which allows you to shift between upper and lower case using the shift key exactly like a standard typewriter. The next paragraph describes how to incorporate this simple hardware modification if you have not already done so. When incorporated, the 80 column display coupled with a good screen-oriented word processor (available from Microware) will provide a top-rate text processing work station.

A wire is connected to the shift key and connected via a plug to the same I/O connector of the APPLE II. On the APPLE II PLUS, a small clip such as the "E-Z Hook" may be attached (without soldering) to the rightmost, exposed,

vertical connector pin on the keyboard encoder circuit board, located beneath the keyboard. Looking from the front of the computer, this pin is directly beneath the "*" key. On older Apples, the shift key wire is attached to a solder pad located just under the shift key on the top (outside) of the computer. They key-cap must be temporarily removed to attach the wire. In either case, the other end of the wire connects to a Dual-in-line plug, pin 4 (PB2). This plug is then inserted into the Game I/O connector (same paddle plug).

Another solution to the lower case problem is the use of one of the popular replacements for the Apple's keyboard encoder circuit board.

When using the "one-wire" shift key modification, the keyboard operation is as follows:

Press and hold shift, then press esc (escape), then release esc and shift (sounds harder than it really is).

This turns on the effect of the shift key modification. To turn it off, repeat the above. Each "shift-esc" reverses the on/off switch for the shift key modification.

As long as the shift-esc "switch" is on, you use the keyboard just as you would on a standard typewriter. To revert to the normal APPLE mode, just press shift-escape again. There is one exception to the rule here: To set the "@", "^", and "]" characters, you must press shift-esc to leave lower case mode and type shift P for "@", etc. Fortunately, these characters are rarely used.

While on the subject of keyboards, you may wish to review the earlier paragraph on "substitution" characters. Many of these are available only with the 80 column display.

CHAPTER III

USING THE CLOCK AND MULTI-TASKING

We suggest that you skip this section until you have mastered the fundamentals of OS/9. The multi-tasking feature of OS/9 is not dependent upon the installation of a system time-of-day clock card.

The OS/9 system really shines when you enable the time-of-day clock. This term refers to the practice of keeping the processor within a computer (two processors in the case of APPLE OS/9) busy at all times, even when waiting on the operator or a peripheral. The concept is to have two or more programs or "processes" bidding for processor time. Your OS/9 system provides this feature and makes possible many time-saving capabilities for the user. To utilize multi-tasking, you will load a clock device descriptor module appropriate for your hardware and enable it.

The following options are (at the publishing date) available:

1. You have no clock board for the APPLE: use NOCLK
2. Westside Electronics Superclock II (tm), slot 5: use WESCLK5
3. Mountain Computer CPS multi-function card, slot 7: use CPSCLK7

To start the clock, simply type:

```
LOAD <name>
```

where <name> is one of the names listed above.

Then type:

```
SETIME
```

to run the set time-of-day utility which will prompt you to enter same. Now type:

```
DATE T
```

which will display the date and time. Try this a few times. If you specified "NOCLK", the time is maintained very roughly by a delay loop in the 6502 processor's program within an idle section. If you do have a clock board, the current date/time is obtained from that board, irrespective of the date time which you entered. In fact, if you reply with a <cr> to SETIME's prompt, the date/time will automatically be established within one second (n/a for "NOCLK"). On the unusual occasions the board's date/time is incorrect, use the software supplied by the manufacturer to set the time within the logic I.C. on the board. This is most often due to a battery failure.

With the clock running now, type:

SHOWTIME&

and look on the video for the 40 column display. In the upper right corner you will see the date/time being displayed and repeatedly updated. The command you entered told OS/9 to run SHOWTIME and (&) continue the current program (your SHELL command processor). If you now type:

PROCS

you will see a list of active procedures (tasks), one of which will be SHOWTIME. SHOWTIME tells OS/9 to put it "to sleep" for one second, then it executes and updates the display, sleeps again, etc, etc. Try entering a few commands such as DIR and MDIR while SHOWTIME is present. Since it is a very simple program, its execution time has little effect on your activities. Read about the SETPR command and try changing SHOWTIME's and your (SHELL) priorities, using PROCS to see the results. The higher the priority number, the more "rank" a process has when a fight for processor time occurs. A process at priority 255 is akin to a 5-star general! For the adventuresome, read the User's guide on SLEEP and try it out.

As you will see in the next section, the multi-tasking is quite useful to do "background" work such as printing files on hard copy while simultaneously word-processing or programming, etc. A later section will also discuss the Time-

sharing monitor for adding additional terminals to your APPLE to allow multiple workstations.

With the clock operating, OS/9 is being interrupted at approximately 100 millisecond intervals by a program within the 6502 I/O package. Each of these intervals is called a "tick". The accuracy of these intervals is very rough due to the varying workload on the 6502. If you are using a clock board for the date/time, the actual time-of-day will be as accurate as the board is. If you have no clock board (NOCLK was used above), you may want to reenter the date/time every hour or two (use the SETIME command). It is very beneficial to keep the date correct since OS/9 automatically logs the creation and last modified date/time for all files and many language listings.

Addition of a simple 60 Hz synchronizing signal will allow the date/time to be much more accurate with NOCLK. This addition also makes the clock "tick" rate much closer to 100 milliseconds. This may be important for applications making use of the F$SLEP function in assembly language. In most applications, this sync signal maintains the time accurately enough to eliminate the need for a clock board.

USING THE PRINTER

The printer interface device descriptors (same jargon as used in the section on clocks) are, at the publishing date, as follows:

EPSON1	for an EPSON brand parallel interface to Centronics-like printers, including the MX80 and MX100, for slot 1.
CPSLPT7	for a Centronics parallel interface via the Mountain Computer CPS Multifunction card, slot 7.

As discussed later, it is relatively easy to change the slot number (see SETDDM, below). Programmers may choose to adapt the supplied driver source code for other interface

cards, perhaps using the interprocessor interrupt described in Chapter V.

To access the printer for the first time after booting, type:

```
LOAD <name>      (one of the above names, as appropriate)
```

This command need only be entered once and may be within the STARTUP file.

Once loaded, to access the printer use the name "/P" to refer to the printer. For example:

```
LIST /DØ/SYS/ERRMSG >/P
```

will send the indicated file to the printer and return an OS9: prompt when the printing is finished. The ">" is the "I/O redirection" convention of OS/9 and in this case means "Don't send output to the screen; send it to the printer." THE ">" signal applies to standard listing output "path". Error messages go on a path called the "error path" which may be redirected using ">>" instead of or in addition to the ">". For example:

```
Like BASIC Ø9 MYPROGRAM >/P >>/P
```

will invoke BASIC Ø9 with said file name and reroute standard listings as well as error messages to the printer. The program itself never knows. This is the idea behind "transparent" I/O wherein you define I/O paths and devices when the program is run, rather than when it is written.

Consider:

```
DIR E /DØ/CMDS >/P&
```

This command will send a directory listing to the printer and continue (the &) the current program.

Similarly:

```
LIST /DØ/SYS/ERRMSG >/P&
```

will list said file name "in the background."

To type an unwanted printout, type:

```
KILL n
```

where n is the process number of the "process" doing the printout (e.g., LIST).

Note: If the printer is off-line and the abort will not take effect until it is restored to on-line and prints one more line.

Do you recall how to see the list of processes (to find the "n" for LIST)? Hint: PROCS.

To use the printer from BASIC/Ø9 without using the ">" redirection scheme:

```
DIM LP:INTEGER
OPEN #LP,"/P"
PRINT #LP,"HELLO ON THE PRINTER"
PRINT "HELLO ON THE CRT"
```

or better style:

```
DIM LP,OUTPATH:INTEGER
INPUT "PRINTED REPORT (Y/N): ";YN$
IF YN$="Y" THEN
  OPEN #LP,"/P"
  OUTPATH=LP \ REM assignment of true path number
ELSE
  OUTPATH=1 \ REM 1 is screen unless redirected by SHELL
ENDIF
PRINT #OUTPATH,"HELLO ON WHATEVER DEVICE!"
```

OS/9 allows you to have multiple printers; you'll just need a device descriptor and possibly a device driver for each. The second printer is typically named "/P1" in the device

descriptor module. A single device driver, properly written, may drive multiple printers which use identical interface cards. The 100 millisecond clock tick of OS/9 and the F$SLEP system call (assembly language) allow a driver to be written completely independent of the 6502 I/O package as well as the OS/9 itself.

USING THE SERIAL RS-232 PORT

This section describes the supplied support for the Mountain Computer CPS multifunction card's serial port. Other manufacturers' serial ports are not available (as of the publishing date), however, then you may write your own given the information in the System Programmer's manual and the section of the assembler manual which describes the 6502 I/O package.

As supplied, the CPS serial port is configured for asynchronous operation at 300 baud (bits per second), 8 bits per character, no parity. These parameters can be altered as described later. To use the port, these steps are suggested:

1. Obtain the cable to connect the peripheral (CRT, modem, printer, plotter, etc.) from the manufacturer or fabricate it per the instructions in the manual for the card.

2. Ensure that Data Set Ready and Carrier Detect are strapped to +12 VDC or connected to a device off-line signal from the peripheral. OS/9 will not send or receive on the port if either of these signals are false (-12 VDC).

3. Ensure that the CPS card's Clear to Send signal (an input) is connected to +12 VDC or connected to the peripheral's ready/busy signal which controls the acceptance of data from the CPS card. If this signal goes false, the current character will be the last output until the signal again goes true.

4. Ensure that the card operates properly and is able to successfully communicate with the peripheral using the manufacturer's test programs.

Install the card in slot 7 for OS/9 (this is alterable, described below). Set up the device for the baud rate, bits per character, and parity as described in the above paragraph. If the peripheral cannot achieve these parameters, you will have to change them within the 6502 I/O package. This is explained in the end of this section.

Having accomplished the above, type:

```
LOAD T1
```

which is a file in the usual CMDS directory. Now the command:

```
DIR >T1
```

will send a directory listing to the serial port. If any of the interface signals (carrier detect, data set ready, or clear to send) are false (-12 VDC), the transmission will be postponed until all are true. Conversely, the command:

```
LIST </T1
```

will copy the incoming serial data (if any) to the CRT until an end-of-file is received or you manually abort the operation via a control Q.

In some applications, the peripheral may transmit a large number of characters and exceed the ability of OS/9 to dispose of them. For example:

```
LIST </T1 >/P
```

will copy the serial input data to the printer, however, the printer speed may be slow enough to cause an "overrun" condition. OS/9 does buffer all input (and output), however, large speed mismatches will cause data loss and an error. In virtually any situation, lowering the line speed

(baud rate) will lessen speed mismatch problems. Some applications may require a "protocol" which is a program to accept data from /T1 in manageable blocks and send a brief acknowledging message to the sender to trigger another block. The subject of protocol is quite extensive and cannot be properly treated here.

Fortunately, the clear to send line can be wired to the peripheral to solve the outbound data flow rate problem without protocol. Conversely, the peripheral can send the pause SETDDM command's PAUSE character to force OS/9 to momentarily cease transmission. The NULL=n parameter of SETDDM may solve certain inteface problems. (SETDDM is explained below.) Thus, in most applications, protocol is not required for flow control. Critical applications may need a particular protocol to ensure error-free transmission.

Most applications are simple enough to avoid complex interface methods. Indeed, a printer or similar device is easily integrated. The addition of a second CRT (a CRT in the $600-900 class) or hardcopy terminal is readily accomplished and greatly enhances the usefullness of the system. A letter quality printer with a serial interface may also be attached.

CHANGING THE CPS CARD'S SERIAL DATA RATE
(Please refer to the CPS card manual and the data sheet for the Signetics 2651 integrated circuit for full details)

Two eight-bit variables are required to define the data rate and character size/parity for the serial port. These variables are not stored in the device descriptor module but rather are located within the 6502 I/O package. These variables are called "CPS1" and "CPS2" and their addresses are given in the appendix. As shown in table 6 of the Signetics data sheet, the baud rate may be defined by placing one of the following values in location "CPS2":

Baud Rate	Hexadecimal Value	
110	$32	
150	$34	
300	$35	
600	$36	
1200	$37	
2400	$3A	
4800	$3C	
9600	$3E	
19200	$3F	(see Signetics data sheet for other speeds)

Likewise, the character size and parity go in location "CPS1" per the following (binary layout):

number of stop bits:	Ø1xx xxxx	1 stop bit
	11xx xxxx	2 stop bits
character length:	xxxx 1Øxx	7 bits per character
	xxxx 11xx	8 bits per character
		(5 and 6 also available)
parity selection:	xxxØ xxxx	no parity
	xxØ1 xxxx	odd parity
	xx11 xxxx	even parity

Calculate the desired value, perhaps referring to the data sheet, and place them in memory BEFORE first LOAD'ing T1. The values may be stored using the OS/9 Debugger or as POKE's from BASIC Ø9. Once /T1 is used, changing these values will have no effect. Further details on the 6502 I/O package control of the Signetics USART chip may be obtained by writing to Conejo Computer Products, requesting same. Chapter V describes how to make the change permanent in the 6502 I/O package.

Although not implemented, the synchronous mode offered by the Signetics may be utilized for communications circuits such as IBM Bi-sync, its variants, and DEC's DDCMP. There are currently no plans to provide this, subject to user demand.

THE SETDDM UTILITY

This utility command is supplied in the APPLE OS/9 package and is not discussed in the User's guide.

The purpose of this utility is to allow a device descriptor module (DDM, described in the System Programmer's Manual) to be modified. The TMODE utility modifies a COPY of the device characteristics contained in the DDM. Thus, TMODE only works for a device for which a path has been established, e.g., the device is "open" for read or write. TMODE works for the screen and keyboard since they are most always open on paths to SHELL. Once a DDM has been modified by SETDDM, programs which subsequently open a path to that device will use the newly modified characteristics.

The SETDDM command is (syntax)

SETDDM <device-name> <arglist>

where <device name> is /P or /T1, etc. and
<arglist> is optional and identical to those discussed in the User's Guide for the TMODE command.

Examples:

SETDDM /P<cr>	simply displays current settings for the printer	
SETDDM /T1 NULL=50	-echo <cr>	alters two characteristics

USING SETDDM TO ALTER AN INTERFACE CARD'S SLOT NUMBER

The command:

SETDDM <device name> TYPE=n

redefines the DDM's characteristic for slot number and card type. In the current version, the following possibilities exist:

Looking at the n in TYPE=n as eight binary bits:

7654	3210
xxMM	MSSS

Note: xx are reserved and unavailable where MMM is the manufacturer or board type
SSS is the APPLE slot number, excluding zero

Board type codes:

EPSON Parallel printer	MMM = ØØØ
Mtn Computer's CPS parallel	MMM = ØØ1
(future growth)	

For example:

SETDDM /P TYPE=Ø2
Is for modifying the supplied EPSON1 DDM for slot 2.
SETDDM /P TYPE=12
Is for modifying CPSLPT7 for slot 2.

The MMM code is present to allow the 6502 to execute the appropriate code within the I/O package, given the interface method. The on-board firmware cannot be used since it "loops on busy" which would have a severely detrimental effect on the ability of OS/9 to do concurrent I/O. At present, the clock DDM's need no MMM code in that the drivers are totally 6809 code. Clocks do need the SSS code, however.

The 40 column display VIDEX board (device /TERM) need no MMM or SSS; the TMODE command is used to set the slot and, since these devices have exclusive use of the COUT vector for the 6502, no MMM code is needed.

SAVING A MODIFIED DEVICE DESCRIPTOR MODULE (DDM)

Case in point: modify EPSON1 to reside in slot 7:

```
LOAD EPSON1
SETDDM /P TYPE=7
(other SETDDM commands as required)
SAVE /DØ/CMDS/TEMP EPSON1
VERIFY <TEMP >EPSON7 U
ATTR EPSON7 E
DEL TEMP
UNLINK EPSON1 (repeat this until module not found error)
LOAD EPSON7
DIR >/P
```

From now on, use EPSON7 rather than EPSON1.

If the CPS card is moved from slot 7, you must modify the DDMs for both CPSLPT7 and T1 to reflect the new slot number.

Do not SETDDUM or SAVE or LOAD with a device currently in use!

CHAPTER IV

USING BASIC Ø9

This section contains a brief outline on using BASIC Ø9 for the newcomer. Please spend a few hours studying the BASIC Ø9 manual before reading this section.

BASIC Ø9 is in directory CMDS, file BASICØ9. If you are going to be using it frequently, LOAD BASIC Ø9 immediately after you boot the system. If you only type:

```
BASICØ9
```

without previously doing the LOAD BASICØ9, OS/9 will discard BASIC Ø9 when you type "BYE" to exit BASIC Ø9.

The easiest way to learn BASIC Ø9 is to study the manual and try it! After you have typed in the BASIC Ø9 from the SHELL (from the OS9: prompt), you will see a B: prompt telling you that you are in the command level. Unlike other Basics, you cannot enter a direct command from the command level (i.e., "PRINT 2+2"). You enter these in the debugger mode which has a D: prompt. More on this later.

From the B: prompt, if you hit <cr>, a directory of modules or procedures is displayed. From command level you must specify which procedure you wish to alter and/or execute. On initial entry, there are no modules. From here, you may wish to load and run one of the supplied programs. To do this, place the diskette with the programs in, say, drive D1 and type: CHD /D1/?, where ? is the directory in which the program is located. Now type: LOAD <program name>. Hitting <cr> now will show the procedures loaded. Type RUN <procedure name> to execute a selected procedure.

Enter a command to invoke the procedure editor:

```
E MINE      means edit procedure named MINE
```

The E: prompt means you are in the editor. Using the BASIC Ø9 manual, enter a brief example program. You'll need to spend a few minutes getting the hang of the editor.

To exit to the command level, type "Q<cr>". From the B: prompt, hit <cr> to see the procedure directory. Now run your program: RUN<cr>. If it "bombs out" with an error, you will be in the debugger with a "D:" prompt. Exit with another "Q<cr>" command. Try creating a second procedure named "YOURS" with the editor. Now the procedure directory will show two names, with an "*" next to the last one run or edited. If you type "RUN", this is the one which will go. "RUN <name>" will run a specific one.

To save a bunch or procedures, type:

SAVE* <pathname> (defaults to the current data directory)

This dumps all procedures into a single file. Now a KILL* will purge the procedure directory. To get them back, type LOAD <pathname> which will load 'em all back again. LOAD/SAVE of specific ones is explained in the manual.

Now it's time to REALLY study the manual and the supplied BASIC Ø9 programs to get it down pat.

After you've become a semi-expert, here are some of the common points of confusion:

1. B: LIST <procname> >/P to get a pretty listing

2. LOAD <pathname> then KILL <procedure> to reuse old modules.

3. When invoked from the shell via:
   BASIC Ø9 <pathname>
   pathname must match the name of one of the procedures.

4. If you get fancy with packed (compiled) procedures, keep them in the execution directory.

5. You'll make a lot of mistakes involving variables' type, i.e., using a variable as if it were integer (as path number variables) but forgetting to DIM IT: INTEGER.

6. If you do not use an 80 column display, your code will naturally tend to be awkward due to the temptation to fit it onto the 40 column display. Also, lower case dramatically improves legibility.

7. If you run out of memory (error 207), use the MDIR command to see what modules are in memory. Don't forget that the shell command: "BASICØ9 <program> #nnK" may be needed if <program> will not fit in the default (#nnk omitted) memory size. Sometimes rebooting is required to clean up a situation where lots of modules have come and gone from memory, leaving many discontinuous regions. With experience, you will learn how to avoid this.

For applications requiring large amounts of memory, you may wish to obtain a product called RUNB which is the run-time code of BASIC Ø9, minus the compiler, editor, and debugger. With this product, you may run compiled (and debugged) code. RUNB is approximately half the size of BASIC Ø9.

The more you use BASIC Ø9, the more you will appreciate the speed with which you can get a program coded and running correctly. Not having a syntax error pop up 2-1/2 months after the program was "done" is a real delight. You'll soon learn to develop reusable procedure modules so that, with time, the software library begins to build on itself!

Don't forget the right-arow key for multiple statement lines. Many of the "substitution" keys mentioned in Chapter I are there for BASIC Ø9.