DOS DEVICE DETECTIVE

APPLE UTILITIES

Bring the power and convenience of device-independent file handling to DOS 3.3 with this small program, and save yourself the trouble of continually typing in slot and drive numbers.

Device-independent file access is a capability of many disk operating systems. ProDOS is one such operating system. To access a file, the user specifies the path name. ProDOS does not need to know the physical location of the file. It searches through the devices connected to the system until it locates and loads the specified file.

DOS 3.3, on the other hand, is device dependent, and this can cause file access problems. FILE NOT FOUND errors and inadvertent creation of text files for files that already exist on some alternate slot and/or drive are among the more common problems.

The amount of code that programmers have been forced to devote to device-dependent file handling is legion. A typical solution is to fix the operation of the program to one system configuration, e.g., the program disk in drive 1, data disk in drive 2, and both drives connected to the boot slot. Not only does this solution limit the transportability of the software, it also requires extensive handholding and documentation for those novice users who neither know nor care about slots, drives, and the like.

A better solution is to patch DOS 3.3 so that it provides for device-independent file access. Just such a patch is the subject of this article. With DOS Device Detective installed, if the requested file is on the system (in any slot, any drive), the patched DOS will find it. FILE NOT FOUND errors for existing files and the myriad of other annoyances become a thing of the past.

USING THE PATCH

Because the program installs itself between DOS and its buffers (by moving the DOS buffers down in memory), it should be executed before any files are opened or any variables are declared. Once installed, the patch will remain active until the system is rebooted or the INIT command is given. To have the patch installed any time DOS is booted, simply SAVE the following one-line BASIC program as the Hello program (or part thereof) on your boot disk:

```
1Ø PRINT CHR$(4)"BRUN DETECTIVE,A$2ØØØ"
```

DEVICE-INDEPENDENT FILE ACCESS

For longtime users of DOS 3.3, the effect of the patch can be a bit disconcerting at first. As an example, attempt to LOAD a nonexistent BASIC program by typing LOAD XXXX. After searching the default disk, instead of a FILE NOT FOUND error, the alternate drive connected to the same slot will come on. If you have multiple disk cards, the drives connected to each of them will be accessed until each disk drive on the system has been searched for the nonexistent file. Finally, the original disk drive will come on again briefly as DOS Device Detective admits defeat with a FILE NOT FOUND error.

If you follow the unsuccessful LOAD with a CATALOG command, the directory of the disk in the first drive searched will be presented, indicating that despite the extended romp through the various drives, the default slot and drive have not been changed. On the other hand, had the file been located on one of the other drives, the file would have been LOADed, and the drive where the file was found would become the default drive.

On cataloging the disk, you should notice one further difference: the normal "DISK VOLUME" header has been replaced with "DETECTIVE." The new header indicates that DOS Device Detective is currently patched to the system.

The search for the file always begins with the disk in the default drive. Since the programs and files belonging to a particular application are almost always resident on the same disk, beginning the search with the default or last-accessed drive results in a considerable reduction in file access time relative to a fixed search pattern. Drives that are connected to slots other than the default slot are searched starting with the lowest slot number.

If two files on the system have the same name, only one will be accessed on a given call to DOS. Which of the two is accessed depends on which drive — as determined by the search pattern — is searched first. Incidentally, specifying a nonexistent device as the default will only slow the search; if the file is on the system, DOS Device Detective will find it. Similarly, disk cards with only a single drive attached (e.g., the typical Apple IIc configuration with its single built-in drive) will also slow the search. This is because DOS 3.3 tries 48 times before giving up on the nonexistent second drive.

FILE ACCESS COMMANDS

Each of the file access commands (LOAD, BLOAD, RUN, BRUN, VERIFY, DELETE, LOCK, UNLOCK, APPEND, RENAME, CHAIN and EXEC) initiates a search for the specified file. If the file is found, then the specified command is performed, and the slot and drive — even if different than the ones specified with the command — are defaulted to the device on which the file was located. If the file is not located, a FILE NOT FOUND error message is generated, and the default slot and drive parameters are unchanged.

> *A better solution is to patch DOS 3.3 so that it provides for device-independent file access.*

The changing of the default slot and drive parameters upon locating the file provides a simple method of determining the location (or existence) of a particular file or diskette on the system. To locate a particular file, simply VERIFY the file name. If the file is on the system, the default slot and drive parameters will be changed to reflect the location of the file. (These slot and drive values may be found by PEEKing 43626 ($AA6A) and 43624 ($AA68), respectively.) If the file is not found, you can be assured that it is not on the system.

Similarly, to locate a particular diskette, VERIFY a file name that you know is on the desired diskette. If the file is found, then the default slot and drive parameters will point to the desired diskette. If the file is not found, your program can request that the user insert the appropriate disk (in any slot, any drive), and then repeat the VERIFY command. By using a unique Hello program name on each diskette which can then be verified, each diskette can be addressed uniquely — independent of its physical location — in a fashion similar to ProDOS volumes.

The program DETECTIVE.DEMO demonstrates the use of DOS Device Detective (see **Listing 3**). When you run this program, after a short title screen presentation, the DETECTIVE patch is installed. Remove the disk containing the DETECTIVE program, and put it in any drive on your system. Press Return, and DETECTIVE will search the drives on your system until it finds itself. The demo will then display the slot and drive numbers where the file was found. If none of the disks on your system contains the file DETECTIVE, the program will tell you that, also.

ENTERING THE PROGRAMS

If you have an assembler, enter the source code from **Listing 1** and assemble it. If you don't have an assembler, enter the Monitor with CALL −151. Then enter the hex code from **Listing 2** and save it with the command:

BSAVE DETECTIVE,A$2000,L$11A

To enter DETECTIVE.DEMO, type in the Applesoft program in **Listing 3**, and save it with the command:

SAVE DETECTIVE.DEMO

For more information on entering *Nibble* programs, see the Program Listings section at the end of this issue.

FILE CREATION COMMANDS

The file creation commands (OPEN, SAVE and BSAVE) behave similarly to the file access commands, except that upon failing to locate the specified file on any drive, the file is *created* on the default disk. For example, to SAVE a new BASIC program to the disk in drive 2 connected to slot 6, issue the command:

SAVE *progname*, S6, D2

If the program name is truly unique, DOS will search each and every drive on the system for the file, and then return to slot 6, drive 2 to save it. If, however, a file of that name already exists on the system, DOS will attempt to save the new program over it. If the found file is of the same file type (and not locked), it will be replaced by the new file. If it is of some other file type, DOS will exit with a FILE TYPE MISMATCH error and the slot and drive defaults will be changed to point to the disk containing the offending file.

HOW THE PATCH WORKS

DOS Device Detective (**Listing 1**) assumes an Apple II (any variant) with a minimum of 48K RAM and DOS 3.3 located at $9D00 (i.e., the usual configuration). The first section of the program, labeled "Install," is used to relocate the patch into high RAM (over the first DOS buffer); it is discarded once the patch is in place. Although the program is assembled to reside at $9BE3, the relocation code is written to be BRUN from $2000 (8192).

The Install section of the code first relocates the patch into the page of memory immediately below DOS. It then calls a subroutine within the patch itself, called ATTACH, which modifies DOS to point to the patch. Finally, the Install section exits through another routine within the patch, called CREATE, which instructs DOS to rebuild its buffers below the patch. Both of these routines are discussed in more detail later.

DOS 3.3 has a hierarchical architecture consisting of three nested systems. The outside or highest level is the Command Interpreter, which serves as the interface between the user and actual disk access.

The intermediate level is the File Manager which, as the name suggests, handles all of the details associated with reading, writing, organizing and generally keeping track of files. The File Manager is called by the Command Interpreter to perform the tasks that the user has requested.

The lowest level is the Read/Write a Track/Sector (RWTS) subroutine, which handles the essentials of actually reading from and writing to the physical device. It is called by the File Manager, and knows nothing about files, but only tracks and sectors.

The bulk of the work, then, resides with the File Manager, and it is here that DOS Device Detective patches DOS.

All file access, including such high-level activities as renaming and deleting files, initially requires that the file be opened. This burden falls to a section of the File Manager code known as COMOPEN — the COMmon OPEN routine used for all file access within DOS 3.3 (see "Opening and Closing Files" by Sandy Mossberg, *Nibble* Vol. 5/No. 4, and *Beneath Apple DOS* by Don Worth and Pieter Lechner, Quality Software, 1981).

Before a file may be opened, it must be located (or allocated, if it is a new file) within the Volume Table of Contents (VTOC). Locating a file involves retrieving such information as its size, file type, and location on the diskette. The COMOPEN routine contains within it a call to another subroutine (which I refer to as ALLOC, for ALLOCate) whose job it is to locate the file on the diskette and stuff such information into various data tables within DOS. DOS Device Detective is patched to DOS by replacing this call with a jump to the patch, so that all attempts by the File Manager to open a file will now be vectored to the patch. The call to ALLOC, then, becomes a job for the patch rather than the File Manager, and, as will be explained, occurs in a much more indirect fashion.

This modification to DOS (and another modification to the section of code that handles the INIT command, discussed later) is the function of the ATTACH subroutine (**lines 175-193**) in **Listing 2**. The ATTACH subroutine also replaces the normal DOS catalog header with the "DETECTIVE" header, and is called by the Install code when the patch is first installed on the system.

Any call to COMOPEN, then, will now enter the patch at SCHDRV (**line 60**), which immediately calls another subroutine, LOCFIL (**lines 127-130**). LOCFIL, in turn, calls yet another subroutine, GETFIL (**lines 132-139**), which, after some finagling with the system stack, finally calls ALLOC.

Why the subterfuge and indirection in calling ALLOC? After all, the File Manager COMOPEN routine calls it directly. Well, when the File Manager is called by the Command Interpreter, one of the first things it does is save the stack pointer in a safe location. In this way, if it ever runs into a serious problem, such as an I/O ERROR, it can simply replace the stack pointer (so that it now points to the appropriate return address within the Command Interpreter) and RTS to the original caller. In doing so, however, it skips over the tangled web of nested subroutine calls (such as COMOPEN to ALLOC to. . .) that got it into trouble in the first place. This may be fine for the File Manager, but not for DOS Device Detective, which is now part of the tangled web. DOS Device Detective needs to know of the error so that it can check another drive.

When LOCFIL calls GETFIL, the return address to LOCFIL is pushed onto the system stack. GETFIL retrieves the stack pointer so carefully stashed away by the File Manager, and saves it in another location. It then stores the current stack pointer — which points to LOCFIL's return address — in the File Manager's bail-out location, and jumps to ALLOC. If no grievous errors occur, then ALLOC returns in the normal fashion to LOCFIL. If, however, the File Manager chooses to take the bail-out route due to, say, an I/O ERROR, control still returns to LOCFIL. In either case, following the call to ALLOC, LOCFIL restores the original stack pointer value to the File Manager's hiding place. LOCFIL then returns to whatever called it which, at this point of the story, is still SCDRV, the first line of the patch.

Error status within all three sections of DOS is indicated by the Carry bit in the system's Status Register; Carry clear means "no error," while Carry set means that an error has occurred. Upon the return from LOCFIL, the absence of an error indicates that a file having the requested name exists in the diskette's directory. As this is what the patch was looking for, SCHDRV branches to DONE (**lines 73-77**), which replaces the default slot and drive parameters with those corresponding to the slot and drive in which the file was found. DONE then jumps back into COMOPEN at a point immediately following the patch. Next, COMOPEN checks that the found file is of the correct file type (if not, it exits to a FILE TYPE MISMATCH error), completes the opening of the file, and returns to the command interpreter.

If LOCFIL returns with an error, then the file was not found. SCHDRV then sets the drive number to the alternate drive (**lines 62-64**), and tests this value against the default drive number. If they match, then both drives on this disk card have been searched, and the routine branches to NEWSLT. Otherwise, SCHDRV jumps to the beginning of COMOPEN, which, after resetting some parameters, returns control to SCHDRV. Then the whole process just described is repeated on the alternate drive.

If both drives on a given disk card have been searched, control is passed to NEWSLT (**lines 83-87**), which begins the search for another disk card. NEWSLT determines whether the current call for a new slot is the first for this file by comparing the current slot value to the default slot. If they match, then NEWSLT initializes the search and falls through to CHGSLT. Otherwise, NEWSLT branches to CHGSLT with the current slot value.

CHGSLT (**lines 92-103**) repeatedly increments the slot value until it finds a slot that is not the default slot (which has already been searched), and which contains a disk card. If such a slot is found, then the routine exits once again to the beginning of COMOPEN, which returns control to SCHDRV. Otherwise, control is passed to NOTFOUND (**lines 109-121**), which resets the slot and drive parameters to their original values, allocates the file via LOCFIL, and returns control to COMOPEN. COMOPEN then checks whether the file may be created and, if so, creates the file on the disk. Otherwise, it exits with the FILE NOT FOUND error.

DISCONNECTING THE PATCH AND INIT

I mentioned earlier that installing DOS Device Detective also patches the INIT command handler. The purpose of this second patch is to remove the first patch and return DOS to its nonpatched state whenever the INIT command is used. If this isn't done, then any diskette initialized with the patched DOS will contain a copy of DOS that is incapable of accessing any files. This is because the initialization routine will write only DOS and not the DOS Device Detective patch to the diskette. Consequently, whenever INIT is used, DOS Device Detective is disconnected from the system. You can verify this, by the way, by executing a CATALOG after initializing a diskette — the "DETECTIVE" header will have been replaced with the normal "DISK VOLUME."

DOS Device Detective is not gone, just disconnected. You can reconnect it following an INIT command by calling the ATTACH routine (CALL 40119). Similarly, you can disconnect DOS Device Detective at any time by calling the DISCON routine (CALL 40072). As with the INIT command, calling DISCON simply disconnects DOS Device Detective and returns DOS to its normal state — it does not remove the routine from memory. Incidentally, DISCON (or INIT) does not return DOS completely to normal. Any diskette initialized with a copy of DOS from which DOS Device Detective has been disconnected, when booted, will leave one page ($9C00-$9CFF) of protected memory between itself and its buffers. You may use this area (when not using DOS Device Detective) for your own machine language programs.

The last routine to be discussed was included with the permanent DOS Device Detective code (rather than the Install code) because of its general utility. It is the CREATE routine (**lines 198-207**), and it is used to create (or remove) space between DOS and its buffers (see "Managing and Moving Disk Buffers" by W. Reynolds, *Nibble Express* Vol. 1). To use this routine, you must POKE the appropriate values (low address, high address, and the number of buffers DOS is to build, respectively) into the first three page-zero locations, and then CALL 40166.

As an example of using CREATE, consider removing DOS Device Detective entirely from the system. In this case, CREATE will be used to remove the one-page space between DOS and its buffers, eliminating DOS Device Detective in the process. First, disconnect DOS Device Detective by CALLing DISCON, then:

```
POKE 0, 0: POKE 1, 157: POKE 2, 3: CALL 40169
```

This sequence of commands instructs DOS to rebuild three buffers beginning at $9D00. DOS will now be completely back to normal. The effect of using CREATE is similar to that of the DOS MAXFILES command, so be sure to use it only when no files are open and before any string variables have been declared.

DOS Device Detective listings begin on page 106

Listing 2 for for Ultra Fast Pix

ULTRA.FAST *(continued)*

```
                8630 *       ZERO PAGE SWAP AREA
                8640 *
79D4-           8650 REGSAV .BS REGNUM   RESERVE JUST ENOUGH ROOM
                8660 *
79ED-           8670        .BS $7A00-* MOVE TO NEAREST PAGE BEGINNING
                8680 *
7A00- 96 97 9A
7A03- 9B 9D 9E
7A06- 9F A6     8690 WRTABL .HS 96979A9B9D9E9FA6
7A08- A7 AB AC
7A0B- AD AE AF
7A0E- B2 B3     8700        .HS A7ABACADAEAFB2B3
7A10- B4 B5 B6
7A13- B7 B9 BA
7A16- BB BC     8710        .HS B4B5B6B7B9BABBBC
7A18- BD BE BF
7A1B- CB CD CE
7A1E- CF D3     8720        .HS BDBEBFCBCDCECFD3
7A20- D6 D7 D9
7A23- DA DB DC
7A26- DD DE     8730        .HS D6D7D9DADBDCDDDE
7A28- DF E5 E6
7A2B- E7 E9 EA
7A2E- EB EC     8740        .HS DFE5E6E7E9EAEBEC
7A30- ED EE EF
7A33- F2 F3 F4
7A36- F5 F6     8750        .HS EDEEEFF2F3F4F5F6
7A38- F7 F9 FA
7A3B- FB FC FD
7A3E- FE FF     8760        .HS F7F9FAFBFCFDFEFF
                8770 *
                8780 *
                8790 *       READ TABLE DEFINITION
                8800 *
                8810 *       SIX DATA BITS ARE 1-6
                8820 *
                8830 *       READ6L = 65432100
                8840 *       READ2R = 00000065
                8850 *       READ4L = 43210000
                8860 *       READ4R = 00006543
                8870 *       READ2L = 21000000
                8880 *       READ6R = 00654321
                8890 *
                8900 *       SO FOUR BYTES READ ARE
                8910 *       SPLIT INTO THREE BYTES AS:
                8920 *
                8930 * BYTE 1 = D1(READ6L)+D2(READ2R)
                8940 * BYTE 2 = D2(READ4L)+D3(READ4R)
                8950 * BYTE 3 = D3(READ2L)+D4(READ6R)
                8960 *
7A00-           8970 READ6R .EQ WRTABL   PLACE WRTABL IN SPARSE READ6R
7A40-           8980        .BS $7A80-* MOVE UP TO LAST 80 BYTES IN PAGE
                8990 *
7A80- 00 00 00
7A83- 00 00 00
7A86- 00 00     9000        .HS 0000000000000000 80-87
7A88- 00 00 00
7A8B- 00 00 00
7A8E- 00 00     9010        .HS 0000000000000000 88-8F
7A90- 00 00 00
7A93- 00 00 00
7A96- 00 01     9020        .HS 0000000000000001 90-97
7A98- 00 00 02
7A9B- 03 00 04
7A9E- 05 06     9030        .HS 0000020300040506 98-9F
7AA0- 00 00 00
7AA3- 00 00 00
7AA6- 07 08     9040        .HS 0000000000000708 A0-A7
7AA8- 00 00 00
7AAB- 09 0A 0B
7AAE- 0C 0D     9050        .HS 000000090A0B0C0D A8-AF
7AB0- 00 00 0E
7AB3- 0F 10 11
7AB6- 12 13     9060        .HS 00000E0F10111213 B0-B7
7AB8- 00 14 15
7ABB- 16 17 18
7ABE- 19 1A     9070        .HS 001415161718191A B8-BF
7AC0- 00 00 00
7AC3- 00 00 00
7AC6- 00 00     9080        .HS 0000000000000000 C0-C7
7AC8- 00 00 00
7ACB- 1B 00 1C
7ACE- 1D 1E     9090        .HS 0000001B001C1D1E C8-CF
7AD0- 00 00 00
7AD3- 1F 00 00
7AD6- 20 21     9100        .HS 0000001F00002021 D0-D7
7AD8- 00 22 23
7ADB- 24 25 26
7ADE- 27 28     9110        .HS 0022232425262728 D8-DF
7AE0- 00 00 00
7AE3- 00 00 29
7AE6- 2A 2B     9120        .HS 0000000000292A2B E0-E7
7AE8- 00 2C 2D
7AEB- 2E 2F 30
7AEE- 31 32     9130        .HS 002C2D2E2F303132 E8-EF
7AF0- 00 00 33
7AF3- 34 35 36
7AF6- 37 38     9140        .HS 0000333435363738 F0-F7
7AF8- 00 39 3A
7AFB- 3B 3C 3D
7AFE- 3E 3F     9150        .HS 00393A3B3C3D3E3F F8-FF
                9160 *
7B00-           9170 READ4R .BS $100
7C00-           9180 READ2R .BS $100
7D00-           9190 READ6L .BS $100
7E00-           9200 READ4L .BS $100
7F00-           9210 READ2L .BS $100
8000-           9220 BUFMEM .BS 5469     WRITE PRENIBBLE BUFFER
955D-           9230 BUFEND .EQ *
                9240 *---------------------------------
                9250 *
205D-           9260 ZZSIZE .EQ *-SETUP  PROGRAM SIZE
```

END OF LISTING 2

DOS Device Detective

Article on page 62

Listing 1 for DOS Device Detective

DETECTIVE Source Code

```
SOURCE FILE: DETECTIVE.S
0000:              1 * DETECTIVE
0000:              2 * BY JOHN VOKEY
0000:              3 * COPYRIGHT 1987 BY MICROSPARC,INC.
0000:              4 * CONCORD, MA  01742
0000:              5 * DOS 3.3 TOOLKIT ASSEMBLER
0000:              6 *
----- NEXT OBJECT FILE NAME IS DETECTIVE
9BE3:              7               ORG      $9D00-$11D
9BE3:              8 *==================================
9BE3:              9 ;                Install
9BE3:             10 *==================================
9BE3:             11 ;
9BE3:             12 ;----------------------------------
9BE3:             13 ; To execute the code:
9BE3:             14 ;          BRUN DETECTIVE, A$2000
9BE3:             15 ;----------------------------------
9BE3:             16 ;
9D00:             17 BUFPTR  EQU  $9D00     1st buffer pointer
AA57:             18 BUFCNT  EQU  $AA57     # of buffers loc
A7D4:             19 BUFBLD  EQU  $A7D4     build buffers subroutine
0000:             20 CODLOC  EQU  $0        Temp locs for Create
9BE3:             21 ;
201D:             22 BOOTLOC EQU  $2000+$1D
9BE3:A2 00        23         LDX  #$0       one page to move
9BE5:BD 1D 20     24 LOOP    LDA  BOOTLOC,X
9BE8:9D 00 9C     25         STA  SCHDRV,X
9BEB:E8           26         INX
9BEC:D0 F7        27         BNE  LOOP
9BEE:20 B7 9C     28         JSR  ATTACH    Patch DOS
9BF1:A9 00        29 INSTALL LDA  #>SCHDRV
9BF3:85 00        30         STA  CODLOC
9BF5:A9 9C        31         LDA  #<SCHDRV
9BF7:85 01        32         STA  CODLOC+1
9BF9:A9 03        33         LDA  #3
9BFB:85 02        34         STA  CODLOC+2
9BFD:4C E6 9C     35         JMP  CREATE    move buffers, exit
9C00:             36 ;
9C00:             37 *==================================
9C00:             38 ;             EQUATES
9C00:             39 *==================================
9C00:             40 ;
B1C9:             41 ALLOC   EQU  $B1C9     search VTOC for file
AB28:             42 COMOPEN EQU  $AB28     Common OPEN routine
B5C0:             43 FMDRV   EQU  $B5C0     FileManager drive
B5C1:             44 FMSLT   EQU  FMDRV+1   FileManager slot
AA68:             45 DEFDRV  EQU  $AA68     Default Drive
AA6A:             46 DEFSLT  EQU  DEFDRV+2  Default Slot
B5F7:             47 WASLT   EQU  $B5F7     Work area slot
B5F8:             48 WADRV   EQU  WASLT+1   Work area drive
AB46:             49 BACKIN  EQU  $AB46     OPEN reentry point
00A2:             50 SIGBYTE EQU  $A2       DOS card signature byte
B39B:             51 SREG    EQU  $B39B     File Manager S save
B3AF:             52 HEADER  EQU  $B3AF     "DISK VOLUME"
A54F:             53 INIT    EQU  $A54F     INIT command handler
AB43:             54 PATCH   EQU  $AB43     DOS patch location
9C00:             55 ;
9C00:             56 *==================================
9C00:             57 ;          Search Drives
9C00:             58 *==================================
9C00:             59 ;
9C00:20 68 9C     60 SCHDRV  JSR  LOCFIL    File in VTOC?
9C03:90 10        61         BCC  DONE      Yes, done
9C05:AD C0 B5     62         LDA  FMDRV     No, try other drive
9C08:49 03        63         EOR  #3        complement
9C0A:8D C0 B5     64         STA  FMDRV
9C0D:CD 68 AA     65         CMP  DEFDRV    done both drives?
9C10:F0 12        66         BEQ  NEWSLT    Yes, next slot
9C12:4C 28 AB     67 OUT1    JMP  COMOPEN   No, try again
9C15:             68 ;
9C15:             69 *==================================
9C15:             70 ;            File Found
9C15:             71 *==================================
9C15:             72 ;
9C15:AD C1 B5     73 DONE    LDA  FMSLT     set new defaults
9C18:8D 6A AA     74         STA  DEFSLT
9C1B:AD C0 B5     75         LDA  FMDRV
9C1E:8D 68 AA     76         STA  DEFDRV
9C21:4C 46 AB     77 OUT     JMP  BACKIN    and exit
9C24:             78 ;
9C24:             79 *==================================
9C24:             80 ;               New Slot
9C24:             81 *==================================
9C24:             82 ;
9C24:AD C1 B5     83 NEWSLT  LDA  FMSLT     get current slot
```

```
9C27:4D 6A AA           EOR  DEFSLT    same as default?
9C2A:D0 03              BNE  CHGSLT    No, not 1st pass
9C2C:8D C1 B5           STA  FMSLT     Initialize list
9C2F:            ;
9C2F:            *================================
9C2F:            ;             Change Slot
9C2F:            *================================
9C2F:            ;
9C2F:EE C1 B5    CHGSLT INC  FMSLT     next slot, please
9C32:AD C1 B5           LDA  FMSLT
9C35:CD 6A AA           CMP  DEFSLT    Default slot?
9C38:F0 F5              BEQ  CHGSLT    Yes, try again
9C3A:C9 08              CMP  #8        Any left?
9C3C:B0 0E              BCS  NOTFOUND  No, exit
9C3E:09 C0              ORA  #$C0      check for DOS card
9C40:8D 45 9C           STA  SLFMOD+2  (self-modified code)
9C43:AD 00 C0    SLFMOD LDA  $C000     get sig byte
9C46:C9 A2              CMP  #SIGBYTE  DOS Card?
9C48:F0 C8              BEQ  OUT1      Yes, search drives
9C4A:D0 E3              BNE  CHGSLT    No, next slot
9C4C:            ;
9C4C:            *================================
9C4C:            ;          File not Found
9C4C:            *================================
9C4C:            ;
9C4C:AD 68 AA    NOTFOUND LDA DEFDRV    restore defaults
9C4F:8D C0 B5           STA  FMDRV
9C52:8D F8 B5           STA  WADRV
9C55:AD 6A AA           LDA  DEFSLT
9C58:8D C1 B5           STA  FMSLT
9C5B:0A                 ASL  A
9C5C:0A                 ASL  A
9C5D:0A                 ASL  A
9C5E:0A                 ASL  A
9C5F:8D F7 B5           STA  WASLT     (slot * 16)
9C62:20 68 9C           JSR  LOCFIL    allocate file
9C65:38                 SEC
9C66:B0 B9              BCS  OUT       and exit
9C68:            ;
9C68:            *================================
9C68:            ; Check for File & Trap I/O error
9C68:            *================================
9C68:            ;
9C68:20 72 9C    LOCFIL JSR  GETFIL    force FM to return here
9C6B:AD 81 9C           LDA  SAVSTK    restore true caller
9C6E:8D 9B B3           STA  SREG
9C71:60                 RTS
9C72:            ;
9C72:8A          GETFIL TXA            save X
9C73:AE 9B B3           LDX  SREG      get stashed stack pointer
9C76:8E 81 9C           STX  SAVSTK    save
9C79:BA                 TSX            force LOCFIL as return
9C7A:8E 9B B3           STX  SREG
9C7D:AA                 TAX            recover X
9C7E:4C C9 B1           JMP  ALLOC     and attempt to locate file
9C81:00          SAVSTK DFB  0
9C82:            ;
9C82:            *================================
9C82:            ;    Disconnect Patch on INIT
9C82:            *================================
9C82:            ;
9C82:20 88 9C    INITFIX JSR DISCON    disconnect Patch
9C85:4C 4F A5           JMP  INIT      and do INIT command
9C88:            ;
9C88:            *================================
9C88:            ;         Disconnect Patch
9C88:            *================================
9C88:            ;
9C88:A2 0B       DISCON LDX  #11       restore original header
9C8A:BD A5 9C    LOOP2  LDA  MSGE,X
9C8D:9D AF B3           STA  HEADER,X
9C90:CA                 DEX
9C91:10 F7              BPL  LOOP2
9C93:A2 02              LDX  #2        restore DOS code
9C95:BD B1 9C    LOOP3  LDA  FIX1,X
9C98:9D 43 AB           STA  PATCH,X
9C9B:BD B4 9C           LDA  FIX2,X
9C9E:9D 4F A5           STA  INIT,X
9CA1:CA                 DEX
9CA2:10 F1              BPL  LOOP3
9CA4:                   MSB  ON
9CA4:                   LST  ON,GEN
9CA4:60                 RTS            and return
9CA5:A0 C5 CD    MSGE   ASC  "         EMULOV KSID"
9CA8:D5 CC CF
9CAB:D6 A0 CB
9CAE:D3 C9 C4
9CB1:20 C9 B1    FIX1   DFB  $20,$C9,$B1
9CB4:A9 40 20    FIX2   DFB  $A9,$40,$20
9CB7:            ;
9CB7:            *================================
9CB7:            ;           Attach Patch
9CB7:            *================================
9CB7:            ;
9CB7:A2 0B       ATTACH LDX  #11       new header
9CB9:BD D4 9C    LOOP4  LDA  MSGE2,X
9CBC:9D AF B3           STA  HEADER,X
9CBF:CA                 DEX
9CC0:10 F7              BPL  LOOP4
9CC2:A2 02              LDX  #2        Patch DOS code
9CC4:BD E0 9C    LOOP5  LDA  PCH1,X
9CC7:9D 43 AB           STA  PATCH,X
9CCA:BD E3 9C           LDA  PCH2,X
9CCD:9D 4F A5           STA  INIT,X
9CD0:CA                 DEX
9CD1:10 F1              BPL  LOOP5
9CD3:60                 RTS            and return
9CD4:A0 A0 A0    MSGE2  ASC  "           EVITCETED"
9CD7:C5 D6 C9
9CDA:D4 C3 C5
9CDD:D4 C5 C4
9CE0:4C          PCH1   DFB  $4C
9CE1:00 9C              DW   SCHDRV
9CE3:4C          PCH2   DFB  $4C
9CE4:82 9C              DW   INITFIX
9CE6:            ;
9CE6:            *================================
9CE6:            ;           Create Space
9CE6:            *================================
9CE6:            ;
9CE6:A5 00       CREATE LDA  CODLOC    get low byte of space
9CE8:38                 SEC
9CE9:E9 26              SBC  #$26      name, pointers, etc.
9CEB:8D 00 9D           STA  BUFPTR
9CEE:A5 01              LDA  CODLOC+1
9CF0:E9 00              SBC  #0
9CF2:8D 01 9D           STA  BUFPTR+1
9CF5:A5 02              LDA  CODLOC+2  get number of buffers
9CF7:8D 57 AA           STA  BUFCNT
9CFA:4C D4 A7           JMP  BUFBLD    rebuild buffers, exit

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

END OF LISTING 1

Note: Key Perfect addresses do not match Listing 1.

```
                KEY PERFECT 5.0
                    RUN ON
                   DETECTIVE
   ========================================
   CODE-5.0     ADDR# - ADDR#     CODE-4.0
   --------     -------------     --------
   B5E9665F     2000 - 204F         2769
   9B77ECC7     2050 - 209F         2CD8
   78946536     20A0 - 20EF         2975
   9BC54560     20F0 - 2119         1476
   C630353C = PROGRAM TOTAL =       011A
```

Listing 2 for DOS Device Detective

DETECTIVE Hex Listing

```
2000- A2 00 BD 1D 20 9D 00 9C
2008- E8 D0 F7 20 B7 9C A9 00
2010- 85 00 A9 9C 85 01 A9 03
2018- 85 02 4C E6 9C 20 68 9C
2020- 90 10 AD C0 B5 49 03 8D
2028- C0 B5 CD 68 AA F0 12 4C
2030- 28 AB AD C1 B5 8D 6A AA
2038- AD C0 B5 8D 68 AA 4C 46
2040- AB AD C1 B5 4D 6A AA D0
2048- 03 8D C1 B5 EE C1 B5 AD
2050- C1 B5 CD 6A AA F0 F5 C9
2058- 08 B0 0E 09 C0 8D 45 9C
2060- AD 00 C0 C9 A2 F0 C8 D0
2068- E3 AD 68 AA 8D C0 B5 8D
2070- F8 B5 AD 6A AA 8D C1 B5
2078- 0A 0A 0A 0A 8D F7 B5 20
2080- 68 9C 38 B0 B9 20 72 9C
2088- AD 81 9C 8D 9B B3 60 8A
2090- AE 9B B3 8E 81 9C BA 8E
2098- 9B B3 AA 4C C9 B1 00 20
20A0- 88 9C 4C 4F A5 A2 0B BD
20A8- A5 9C 9D AF B3 CA 10 F7
20B0- A2 02 BD B1 9C 9D 43 AB
20B8- BD B4 9C 9D 4F A5 CA 10
20C0- F1 60 A0 C5 CD D5 CC CF
20C8- D6 A0 CB D3 C9 C4 20 C9
20D0- B1 A9 40 2D A2 0B BD D4
20D8- 9C 9D AF B3 CA 10 F7 A2
20E0- 02 BD E0 9C 9D 43 AB BD
20E8- E3 9C 9D 4F A5 CA 10 F1
20F0- 60 A0 A0 A0 C5 D6 C9 D4
20F8- C3 C5 D4 C5 C4 4C 00 9C
2100- 4C 82 9C A5 00 38 E9 26
2108- 8D 00 9D A5 01 E9 00 8D
2110- 01 9D A5 02 8D 57 AA 4C
2118- D4 A7
```

END OF LISTING 2

Listing 3 for DOS Device Detective

DETECTIVE.DEMO

```
10  REM *********************
20  REM *   DETECTIVE.DEMO   *
30  REM *  BY JOHN R. VOKEY  *
40  REM * COPYRIGHT (C) 1987 *
50  REM * BY MICROSPARC, INC *
60  REM * CONCORD, MA  01742 *
70  REM *********************
80  REM DISPLAY TITLE PAGE
90  PRINT  CHR$ (14); CHR$ (21): HOME :DRIVE =
     43624:SLOT = DRIVE + 2
100  COLOR= 2: GOSUB 470
110  POKE 33,38: POKE 32,1: POKE 34,1: POKE 3
     5,23
120  FOR I = 5 TO 21: READ S$
130  FOR J = 23 TO I STEP  - 1
140  VTAB J: GOSUB 490
150  NEXT : NEXT
160  DATA    DOS DEVICE DETECTIVE,DEVICE-INDE
     PENDENT DOS,BY JOHN VOKEY,.......,COPYRIG
     HT (C) 1987
170  DATA  MICROSPARC INC.
180  DATA  CONCORD MA 01742
190  DATA  ....
200  REM INSTALL PATCH
210  PRINT : PRINT  CHR$ (4)"BRUN DETECTIVE,A
     $2000"
220  VTAB 10: HTAB 12: PRINT "<PATCH INSTALLE
     D>"
230  REM DELAY FOR 1000 OR KEY
240  VTAB 24: HTAB 15: INVERSE
250  PRINT "PRESS <RETURN>";: NORMAL : POKE  -
     16368,0: FOR I = 1 TO 500: IF  PEEK ( -
     16384) < 128 THEN  NEXT
260  REM DISPLAY INSTRUCTIONS
270  VTAB 7: CALL  - 958: FOR I = 9 TO 12: READ
     S$: FOR J = 23 TO I STEP  - 1: VTAB J
280  GOSUB 490
290  NEXT : NEXT : VTAB 24: HTAB 15: INVERSE
     : PRINT "              ";: NORMAL : REM
      14 SPACES
300  DATA  PLEASE INSERT THE DETECTIVE DISK
310  DATA  INTO ANY DRIVE ON THE SYSTEM,(OR N
     OT AT ALL!)
320  DATA THEN PRESS <RETURN>
330  ONERR  GOTO 510
340  POKE  - 16368,0
350  REM AWAIT KEYPRESS
360  VTAB 13: HTAB 19: GET S$: IF S$ <  >  CHR$
     (13) AND S$ <  >  CHR$ (27) THEN 360
370  IF S$ =  CHR$ (27) THEN 450
380  REM SEARCH FOR FILE
390  PRINT : IF  NOT ERR THEN  PRINT  CHR$ (4
     )"VERIFY DETECTIVE"
400  IF ERR THEN  VTAB 20: HTAB 6: PRINT  CHR$
     (7);"DETECTIVE IS NOT ON THE SYSTEM"
410  IF  NOT ERR THEN  VTAB 20: HTAB 6: PRINT
      CHR$ (7);"DETECTIVE IS IN SLOT " PEEK (
     SLOT)", DRIVE " PEEK (DRIVE)
420 ERR = 0: VTAB 24: HTAB 15: INVERSE : PRINT
     "<ESC> TO EXIT ";: NORMAL
430  GOTO 360
440  REM EXIT
450  POKE  - 16368,0: TEXT : HOME : POKE 216,
     0: END
460  REM FRAME SUBROUTINE
470  HLIN 0,39 AT 1: FOR K = 1 TO 47 STEP 2: PLOT
     0,K: PLOT 39,K: NEXT : HLIN 0,39 AT 47: RETURN

480  REM PRINT SUBROUTINE
490  HTAB (41 -  LEN (S$)) / 2: PRINT S$;: CALL
      - 958: RETURN
500  REM ON ERR TRAP
510 ERR =  PEEK (222): RESUME
```

END OF LISTING 3

A Matter of Timing

Article on page 70

Listing 1 for A Matter of Timing

CLOCK.TEST

```
                         ;
                         **************************************************
                         *                   CLOCK.TEST                   *
                         *             by S. Scott Zimmerman              *
                         *              Copyright (c) 1987                *
                         *              by MicroSPARC, Inc                *
                         *              Concord, MA  01742                *
                         **************************************************
                         * MicroSPARC Assembler 3.0
                                  ORG $300
                         TINDX    EQU $19          ;Time loop index
                         WAIT     EQU $FCA8        ;Pause accum amount
                         BELL     EQU $FF3A        ;Beep routine

                         TIMCNT   EQU $B511        ;Inner loop time count

0300  A0 0A                       LDY #10          ;Make a pause before
0302  A9 FF              PAUSLOOP LDA #$FF         ; starting test
0304  20 A8 FC                    JSR WAIT         ;Go wait a while
0307  88                          DEY              ;End of pause loop?
0308  10 F8                       BPL PAUSLOOP     ;No, go loop again
030A  20 3A FF                    JSR BELL         ;Yes, sound start beep
030D  A0 05                       LDY #5           ;Set index for "seconds"
030F  A9 11              SECLOOP  LDA #TIMCNT      ;Set the time loop index
0311  85 19                       STA TINDX        ; to TIMCNT
0313  A9 B5                       LDA #TIMCNT/
0315  85 1A                       STA TINDX+1
0317  EA                 TIMELOOP NOP              ;Empty loop
0318  A5 19                       LDA TINDX        ;Do a 16-bit (double
031A  D0 02                       BNE DECINDX      ; precision) decrement
031C  C6 1A                       DEC TINDX+1      ;Dec HOB as needed
031E  C6 19              DECINDX  DEC TINDX        ;Always dec LOB
0320  A5 19                       LDA TINDX        ;Is the time index zero?
0322  05 1A                       ORA TINDX+1
0324  D0 F1                       BNE TIMELOOP     ;No, so loop again
0326  88                          DEY              ;Yes, End of "sec" loop?
0327  D0 E6                       BNE SECLOOP      ;No, loop again
0329  4C 3A FF                    JMP BELL         ;Yes, so do end beep

000  Errors

0300  Hex Start of Object
032B  Hex end of Object
002C  Hex Length of Object
7886  Hex end of Symbols
```

END OF LISTING 1

Listing 2 for A Matter of Timing

AMPER.MUSIC

```
                         ;
                         **************************************************
                         *                                                *
                         *                  AMPER.MUSIC                   *
                         *                                                *
                         *             by S. Scott Zimmerman              *
                         *              Copyright (c) 1987                *
                         *              by MicroSPARC, Inc                *
                         *              Concord, MA  01742                *
                         *            MicroSPARC Assembler 3.0            *
                         **************************************************

                                  ORG $2E4         ;Start in input buffer

                         **************************************************
                         *  Zero-page EQUates and constants:              *
                         **************************************************

                         PITCH    EQU $19          ;Pitch parameter
                         DURATION EQU $1A          ;Duration parameter
                         FREQ     EQU $1B          ;Frequency
                         DURCNT   EQU $1C          ;Duration count, snd loop
                         STOPFLG  EQU $1E          ;Stop flag for no sound

                         RESUMTOK EQU $A6          ;Applesoft RESUME token
                         STOPTOK  EQU $B3          ;Applesoft STOP token

```