PRODOS SPY

Keep an eye ***on ProDOS. What machine language interface calls are used, which disk blocks are accessed and where in Apple memory does disk information go?***

ProDOS SPY is a new type of trace utility to help you find out what a program is doing when it uses a disk drive. This is useful in analyzing and debugging assembly language programs (yours and others'), as well as in analyzing BASIC programs and some types of commercial programs. ProDOS Spy does not interfere with the operation of your disk drive; every time your program uses the drive, it reports what the program is trying to do and what disk sector and part of memory is involved.

USING THE PROGRAM

When you type the command BRUN SPY, ProDOS Spy will install itself between ProDOS and the ProDOS file buffers, where it will be out of the way of BASIC and most assembly language programs. After a title and copyright notice are displayed, the program is ready for use. ProDOS Spy is compatible with other programs that install themselves similarly, as long as those programs follow the Apple guidelines for added ProDOS commands.

When you type a ProDOS command, the command will execute normally, but as it executes, a list of machine language interface (MLI) commands and numbers appear on the screen. (I will interpret this information below; for now it's enough to say that it involves exactly what ProDOS does to execute the command you typed.) To save the information listed by ProDOS Spy, turn on your printer with PR#1, and ProDOS Spy will send the information to it. (This works with most printers and interfaces, but some may disrupt the disk timing and cause an I/O error. If this occurs, you might try disabling your interface's buffer; otherwise you'll have to rely on the screen display.)

To shut off ProDOS Spy temporarily, use the command UNSPY. ProDOS Spy replies with the message "SPY OFF," and ceases tracing commands. When you want ProDOS Spy back on again, type the command SPY. ProDOS Spy then replies with the message "SPY ON," and comes back to life.

BACKGROUND

To interpret what ProDOS Spy tells you, you need some background information (see the references at the end of this article). **Figure 1** shows a diagram of the structure of ProDOS and how ProDOS Spy fits into it. ProDOS is composed of three sections:

1. The BASIC interpreter (BI)
2. The kernel and machine language interface (MLI)
3. Device drivers

The BI is responsible for interpreting the ProDOS commands used from the keyboard or from BASIC programs. It checks these for syntax and breaks them down into sim-

TABLE 1: Modifications for Drives in Slots 4, 5, 6 and 7

	Slot			
Locations	4	5	6	7
$20FA	$18	$1A	$1C	$1E
$2118	$18	$1A	$1C	$1E
$2237	$18	$1A	$1C	$1E
$225A	$18	$1A	$1C	$1E
$2100	$19	$1B	$1D	$1F
$2126	$19	$1B	$1D	$1F
$2240	$19	$1B	$1D	$1F
$2260	$19	$1B	$1D	$1F
$2106	$28	$2A	$2C	$2E
$211B	$28	$2A	$2C	$2E
$223A	$28	$2A	$2C	$2E
$2266	$28	$2A	$2C	$2E
$210C	$29	$2B	$2D	$2F
$2129	$29	$2B	$2D	$2F
$2243	$29	$2B	$2D	$2F
$226C	$29	$2B	$2D	$2F

pler commands that can be handled by the other parts of ProDOS. The BI understands volume names and directories, and it identifies files exclusively by their file names.

The MLI is responsible for executing the sequences of commands sent to it by the BI or other machine language programs. It breaks these commands down into sequences of even simpler commands that can be handled by the device drivers. Although some of its commands identify files by their names, many identify files only by a file number assigned when that file is opened.

The device drivers for disks are quite simple-minded routines. They can transfer 512 bytes (one block) of information from memory to the disk or from the disk to memory. Except for detecting a write-protect tab or an empty disk drive, that's about all they can do.

So, to summarize, the BI accepts commands from the keyboard or a BASIC program and sends one or more (usually several) calls to the MLI. For each call that it receives, the MLI calls the appropriate device driver one or more times. As shown in **Figure 1**, ProDOS Spy installs itself so that it can intercept calls from the MLI to the slot 6 device drivers. It prints something in response to every device driver call that it intercepts. (We will consider the details of how that is done later on.) With this information in mind, we can begin to interpret a ProDOS Spy report.

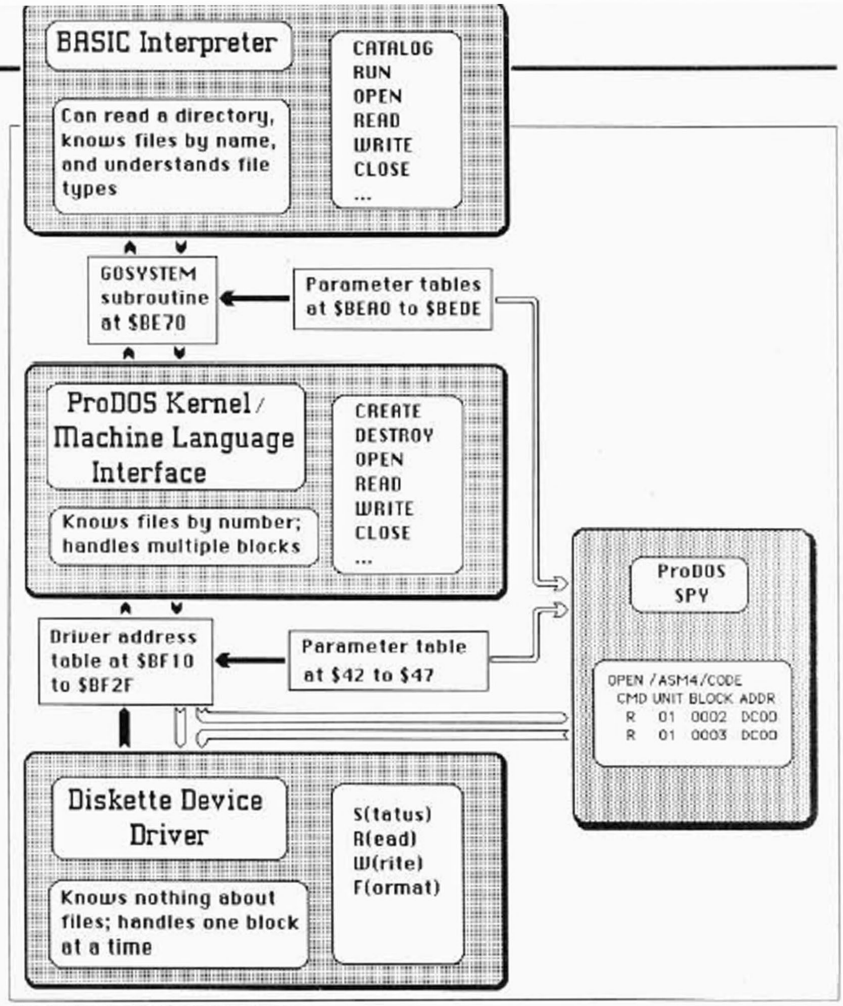

FIGURE 1: ProDOS Structure

INTERPRETATION OF REPORTS

A ProDOS Spy report gives the name of every MLI command being executed. If the command involves a file name, the report also gives the full pathname. Underneath each MLI command, indented, is a list of all the device driver calls used to execute that command.

For each device driver command, four items of information are reported:

1. CMD — The device driver command code. It is either R (for reading data from the disk), W (for writing data to the disk) or S (to determine the status of the device). Another code, F (for format), is possible, but this is not supported by Disk II devices. You may see this if you use a UniDisk or hard disk in slot 6.
2. UNIT — The drive number, either 1 or 2.
3. BLOCK — The number of the 512-byte data block on the disk that is either being read or written. The number is in hexadecimal notation, and for Disk II devices, it will be in the range 0-$117.
4. ADDR — The beginning address (in hexadecimal) of a 512-byte block of memory that supplies the data for writing or accepts the data from reading. With HIMEM in its usual $9600 location, most file transfer operations such as LOAD, BLOAD, SAVE and BSAVE use a buffer at $9200 to transfer the first block of the actual data. If a file extends more than one block, then subsequent blocks are transferred directly to and from memory. Most other operations, such as ON LINE, GET INFO and portions of the file transfer operations, use buffers at $DA00 and $DC00.

Let us consider some examples of ProDOS Spy reports. I made these by sending the ProDOS Spy report to the printer and then adding some comments.

First, let us consider a simple catalog listing (see **Example 1**). Notice that all of the device driver commands are Read commands; you wouldn't expect to write to the disk for a catalog. Notice also that the disk was in drive 1 (UNIT 1).

Block 2 always has the volume entry with the volume name and the size of the volume. Blocks 2-5 have all the rest of the catalog information; a search for a file name always starts with block 2 and proceeds sequentially through the other four blocks. Block 6 is the volume bit map, which shows which blocks of the volume are free. Finally, the internal buffer (BI 1.1) at $DC00 is used; earlier versions may have their internal buffer at a different address.

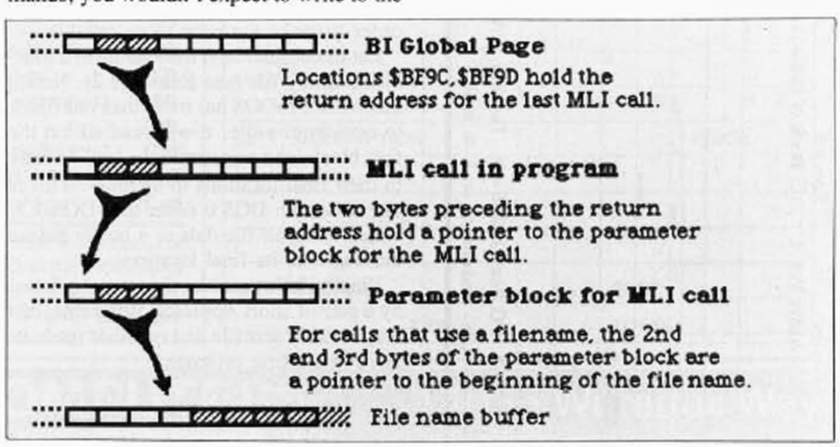

FIGURE 2: Finding the File Name

EXAMPLE 1: Simple Catalog Listing

```
]CAT                          I typed this command
ON LINE                       The start of the Spy report
  CMD UNIT  BLOCK  ADDR
   R   01   0002   DC00       Get the volume name
GET INFO /UTILITIES/
  CMD UNIT  BLOCK  ADDR
   R   01   0002   DC00       Check the file type
   R   01   0006   DC00         and access allowed
OPEN     /UTILITIES
   CMD UNIT  BLOCK  ADDR
   R   01   0002   DC00       Prepare to read directory
   R   01   0002   9300       Directory to system buffer

/UTILITIES                    This comes from ProDOS, not Spy

 NAME            TYPE  BLOCKS  MODIFIED

*STARTUP          BAS      4  15-MAR-84  This is
*SU               BAS     35  15-MAR-84    just
*SU1.OBJ          BIN     27  15-MAR-84    the
*SU2.OBJ          BIN     10  15-MAR-84    usual
*SU3.OBJ          BIN     61  15-MAR-84    CAT
*SU4.OBJ          VAR     18  15-MAR-84    listing
*PRODOS           SYS     31  15-MAR-84
*BASIC.SYSTEM     SYS     21  15-MAR-84

ON LINE
  CMD UNIT  BLOCK  ADDR       Get the volume
   R  01    0002   DC00         name (again?)
GET INFO /UTILITIES
  CMD UNIT  BLOCK  ADDR       Get the number of blocks
   R  01    0002   DC00         total and number used
   R  01    0006   DC00
BLOCKS FREE:   66     BLOCKS USED:  214   Last line
                                           of the CAT
```

Notice that ProDOS reads the same blocks several times, even when the information could not possibly have changed. I presume that Apple sacrificed some efficiency in order to make the code more reliable.

Let us consider next the loading of a four-block binary file (see **Example 2**). Notice that when ProDOS has more than one block to read from a file, it will read all but the first block (and sometimes the last) directly to their final locations in memory. This is one reason ProDOS is faster than DOS 3.3, which reads all file data to a buffer before moving it to its final location.

Finally, let's consider the trace produced by a pair of short Applesoft programs, one that writes a text file and one that reads it. Here is the first program:

```
10 D$ = CHR$ (4):N$ = "TEST"
20  PRINT D$"OPEN"N$: PRINT D$"
    WRITE"N$
30  FOR I = 1 TO 5: PRINT "NIBBLE
    ": NEXT
40  PRINT D$"CLOSE"
```

Example 3 shows its trace. Notice that nothing is actually written to disk until the CLOSE command is received. That is because so little was written that the file buffer is able to hold it all.

Here is the program that reads the text file:

```
10 D$ = CHR$ (4):N$ = "TEST"
20  PRINT D$"OPEN"N$: PRINT D$"
    READ"N$
30  FOR I = 1 TO 5: INPUT A$:
    PRINT A$: NEXT
40  PRINT D$"CLOSE"
```

Example 4 shows the corresponding trace. The CLOSE command did not show up because it didn't cause anything to be written to the disk, and it therefore didn't cause a call to the device driver. This is generally true — if an MLI call does not actually cause a device driver call, it won't be reported by ProDOS Spy because Spy never gets control.

ENTERING THE PROGRAM

The listings for "ProDOS Spy" can be found in the Program Listings section at the end of the magazine. Type in SPY (**Listing 1**). Notice that the published listing was made with the ProDOS Assembler by Apple. If you use a different assembler, you may have to change some of the pseudo-opcodes (like DB or DS). For an explanation of pseudo-ops, consult your assembler documentation.

If you do not use an assembler, save the program with the command:

BSAVE SPY,A$207B,L$4A8

If you are using Key Perfect, be sure that the spaces reserved by DS pseudo-ops are zeroed. First BLOAD SPY and enter the Monitor with CALL −151. Then key in the following Monitor commands and BSAVE the program as shown above:

```
21FE:00 00
2419:00 00 00 00 00 00
2423:00 00 00 00
```

For help with entering *Nibble* listings, see "A Welcome to New *Nibble* Readers" at the beginning of this issue.

HOW PRODOS SPY WORKS

As I mentioned above, the BI accepts commands from the keyboard or a BASIC program and sends one or more calls to the MLI. The MLI calls the appropriate device driver one or more times. Each of these calls has a strictly defined form:

JSR $BF00
DFB Function code
DW Parameter Block Address

The address $BF00 is the entrance point to the MLI. The function code is a number that indicates which of the 26 MLI functions is being called. The Parameter Block Address is the address of the beginning of a block of data needed for that particular function. This block varies from two to twelve bytes in size and contains such things as the file number, file buffer address or address of the file name. When the MLI finishes executing the call, it returns control to the instruction following the Parameter Block Address.

Calls to the device drivers are simply a JSR to the driver address. Since different drivers may be installed in any particular system, ProDOS keeps a table of the addresses of installed drivers at $BF10-$BF2F. Calls to a device driver also involve a parameter table, but it is always the same size and it is always at addresses $42-$47.

This block supplies most of the information reported by Spy. Location $42 contains the command code: 0, 1, 2 or 3 for status, read, write and format, respectively. Location $43 contains the unit number, and for

```
]BLOAD CYPHER                    My command
ON LINE                          Start of the Spy trace
  CMD UNIT  BLOCK  ADDR
   R   Ø1   0002   DC00          Get volume name
GET INFO /ASM4/CYPHER
  CMD UNIT  BLOCK  ADDR
   R   Ø1   0002   DC00          Find file entry (on
   R   Ø1   0003   DC00            block 3)
OPEN     /ASM4/CYPHER
   CMD UNIT  BLOCK  ADDR
   R   Ø1    0002   DC00         Get file info
   R   Ø1    0003   DC00           (again?)
   R   Ø1    0111   9500         Block list to buffer
   R   Ø1    0110   9300         First data block to buffer
READ       FILE Ø
  CMD UNIT  BLOCK  ADDR          The next 3 data blocks are
   R  Ø1  Ø112   2200              read directly to their
   R  Ø1  Ø113   2400              final memory location,
   R  Ø1  Ø114   2600              bypassing the buffer.
```

EXAMPLE 2: Loading a Four-Block Binary File

```
ON LINE
  CMD UNIT  BLOCK  ADDR
   R   Ø1   0002   DC00          Get volume name
GET INFO /ASM4/TEST
  CMD UNIT  BLOCK  ADDR
   R   Ø1   0002   DC00          Get file info
OPEN     /ASM4/TEST
   CMD UNIT  BLOCK  ADDR
   R   Ø1    0002   DC00         Read directory
   R   Ø1    010F   9300         Read first block
WRITE      FILE 1
  CMD UNIT  BLOCK  ADDR          The next 3 data blocks are
   S  Ø1    0000   9300         Is writing allowed?
CLOSE      FILE 1
  CMD UNIT  BLOCK  ADDR
   W   Ø1   010F   9300          Write the file
   R   Ø1   0002   DC00          Read file info
   R   Ø1   0002   DC00
   W   Ø1   0002   DC00          Writes modified file info
```

EXAMPLE 3: Trace for a Program That Writes a Text File

```
ON LINE
  CMD UNIT  BLOCK  ADDR
   R   Ø1   0002   DC00          Volume name, as usual
GET INFO /ASM4/TEST
  CMD UNIT  BLOCK  ADDR
   R   Ø1   0002   DC00          File info, as usual
OPEN     /ASM4/TEST
   CMD UNIT  BLOCK  ADDR
   R   Ø1    0002   DC00
   R   Ø1    010F   9300         Block 1 to file buffer
NIBBLE
NIBBLE
NIBBLE                           From the program, not from Spy
NIBBLE
NIBBLE
```

EXAMPLE 4: Trace for a Program That Reads a Text File

slot 6 devices, it will be either $60 for drive 1 or $E0 for drive 2. Finally, the beginning of the memory range involved in the command is stored in memory locations $44 and $45, and the disk block number is stored in $46 and $47.

When ProDOS Spy gets control at the beginning of a device driver call, it first prints out the MLI call that led to that device driver call. Fortunately, when the MLI receives a call, it stores the return address in the system global page, at locations $BF9C and $BF9D.

As I mentioned above, the return point for an MLI call is five bytes after the call itself. Between the call and the return point are a code for the MLI command and an address for the parameter block. Spy recovers the command code and uses it to print the appropriate command name.

To avoid repetitive printing of the same MLI information for several device driver calls, Spy prints the MLI information only if the MLI command has changed since the last call.

Some MLI calls use file names or file numbers. Spy uses a similar trick for these as it did for the MLI command code. It reads $BF9C,$BF9D to find the return point for the MLI call; two bytes before that point it reads the address of the parameter block. For calls that use a file number, the number is one byte after the beginning of the parameter block. For calls that use a file name, a pointer to the file name is one byte after the beginning of the parameter block. (Actually, this pointer refers to a byte giving the length of the file name, and the file name itself follows immediately.) That is, ProDOS Spy follows a trail of two pointers (for file numbers) or three pointers (for file names) before it gets the information it wants. **Figure 2** illustrates the process for file names. If this isn't indirect addressing, nothing is!

MODIFICATIONS

You can modify ProDOS Spy to work with disk drives connected to other slots. This involves modifying each reference to the addresses DEVADR61, DEVADR61+1, DEVADR62 and DEVADR62+1 (addresses $BF1C, $BF1D, $BF2C and $BF2D, respectively). To do this, BLOAD SPY, modify the locations according to **Table 1**, and save Spy before running it with the command:

BSAVE SPY,A$207B,L$4A8

REFERENCES

1. Apple Computer, Inc., *ProDOS Technical Reference Manual*. Addison-Wesley, Reading, MA. 1985.
2. Mossberg, Sandy. "ProDOS BASIC's Global Page." *Nibble*, Vol. 6/No. 1, p. 96.
3. Worth, Don, and Pieter Lechner. *Beneath Apple ProDOS*. Quality Software, Chatsworth, CA, 1984.

Listing 1 for ProDOS Spy

SPY

```
0000:                    1          LST   ON,NOA,G
0000:                    2 ********************************
0000:                    3 *                              *
0000:                    4 *             SPY              *
0000:                    5 *          by Ken Manly        *
0000:                    6 *      Buffalo Chip Software   *
0000:                    7 *                              *
0000:                    8 *      Copyright (C)  1987     *
0000:                    9 *      by Microsparc, Inc.     *
0000:                   10 *      Concord, MA.  01742     *
0000:                   11 *                              *
0000:                   12 ********************************
0000:                   13 *    EDASM SYSTEM Assembler    *
0000:                   14 ********************************
0000:                   15 ;
0000:          002F     16 LENGTH    EQU   $2F
0000:          0036     17 CSWL      EQU   $36
0000:          003A     18 PCL       EQU   $3A
0000:          003C     19 A1L       EQU   $3C
0000:          003E     20 A2L       EQU   $3E
0000:          0042     21 A4L       EQU   $42
0000:          0042     22 COMMAND   EQU   $42
0000:          0043     23 UNITNUM   EQU   $43
0000:          0044     24 BUFPTR    EQU   $44
0000:          0046     25 BLOCKNUM  EQU   $46
0000:          005E     26 INDEX     EQU   $5E
0000:          0073     27 HIMEM     EQU   $73
0000:          0200     28 INPUT     EQU   $200
0000:          BE06     29 EXTRNCMD  EQU   $BE06
0000:          BE50     30 XTRNADDR  EQU   $BE50
0000:          BE53     31 XCNUM     EQU   $BE53
0000:          BE54     32 PBITS     EQU   $BE54
0000:          BE9E     33 XRETURN   EQU   $BE9E
0000:          BCBC     34 TXBUF     EQU   $BCBC
0000:          BE30     35 VECTOUT   EQU   $BE30
0000:          BE85     36 SYSCALL   EQU   $BE85
0000:          BEC7     37 SREFNUM   EQU   $BEC7
0000:          BEF5     38 GETBUFR   EQU   $BEF5
0000:          BEFB     39 BIHIMEM   EQU   $BEFB
0000:          BF1C     40 DEVADR61  EQU   $BF1C
0000:          BF2C     41 DEVADR62  EQU   $BF2C
0000:          BF58     42 BITMAP    EQU   $BF58
0000:          BF98     43 MACHID    EQU   $BF98
0000:          BF9C     44 CMDADR    EQU   $BF9C
0000:          C000     45 KBD       EQU   $C000
0000:          C010     46 KBSTROBE  EQU   $C010
0000:          C030     47 CLICK     EQU   $C030
0000:          C083     48 RAMRD2    EQU   $C083
0000:          C082     49 ROMRD     EQU   $C082
0000:          C08B     50 RAMRD1    EQU   $C08B
0000:          D000     51 BANKID    EQU   $D000
0000:          F88C     52 INSDS2    EQU   $F88C
0000:          F948     53 PRBLNK    EQU   $F948
0000:          F953     54 PCADJ     EQU   $F953
0000:          FCA8     55 WAIT      EQU   $FCA8
0000:          FD75     56 NXTCHAR   EQU   $FD75
0000:          FD8E     57 CROUT     EQU   $FD8E
0000:          FDDA     58 PRBYTE    EQU   $FDDA
0000:          FDED     59 COUT      EQU   $FDED
0000:          FE2C     60 MOVE      EQU   $FE2C
0000:                   61 ;
0000:                   62          NSB   ON
207B:          207B     63          ORG   $207B
207B:                   64 ;
207B:D8                 65          CLD
207C:A9 04              66          LDA   #<LAST-BEGIN+$100 ;Ask BASIC to
207E:20 F5 BE           67          JSR   GETBUFR         ; reserve space
2081:8D FE 21           68          STA   ORIG            ;Remember where it is
2084:85 3C              69          STA   A1L             ;Temp for protection routine
2086:38                 70          SEC
2087:E9 22              71          SBC   #<BEGIN         ;Subtract present location
2089:8D FF 21           72          STA   RELO            ; to see how far program
208C:A9 04              73          LDA   #<LAST-BEGIN+$100 ; will have to be moved
208E:85 3D              74          STA   A1L+1           ;No. pages to be protected
2090:A5 3C              75 PROTECT   LDA   A1L             ;Protect program space
2092:20 5C 21           76          JSR   SETBIT          ; from ProDOS by setting
2095:E6 3C              77          INC   A1L             ; bits in the bitmap
2097:C6 3D              78          DEC   A1L+1           ;Count down number of pages
2099:D0 F5    2090      79          BNE   PROTECT
209B:                   80 ;
209B:AD 08 BE           81          LDA   EXTRNCMD+2      ;Link SPY into the
209E:8D 2F 22           82          STA   TRYNEXT+3       ; chain of external
20A1:AD 07 BE           83          LDA   EXTRNCMD+1      ; commands, if any
20A4:8D 2E 22           84          STA   TRYNEXT+2
20A7:18                 85          CLC
20A8:A9 22              86          LDA   #<BEGIN
20AA:6D FF 21           87          ADC   RELO            ;If SPY cannot
20AD:8D 08 BE           88          STA   EXTRNCMD+2      ; recognize a command,
2080:A9 00              89          LDA   #>BEGIN         ; it will pass control
20B2:8D 07 BE           90          STA   EXTRNCMD+1      ; to next routine
20B5:                   91 ;
20B5:A5 74              92          LDA   HIMEM+1         ;Make present HIMEN
20B7:8D FB BE           93          STA   BIHIMEM         ; semi-permanent
20BA:                   94 ;
20BA:A9 00              95          LDA   #>BEGIN         ;To start relocation,
20BC:85 3A              96          STA   PCL             ; point program
20BE:A9 22              97          LDA   #<BEGIN         ; counter at beginning
20C0:85 3B              98          STA   PCL+1           ; of program
20C2:A2 00              99 FIXLOOP   LDX   #0
20C4:20 BC F8          100          JSR   INSDS2          ;Disassemble an opcode
20C7:B1 3A             101          LDA   (PCL),Y         ;Check opcode
20C9:F0 28    20F3     102          BEQ   FIXADOR         ;BRK means end of code
20CB:A4 2F             103          LDY   LENGTH          ;Only 3-byte
20CD:C0 02             104          CPY   #2              ; instructions need
20CF:D0 10    20E1     105          BNE   FX1             ; fixing
20D1:B1 3A             106          LDA   (PCL),Y         ;Only instructions
20D3:C9 22             107          CMP   #<BEGIN         ; referring to address
20D5:90 0A    20E1     108          BCC   FX1             ; within program
20D7:C9 26             109          CMP   #<LAST+$100     ; need fixing
20D9:B0 06    20E1     110          BCS   FX1
20DB:18                111          CLC                   ;Fix by adding RELO
20DC:6D FF 21          112          ADC   RELO            ; offset to hi byte
20DF:91 3A             113          STA   (PCL),Y         ; of address
20E1:20 53 F9          114 FX1      JSR   PCADJ           ;Next address
20E4:85 3A             115          STA   PCL             ;Move program counter
20E6:84 3B             116          STY   PCL+1           ; to next instruction
20E8:4C C2 20          117          JMP   FIXLOOP
20EB:                  118 ;
20EB:18                119 FXTBLP   CLC                   ;Add RELO offset to
20EC:6D FF 21          120          ADC   RELO            ; hi byte of each
20EF:99 17 24          121          STA   ATBL,Y          ; address in the table
20F2:C8                122          INY                   ;Next address
20F3:C8                123 FIXADOR  INY
20F4:B9 17 24          124          LDA   ATBL,Y          ;0 means end of
20F7:D0 F2    20EB     125          BNE   FXTBLP          ; table
20F9:                  126 ;
20F9:AD 1C BF          127          LDA   DEVADR61        ;Save the addresses
20FC:8D 1B 24          128          STA   DEV1SV          ; for the slot 6
20FF:AD 1D BF          129          LDA   DEVADR61+1      ; device drivers--
2102:8D 1C 24          130          STA   DEV1SV+1        ; they will be used to
2105:AD 2C BF          131          LDA   DEVADR62        ; jump to the right
2108:8D 1D 24          132          STA   DEV2SV          ; driver after SPY
210B:AD 2D BF          133          LDA   DEVADR62+1      ; finishes
210E:8D 1E 24          134          STA   DEV2SV+1
2111:18                135          CLC
2112:A9 92             136          LDA   #>SPY           ;Calculate the start address
2114:8D 19 24          137          STA   SPYSV           ; for SPY and store it
2117:8D 1C BF          138          STA   DEVADR61        ; in the ProDOS global page
211A:8D 2C BF          139          STA   DEVADR62        ; so that calls to the
211D:A9 22             140          LDA   #<SPY           ; slot 6 device drivers will
211F:6D FF 21          141          ADC   RELO            ; come to SPY
2122:8D 1A 24          142          STA   SPYSV+1
2125:8D 1D BF          143          STA   DEVADR61+1
2128:8D 2D BF          144          STA   DEVADR62+1
212B:                  145 ;
212B:A0 00             146          LDY   #0              ;Move program to
212D:84 3C             147          STY   A1L             ; the space reserved
212F:A9 22             148          LDA   #<BEGIN         ; for it, using the
2131:85 3D             149          STA   A1L+1           ; monitor MOVE routine
2133:A9 23             150          LDA   #>LAST
2135:85 3E             151          STA   A2L
2137:A9 25             152          LDA   #<LAST
2139:85 3F             153          STA   A2L+1
213B:84 42             154          STY   A4L
213D:AD FE 21          155          LDA   ORIG
2140:85 43             156          STA   A4L+1
2142:20 2C FE          157          JSR   MOVE
2145:                  158 ;
2145:A2 8B             159          LDX   #BANNER1-BANNER0
2147:A0 00             160          LDY   #0              ;Print greeting
2149:B9 73 21          161 BNRLP    LDA   BANNER0,Y
214C:2C 98 BF          162          BIT   MACHID
214F:30 03    2154     163          BMI   BNROUT          ;On Apple II,
2151:20 0B 24          164          JSR   UPCASE          ; change lower case to upper
2154:20 ED FD          165 BNROUT   JSR   COUT
2157:C8                166          INY
2158:CA                167          DEX
2159:D0 EE    2149     168          BNE   BNRLP
215B:60                169          RTS                   ;Return to caller
215C:                  170 ;
215C:48                171 SETBIT   PHA                   ;This subroutine protects
215D:29 07             172          AND   #$07            ; the memory page whose hi
215F:AA                173          TAX                   ; address byte is in A
2160:68                174          PLA
2161:4A                175          LSR                   ;Upper five bits of A
2162:4A                176          LSR                   ; make an index into
2163:4A                177          LSR                   ; the bitmap
2164:A8                178          TAY
2165:A9 00             179          LDA   #0              ;Lower three bits of A
2167:38                180          SEC                   ; choose the bit
2168:6A                181 SETLP    ROR                   ; to be set--X counts
2169:CA                182          DEX                   ; while bit is shifted
216A:10 FC    2168     183          BPL   SETLP           ; from carry bit
216C:19 58 BF          184          ORA   BITMAP,Y
216F:99 58 BF          185          STA   BITMAP,Y        ;Set the bit in the bitmap
2172:60                186          RTS
2173:                  187 ;
2173:A0 A0 A0 A0       188 BANNER0  ASC   '                SPY' ;17 SPACES
2177:A0 A0 A0 A0
217B:A0 A0 A0 A0
217F:A0 A0 A0 A0
2183:A0 D3 D0 D9
2187:8D                189          DB    $8D
2188:A0 C1 A0 F5       190          ASC   ' A utility to monitor ProDOS operation'
218C:F4 E9 EC E9
2190:F4 F9 A0 F4
```

.isting 1 for ProDOS Spy

:PY *(continued)*

```
194:EF A0 ED EF
198:EE E9 F4 EF
19C:F2 A0 D0 F2
1A0:EF C4 CF D3
1A4:A0 EF F0 E5
1A8:F2 E1 F4 E9
1AC:EF EE
1AE:8D                  191             DB      $8D
1AF:A0 A0 E2 F9         192             ASC     '  by Ken Manly, Buffalo Chip Software'
1B3:A0 CB E5 EE
1B7:A0 CD E1 EE
1BB:EC F9 AC A0
1BF:C2 F5 E6 E6
1C3:E1 EC EF A0
1C7:C3 E8 E9 F0
1CB:A0 D3 EF E6
1CF:F4 F7 E1 F2
1D3:E5
1D4:8D                  193             DB      $8D
1D5:C3 EF F0 F9         194             ASC     'Copyright (C) 1987 by Micro-Sparc, Inc.'
1D9:F2 E9 E7 E8
1DD:F4 A0 A8 C3
1E1:A9 A0 B1 B9
1E5:B8 B7 A0 E2
1E9:F9 A0 CD E9
1ED:E3 F2 EF AD
1F1:D3 F0 E1 F2
1F5:E3 AC A0 C9
1F9:EE E3 AE
1FC:8D 8D               195             DB      $8D,$8D
1FE:            21FE    196 BANNER1     EQU     *
1FE:            0001    197 ORIG        DS      1
1FF:            0001    198 RELO        DS      1
2200:                   199 ;
2200:           0000    200 HERE        EQU     >*
2200:                   201 ;
2200:                   202 ********************************
2200:                   203 ;
2200:                   204 ;Routine to identify command in input buffer
2200:                   205 ;
2200:D8                 206 BEGIN       CLD
2201:A2 00              207             LDX     #0
2203:BD 00 02           208 NXTCHR1     LDA     INPUT,X         ;Get input command
2206:20 0B 24           209             JSR     UPCASE
2209:DD 07 25           210             CMP     CMD1+1,X        ;Compare with SPY
220C:D0 09    2217      211             BNE     TRYUNSP
220E:E8                 212             INX
220F:EC 06 25           213             CPX     CMD1
2212:D0 EF    2203      214             BNE     NXTCHR1
2214:4C 30 22           215             JMP     SETSPY
2217:A2 00              216 TRYUNSP     LDX     #0
2219:BD 00 02           217 NXTCHR2     LDA     INPUT,X         ;Get input command
221C:20 0B 24           218             JSR     UPCASE
221F:DD 0B 25           219             CMP     CMD2+1,X        ;Compare with DIR
2222:D0 0B    222C      220             BNE     TRYNEXT         ;Mismatch--not our cmd
2224:E8                 221             INX
2225:EC 0A 25           222             CPX     CMD2
2228:D0 EF    2219      223             BNE     NXTCHR2
222A:F0 27    2253      224             BEQ     SETUNSPY
222C:38                 225 TRYNEXT     SEC                     ;Jump to next added command
222D:4C 9E BE           226             JMP     XRETURN         ;Modified at installation
2230:                   227 ;
2230:20 7C 22           228 SETSPY      JSR     SETXTRN
2233:AD 19 24           229             LDA     SPYSV           ;Point the device
2236:8D 1C BF           230             STA     DEVADR61        ; driver vectors for
2239:8D 2C BF           231             STA     DEVADR62        ; slot 6 devices to
223C:AD 1A 24           232             LDA     SPYSV+1         ; the SPY routine
223F:8D 1D BF           233             STA     DEVADR61+1
2242:8D 2D BF           234             STA     DEVADR62+1
2245:AC 18 25           235             LDY     SPYON
2248:B9 18 25           236 ONLP        LDA     SPYON,Y         ;Message for
224B:20 ED FD           237             JSR     COUT            ; SPY on
224E:88                 238             DEY
224F:D0 F7    2248      239             BNE     ONLP
2251:18                 240             CLC
2252:60                 241             RTS
2253:                   242 ;
2253:20 7C 22           243 SETUNSPY    JSR     SETXTRN
2256:AD 1B 24           244             LDA     DEV1SV          ;Restore vector for
2259:8D 1C BF           245             STA     DEVADR61        ; device 1, slot 6
225C:AD 1C 24           246             LDA     DEV1SV+1
225F:8D 1D BF           247             STA     DEVADR61+1
2262:AD 1D 24           248             LDA     DEV2SV          ;Restore vector for
2265:8D 2C BF           249             STA     DEVADR62        ; device 2, slot 6
2268:AD 1E 24           250             LDA     DEV2SV+1
226B:8D 2D BF           251             STA     DEVADR62+1
226E:AC 19 25           252             LDY     SPYOFF
2271:B9 19 25           253 OFFLP       LDA     SPYOFF,Y        ;Message for
2274:20 ED FD           254             JSR     COUT            ; SPY off
2277:88                 255             DEY
2278:D0 F7    2271      256             BNE     OFFLP
227A:18                 257             CLC
227B:60                 258             RTS
227C:                   259 ;
227C:A2 00              260 SETXTRN     LDX     #0              ;Indicate
227E:8E 53 BE           261             STX     XCNUM           ; external command
2281:8E 54 BE           262             STX     PBITS           ; with no parameters
2284:8E 55 BE           263             STX     PBITS+1         ; to parse and no
2287:A9 BE              264             LDA     #<XRETURN       ; further action
2289:8D 51 BE           265             STA     XTRNADDR+1      ; needed
228C:A9 9E              266             LDA     #>XRETURN
228E:8D 50 BE           267             STA     XTRNADDR
2291:60                 268             RTS
2292                    269 ;
2292                    270 ********************************
2292:                   271 ;
2292:08                 272 SPY         CLD                     ;Save everything in sight
2293:08                 273             PHP                     ; on the stack
2294:48                 274             PHA
2295:98                 275             TYA
2296:48                 276             PHA
2297:8A                 277             TXA
2298:48                 278             PHA
2299                    279 ;
2299:A5 43              280             LDA     UNITNUM         ;Bit 7 gives drive #
229B:2A                 281             ROL                     ; 0=drive 1; 1=drive 2
229C:2A                 282             ROL                     ;Move bit 7 to bit 2
229D:2A                 283             ROL                     ; position
229E:29 02              284             AND     #$02            ;Isolate it
22A0:A8                 285             TAY                     ;Use it as index to
22A1:B9 1B 24           286             LDA     DEV1SV,Y        ; pick up the right
22A4:8D B9 24           287             STA     HOOK            ; driver address and
22A7:B9 1C 24           288             LDA     DEV1SV+1,Y      ; modify the JMP at the
22AA:8D BA 24           289             STA     HOOK+1          ; end of this routine
22AD:                   290 ;
22AD:AD 00 D0           291             LDA     BANK1D          ;Save byte to identify
22B0:8D 25 24           292             STA     BANKSV          ; active memory bank
22B3:                   293 ;
22B3:A5 36              294             LDA     CSWL            ;Save output hook which
22B5:8D 23 24           295             STA     CSWLSV          ; now points to the output
22B8:A5 37              296             LDA     CSWL+1          ; handler in ProDOS
22BA:8D 24 24           297             STA     CSWLSV+1
22BD:AD 30 BE           298             LDA     VECTOUT         ;Set the output hook to
22C0:85 36              299             STA     CSWL            ; point to the true output
22C2:AD 31 BE           300             LDA     VECTOUT+1       ; device (the printer,
22C5:85 37              301             STA     CSWL+1          ; for example)
22C7:                   302 ;
22C7:AD 82 C0           303             LDA     ROMRD           ;Turn ROM on
22CA:                   304 ;
22CA:38                 305             SEC                     ;The return address for
22CB:AD 9C BF           306             LDA     CMDADR          ; this MLI call is stored
22CE:E9 03              307             SBC     #3              ; on the system global page--
22D0:85 3C              308             STA     A1L             ; 3 bytes before return addr.
22D2:AD 9D BF           309             LDA     CMDADR+1        ; is the command for this MLI
22D5:E9 00              310             SBC     #0              ; call--subtract 3 from return
22D7:85 3D              311             STA     A1L+1           ; address and use that number
22D9:A0 00              312             LDY     #0              ; to get the MLI cmd being
22DB:B1 3C              313             LDA     (A1L),Y         ; executed.
22DD                    314 ;
22DD:C9 C0              315             CMP     #$C0            ;Handle the READ and WRITE
22DF:B0 03    22E4      316             BCS     CMPR            ; BLOCK commands separately
22E1:4C 97 23           317             JMP     BLOCKCMD
22E4:C9 26 24           318 CMPR        CMP     LASTCALL        ;Skip if we are still working
22E7:D0 03    22EC      319             BNE     NEWCMD          ; on the same MLI command
22E9:4C AF 23           320             JMP     DEVCALL
22EC:8D 26 24           321 NEWCMD      STA     LASTCALL        ;Save new cmd for future ref
22EF:                   322 ;
22EF:29 1F              323             AND     #$1F            ;Isolate lower 5 bits of cmd
22F1:0A                 324             ASL                     ;Multiply by 8
22F2:0A                 325             ASL
22F3:0A                 326             ASL
22F4:A8                 327             TAY
22F5:A2 08              328             LDX     #8              ;Eight chars in each cmd name
22F7:B9 32 24           329 CALLP       LDA     CALLTBL,Y       ;Print MLI command name
22FA:20 ED FD           330             JSR     COUT
22FD:C8                 331             INY
22FE:CA                 332             DEX
22FF:D0 F6    22F7      333             BNE     CALLP
2301:                   334 ;
2301:AD 26 24           335             LDA     LASTCALL        ;Recover command code
2304:C9 C5              336             CMP     #$C5            ;On Line command
2306:F0 78    2380      337             BEQ     EOL             ;Print no more
2308:C9 C7              338             CMP     #$C7            ;Get Prefix command
230A:F0 74    2380      339             BEQ     EOL             ;Print no more
230C:C9 C9              340             CMP     #$C9            ;Newline
230E:B0 3D    234D      341             BCS     REFNUM          ;These cmds use ref numbers
2310:                   342 ;
2310:A9 A0              343             LDA     #$A0
2312:20 ED FD           344             JSR     COUT
2315:                   345 ;
2315:38                 346             SEC                     ;The return address for
2316:AD 9C BF           347             LDA     CMDADR          ; this MLI call is stored
2319:E9 02              348             SBC     #2              ; on the system global page--
231B:85 3C              349             STA     A1L             ; 2 bytes before return addr
231D:AD 9D BF           350             LDA     CMDADR+1        ; is the parm block pointer
2320:E9 00              351             SBC     #0              ; --subtract 2 from return
2322:85 3D              352             STA     A1L+1           ; address and use that number
2324:A0 00              353             LDY     #0              ; to get the parm block
2326:B1 3C              354             LDA     (A1L),Y         ; address
2328:AA                 355             TAX                     ;Low byte
2329:C8                 356             INY
232A:B1 3C              357             LDA     (A1L),Y         ;Hi byte
232C:86 3C              358             STX     A1L             ;Use the new pointer
232E:85 3D              359             STA     A1L+1           ;Pathname pointer is
2330:B1 3C              360             LDA     (A1L),Y         ; at byte 1 in parm block
2332:AA                 361             TAX                     ;Low byte
2333:C8                 362             INY
2334:B1 3C              363             LDA     (A1L),Y         ;Hi byte
2336:86 3C              364             STX     A1L             ;Use the new pointer
2338:85 3D              365             STA     A1L+1           ;Ready to print pathname
233A                    366 ;
233A:A0 00              367             LDY     #0
233C:B1 3C              368             LDA     (A1L),Y         ;Length of filename
233E:AA                 369             TAX                     ; Count chars in filename
233F:C8                 370             INY
2340:B1 3C              371 PATHLP      LDA     (A1L),Y         ;Get a char
2342:09 80              372             ORA     #$80
2344:20 ED FD           373             JSR     COUT            ;Print a char
2347:C8                 374             INY
2348:CA                 375             DEX
2349:D0 F5    2340      376             BNE     PATHLP
234B:F0 33    2380      377             BEQ     EOL             ;Print no more
234D:                   378 ;
234D:20 48 F9           379 REFNUM      JSR     PRBLNK          ;Print a few spaces
2350:A2 05              380             LDX     #5
2352:A0 00              381             LDY     #0
2354:B9 01 25           382 FLLP        LDA     FILE,Y          ; and print the word
2357:20 ED FD           383             JSR     COUT            ; 'FILE'
235A:C8                 384             INY
235B:CA                 385             DEX
235C:D0 F6    2354      386             BNE     FLLP
235E:                   387 ;
235E:38                 388             SEC                     ;The return address for
235F:AD 9C BF           389             LDA     CMDADR          ; this MLI call is stored
2362:E9 02              390             SBC     #2              ; on the system global page--
2364:85 3C              391             STA     A1L             ; 2 bytes before return addr
2366:AD 9D BF           392             LDA     CMDADR+1        ; is the parm block pointer
2369:E9 00              393             SBC     #0              ; --subtract 2 from return
```

```
2368:85 3D          394           STA   A1L+1          ; address and use that number
236D:A0 00          395           LDY   #0             ; to get the parm block
236F:B1 3C          396           LDA   (A1L),Y        ; address.
2371:AA             397           TAX                  ;Low byte
2372:C8             398           INY
2373:B1 3C          399           LDA   (A1L),Y        ;Hi byte
2375:86 3C          400           STX   A1L            ;Use the new pointer
2377:85 3D          401           STA   A1L+1          ;File ref number is
2379:B1 3C          402           LDA   (A1L),Y        ; at byte 1 in parm block
237B:               403 ;
237B:09 B0          404           ORA   #$B0           ;Make a numeral
237D:20 ED FD       405           JSR   COUT           ; and print it
2380:20 8E FD       406 EOL       JSR   CROUT
2383:A2 17          407           LDX   #23
2385:A0 00          408           LDY   #0
2387:B9 EA 24       409 HDLP      LDA   HEADER,Y       ;Print the
238A:20 ED FD       410           JSR   COUT           ; header for the
238D:C8             411           INY                  ; device driver calls
238E:CA             412           DEX
238F:D0 F6   2387   413           BNE   HDLP
2391:20 8E FD       414           JSR   CROUT
2394:4C AF 23       415           JMP   DEVCALL
2397:               416 ;
2397:8D 26 24       417 BLOCKCMD  STA   LASTCALL
239A:29 03          418           AND   #$03
239C:0A             419           ASL
239D:0A             420           ASL
239E:0A             421           ASL
239F:A8             422           TAY
23A0:A2 08          423           LDX   #8
23A2:B9 D2 24       424 CALLP2    LDA   BCALLTBL,Y
23A5:20 ED FD       425           JSR   COUT
23A8:C8             426           INY
23A9:CA             427           DEX
23AA:D0 F6   23A2   428           BNE   CALLP2
23AC:20 8E FD       429           JSR   CROUT
23AF:               430 ;
23AF:20 48 F9       431 DEVCALL   JSR   PRBLNK         ;Print a few spaces
23B2:A6 42          432           LDX   COMMAND        ;Get device driver cmd and
23B4:BD 1F 24       433           LDA   ABBREV,X       ; use it as index to get
23B7:20 ED FD       434           JSR   COUT           ; and print a letter code
23BA:               435 ;
23BA:20 48 F9       436           JSR   PRBLNK         ;Print a few spaces
23BD:A5 43          437           LDA   UNITNUM        ;Hi bit gives drive number
23BF:0A             438           ASL                  ;Shift it into carry
23C0:A9 00          439           LDA   #0
23C2:69 01          440           ADC   #1             ;Drive = carry bit + 0 + 1
23C4:20 DA FD       441           JSR   PRBYTE         ;Print drive number
23C7:20 48 F9       442           JSR   PRBLNK         ; and a few spaces
23CA:A5 47          443           LDA   BLOCKNUM+1     ;Get block number requested
23CC:20 DA FD       444           JSR   PRBYTE         ; and print it
23CF:A5 46          445           LDA   BLOCKNUM
23D1:20 DA FD       446           JSR   PRBYTE
23D4:20 48 F9       447           JSR   PRBLNK         ;Still more spaces
23D7:A5 45          448           LDA   BUFFPTR+1      ;Get buffer address requested
23D9:20 DA FD       449           JSR   PRBYTE         ; and print it
23DC:A5 44          450           LDA   BUFFPTR
23DE:20 DA FD       451           JSR   PRBYTE
23E1:20 8E FD       452           JSR   CROUT          ;Print no more
23E4:               453 ;
23E4:AD 23 24       454           LDA   CSHLSV         ;Restore the output
23E7:85 36          455           STA   CSHL           ; hook which was
23E9:AD 24 24       456           LDA   CSHLSV+1       ; probably pointing
23EC:85 37          457           STA   CSHL+1         ; to ProDOS
23EE:               458 ;
23EE:AD B8 C0       459           LDA   RAMRD1         ;Turn on one RAM bank
23F1:AD B8 C0       460           LDA   RAMRD1
23F4:AD 00 D0       461           LDA   BANKID         ;Is it the right one?
23F7:CD 25 24       462           CMP   BANKSV
23FA:F0 06   2402   463           BEQ   RESTORE        ;Yes, continue restoration
23FC:AD 83 C0       464           LDA   RAMRD2         ;Turn on other RAM bank
23FF:AD 83 C0       465           LDA   RAMRD2
2402:               466 ;
2402:68             467 RESTORE   PLA                  ;Restore everything from
2403:AA             468           TAX                  ; the stack
2404:68             469           PLA
2405:A8             470           TAY
2406:68             471           PLA
2407:28             472           PLP
2408:               473 ;
2408:       2489    474 HOOK      EQU   *+1            ;Jump to the device driver
2408:4C 00 00       475           JMP   0              ; address specified earlier
240B:               476 ;
240B:               477 ;Subroutine to convert lower case
240B:               478 ;
240B:C9 E1          479 UPCASE    CMP   #$E1
240D:90 06   2415   480           BCC   UCRTN
240F:C9 FB          481           CMP   #$FB
2411:B0 02   2415   482           BCS   UCRTN
2413:29 DF          483           AND   #$DF
2415:60             484 UCRTN     RTS
2416:               485 ;
2416:00             486           DB    0              ;Stop reloc. instr operands
2417:00 00          487 ATBL      DW    0              ;No internal address table
2419:               488 ;
2419:       0002    489 SPYSV     DS    2              ;Entry for SPY
241B:       0002    490 DEV1SV    DS    2              ;Driver address for drive 1
241D:       0002    491 DEV2SV    DS    2              ;Driver address for drive 2
241F:               492 ;
241F:D3 D2 D7 C6    493 ABBREV    ASC   'SRWF'         ;Codes for device driver cmds
2423:       0002    494 CSHLSV    DS    2              ;Stash for output hook
2425:       0001    495 BANKSV    DS    1              ;Stash for mem bank ID byte
2426:       0001    496 LASTCALL  DS    1              ;Last MLI command used
2427:               497 ;
2427:               498 ;This table gives the location of the file reference
2427:               499 ; numbers (relative to $BEC7) for MLI commands
2427:               500 ; $C9 (Newline) to $D3 (Getbut)
2427:               501 ;
2427:0B 0F 0F 17    502 REFTBL    DFB   11,15,15,23,23,0,0,0,0,0,0
2431:17 00 00 00
243F:00 00 00
2432:               503 :
```

```
2432:               504 ;This table gives the names of the MLI commands
2432:               505 ; from $C0 (Create) to $D3 (Getbut)
2432:               506 ;
2432:C3 D2 C5 C1    507 CALLTBL   ASC   'CREATE  '
2436:D4 C5 A0 A0
243A:C4 C5 D3 D4    508           ASC   'DESTROY '
243E:D2 CF D9 A0
2442:D2 C5 CE C1    509           ASC   'RENAME  '
2446:CD C5 A0 A0
244A:D3 C5 D4 A0    510           ASC   'SET INFO'
244E:C9 CE C6 CF
2452:C7 C5 D4 A0    511           ASC   'GET INFO'
2456:C9 CE C6 CF
245A:CF CE A0 CC    512           ASC   'ON LINE '
245E:C9 CE C5 A0
2462:D3 C5 D4 A0    513           ASC   'SET PRFX'
2466:D0 D2 C6 D8
246A:C7 C5 D4 A0    514           ASC   'GET PRFX'
246E:D0 D2 C6 D8
2472:CF D0 C5 CE    515           ASC   'OPEN    '
2476:A0 A0 A0 A0
247A:CE C5 D7 CC    516           ASC   'NEWLINE '
247E:C9 CE C5 A0
2482:D2 C5 C1 C4    517           ASC   'READ    '
2486:A0 A0 A0 A0
248A:D7 D2 C9 D4    518           ASC   'WRITE   '
248E:C5 A0 A0 A0
2492:C3 CC CF D3    519           ASC   'CLOSE   '
2496:C5 A0 A0 A0
249A:C6 CC D5 D3    520           ASC   'FLUSH   '
249E:C8 A0 A0 A0
24A2:D3 C5 D4 A0    521           ASC   'SET MARK'
24A6:CD C1 D2 CB
24AA:C7 C5 D4 A0    522           ASC   'GET MARK'
24AE:CD C1 D2 CB
24B2:D3 C5 D4 A0    523           ASC   'SET EOF '
24B6:C5 CF C6 A0
24BA:C7 C5 D4 A0    524           ASC   'GET EOF '
24BE:C5 CF C6 A0
24C2:D3 C5 D4 A0    525           ASC   'SET BUF '
24C6:C2 D5 C6 A0
24CA:C7 C5 D4 A0    526           ASC   'GET BUF '
24CE:C2 D5 C6 A0
24D2:               527 ;
24D2:               528 ;This table gives the names of the MLI commands
24D2:               529 ; from $80 (Readblock) to $83 (Gettime)
24D2:               530 ;
24D2:D2 C5 C1 C4    531 BCALLTBL  ASC   'READ BLK'
24D6:A0 C2 CC CB
24DA:D7 D2 D4 C5    532           ASC   'WRTE BLK'
24DE:A0 C2 CC CB
24E2:C7 C5 D4 A0    533           ASC   'GET TIME'
24E6:D4 C9 CD C5
24EA:               534 ;
24EA:               535 ;Header for the device driver calls
24EA:               536 ;
24EA:A0 A0 C3 CD    537 HEADER    ASC   '  CMD UNIT  BLOCK  ADDR'
24EE:C4 A0 D5 CE
24F2:C9 D4 A0 A0
24F6:C2 CC CF C3
24FA:CB A0 A0 C1
24FE:C4 C4 D2
2501:               538 ;
2501:               539 ;Some useful words
2501:               540 ;
2501:C6 C9 CC C5    541 FILE      ASC   'FILE '
2505:A0
2506:83 D3 D0 D9    542 CMD1      STR   'SPY'
250A:95 D5 CE D3    543 CMD2      STR   'UNSPY'
250E:D0 D9
2510:88 8D          544 SPYON     DFB   8,$8D
2512:CE CF A0 D9    545           ASC   'NO YPS'
2516:D0 D3
2518:8D             546           DFB   $8D
2519:89 8D          547 SPYOFF    DFB   9,$8D
251B:C6 C6 CF A0    548           ASC   'FFO YPS'
251F:D9 D0 D3
2522:8D             549           DFB   $8D
2523:       2523    550 LAST      EQU   *
```

END OF LISTING 1

KEY PERFECT 5.0
RUN ON
SPY

CODE-5.0	ADDR# - ADDR#	CODE-4.0
4AD1ED10	207B - 20CA	262C
4A092898	20CB - 211A	28C5
5E0B10F	211B - 216A	2838
F481ACDC	216B - 21BA	2831
E5CAFB02	21BB - 220A	2AF1
39CE4E30	220B - 225A	25EC
45DBCDC1	225B - 22AA	27B8
1EDDEE0A	22AB - 22FA	2A76
37B54E3C	22FB - 234A	297B
9A0E7C3E	234B - 239A	27A5
4CD5AF2E	239B - 23EA	24DF
E7FE4FB1	23EB - 243A	238E
43F4F1E1	243B - 248A	2B33
D8049B9A	248B - 24DA	29B7
B8545EFF	24DB - 2522	2646
8E09E645	= PROGRAM TOTAL =	04A8