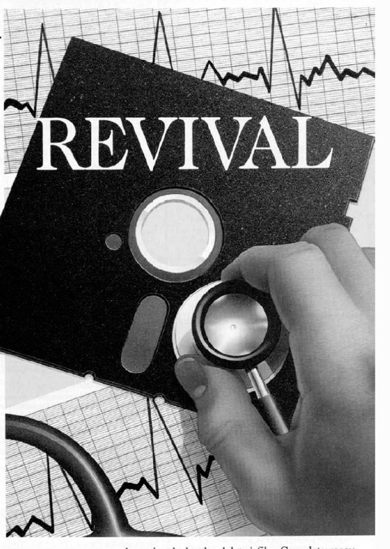
FILE REVIVAL



In the last installment of Disassembly Lines[1], we reviewed the mechanism of file deletion by the ProDOS 8 machine language interface (MLI). Up to and including ProDOS version 1.2, deleted files could not be recovered without a troublesome patch to the ProDOS system file[2]. With the inception of ProDOS version 1.3, the integrity of file index blocks is preserved, making possible both the recovery of deleted files and the repair of damaged disks. In this respect, ProDOS has finally caught up to DOS 3.3.

With single-minded purpose, this article eschews disassembly and plunges into the creation of a sophisticated, menu-driven utility that enables you to recover deleted ProDOS files. PFR is the name, and undeletion is the game. ProDOS File Recovery (PFR) functions with all versions of ProDOS and all disk devices; however, *files to be recovered must have been deleted under ProDOS version 1.3 or later.* For a detailed explanation of this phenomenon, please review the prior column.

USING THE PROGRAM

A deleted file must be recovered immediately after deletion. Save and write operations add data to the disk and may appropriate index and data blocks owned previously by the deleted file. Complete recovery of the deleted file can be assured, therefore, only if no information has been placed on the disk between deletion and restoration.

Volume Menu

To run PFR, type BRUN PFR. You are presented with a menu of active volume names. Up to 16 disk devices are supported by current versions of ProDOS 8. The bottom section of the screen contains program commands. Initially, the first menu item is highlighted (selected). The arrow keys move you around the menu, Escape restarts the program, and the Q key allows you to quit. Pressing Return brings the contents of the selected volume directory into view. If the root directory has neither subdirectory (DIR) nor deleted (DEL) files, DIRECTORY EMPTY appears in the message box and pressing a key

returns you to the volume menu. If a selected directory can't fit into memory, DIRECTORY TOO LARGE is printed.

Directory Menu

Volume directories can house no more than 51 files, whereas 65,535 files may exist within a subdirectory. More information is displayed in this menu: The full directory pathname and level number are displayed at the top of the screen; only DIR and DEL files are contained in the menu, and the help line of the directory menu is expanded. As many as 4 columns of 16 entries can appear simultaneously. The Up- and Down-Arrow keys function as expected. The Right- and Left-Arrow keys operate only if a filename is visible to the side of the selected filename. If the directory houses more than 64 entries, some filenames are hidden. MouseText's Up-Arrow characters above the first visible entry signify that additional filenames may be brought into view by pressing the Up-Arrow key when the top file is selected. MouseText's Down-Arrows are displayed if more entries exist beyond the final visible filename, in which case pressing the Down-Arrow key causes hidden names to materialize. Escape and Q provide the same functions as in the volume menu.

The Return key now has three possible functions. If a DIR file is selected, pressing Return takes you to that directory, provided that the directory isn't "empty" or doesn't contain more than 256 DIR and DEL files. In the latter instance, or if directory size exceeds available memory, the message DIRECTORY TOO LARGE is printed. If the volume bitmap (VBM) can't fit into RAM, BITMAP TOO LARGE appears.

Holding down the Open-Apple key while hitting Return returns you to the prior directory level, no matter which file is highlighted. Thus, pressing Return alone or in combination with the Open-Apple key enables you to move through the contents of the volume with ease.

This month's Disassembly Lines is also our cover feature — it contains a full-featured utility for recovering deleted files.

Undeleting a File

If the Return key is depressed when a DEL file is selected, the usual message box is replaced with the name of the deleted file, the file's storage type (e.g., seedling, sapling, tree, directory), the status of the blocks owned by that file (e.g. free, used), and the all-important query, "Undelete the file (Y/N)?" A default response is provided. If all file blocks are free, the default is Y and pressing either Return or the Y key reincarnates the file. If any file block is reserved by another file, BLOCK(S) IN USE is printed and the default is N, in which case striking RETURN or N cancels the undeletion. Of course, the suggested action may be overridden by an alternate response. If the message BAD BLOCK NUMBER appears during file recovery, the undeletion process is canceled automatically because an index block has been irrevocably destroyed or was created under a pre-1.3 version of ProDOS.

If you are desperate and decide to exhume a file containing blocks reserved by another file, please back up the entire disk before beginning the undeletion. If the main bulk of the file is recovered successfully, copying it to the backup disk ensures that no file on the backup disk will be corrupted. The backup disk must now become your working disk, and your old working disk should be reformatted.

The name of a deleted file may not meet your expectation if the file had been renamed before it was deleted. All versions of ProDOS handle the RENAME command by overwriting the filename field (FNF) with the new name and changing the length nibble. If a longer filename had been substituted for a shorter name, no incongruity is encountered. In the reverse case, a portion of the former longer name remains in the FNF. Since the length nibble had been zeroed when the file was deleted, PFR prints all ASCII characters in the FNF and may generate strange names under the circumstances just outlined.

For example, suppose you save a BASIC program with the name RANDOM. The filename field contains that name, and the length nibble is 6. You then rename that file RATF. The length nibble is now 4; only the first four characters of the filename field are changed. If you delete RATF and try to recover it, PFR will list it as RATFOM, because the rename process had converted a longer filename to a shorter one, leaving all the trailing characters in the filename field intact. The trick to pinpointing a deleted file, therefore, is to search the menu strings from left to right for your target filename.

With the inception of ProDOS version 1.3, the integrity of file index blocks is preserved.

PFR is capable of revitalizing deleted DIR files. Since DIR files can be deleted only if they are empty, why in the world would PFR waste its time on such nonsense? Have you ever used utilities such as ProSEL or Copy II Plus to delete entire directories? What if one of the deleted files in the deleted directory was vital? You could not exhume the file unless the directory was first restored. Whatever you may think of PFR, it is no dummy.

After successfully undeleting a file, you are returned to the directory menu which contained the target file. If a nondirectory file has been exhumed, it no longer appears in the menu. A restored directory file still appears, but its file type has been changed from DEL to DIR. If your target nondirectory file is the lone file in the current menu, and that file is exhumed, you are returned to the prior directory level instead of the empty menu.

To practice using PFR, run PRACTICE.PFR, a short Applesoft program that creates and deletes various types of files on the volume whose name is inserted into line 20. At least 117 blocks of disk space are required, and a fresh ProDOS RAM disk is an ideal medium. It's better to become friendly with PFR before a crisis strikes.

ENTERING THE PROGRAM

If you have an assembler, enter the source code from **Listing 1** and assemble it; save the object file as PFR. If you don't have an assembler, enter the Monitor with CALL −151 and key in the hex code from **Listing 2**. Save the program with the command:

```
BSAVE PFR,A$900,L$AD3
```

Type in the Applesoft program shown in **Listing 3** and save it with the command:

```
SAVE PRACTICE.PFR
```

If PFR is to be run on an unenhanced IIe (PFR will not run on a II Plus), changes must be made to prevent the program from trying to put MouseText characters on the screen. To make the necessary changes, load PFR into memory with the command:

```
BLOAD PFR
```

Enter the monitor (CALL −151) and type:

```
0A25:4C 41 0A
```

followed by:

```
12E7:A0 A0 DE A0 DE A0 DE A0 DE A0 DE
12F2:A0 A0 00
12F5:A0 A0 F6 A0 F6 A0 F6 A0 F6 A0 F6
1300:A0 A0 00
```

Resave it to disk with the command:

```
BSAVE PFR,A$900,L$AD3
```

For help with entering Nibble listings, see the Typing Tips section in this magazine.

HOW THE PROGRAM WORKS

Since the program is lengthy, I'll concentrate on its organization and major features. The detailed comments in Listing 1 should provide you with explanations missed by this description.

Organization

Control of PFR is centered in the main program loop which is divided into five parts:

1. Initialization (**lines 63-74**)
2. Volume menu (**lines 78-89**)
3. Directory menu (**lines 93-130, 176-183**)
4. Quit code (**lines 134-136**)
5. Error handler (**lines 140-172**).

Main loop subroutines occupy **lines 187-802**. The section of PFR devoted to file undeletion is found in **lines 806-880**, and the undeletion subroutines encompass **lines 886-1322**. Parameter lists (**lines 1326-1351**), storage locations (**lines 1355-1388**), tables (**lines 1392-1393**) and text (**lines 1399-1431**) are placed toward the end of **Listing 1**. Pointers (**lines 39-42**) are in page zero, and major buffers are external (**lines 43-49**).

External Buffers

A standard ProDOS buffer, TXBUF2 (secondary text buffer at $280-$2FF), holds the filename of the current directory. Its leading byte contains the length of that filename.

RBUF (route buffer at $1500-$15FF) houses the chain of key directory block numbers accessed by the user. The low-order block numbers are stored in the first 128 bytes of RBUF, and the high-order numbers are held in the second half of the page. ROUTEFLG (described below) indexes the active directory number in RBUF.

PBUF (pointer buffer at $1600-$18FF) occupies three pages; the first two pages hold the low- and high-order bytes, respectively, of the pointers to file entries within the current directory. File type codes are stored in the last page of PBUF.

KBUF (key buffer at $1900-$1AFF) contains the contents of the key block of the file selected for undeletion.

SBUF (subindex block buffer at $1B00-$1CFF) is a repository for the contents of the subindex block(s) of the file being exhumed.

OBUF (OPEN buffer at $1D00-$20FF) services the MLI OPEN call. It houses the open file's active index block in its upper two pages and a data block in the lower two pages.

Because DBUF (directory buffer at $2100-HIMEM) receives the entire current directory, its length is variable. This buffer also is used to store the volumes generated by the ON__LINE call.

VBUF (VBM buffer) begins on the first page boundary following the end of DBUF, and its size also varies.

Thus, pressing Return alone or in combination with the Open-Apple key enables you to move through the contents of the volume with ease.

Pointers

PPTR points to text printed via the PRTMSG subroutine **(lines 189-199)**. DPTR zeroes in on file entries within DBUF and VBMBUF. SPTR points at the selected entry within DBUF. IXPTR points to block numbers within index blocks.

Flags

ROUTEFLG **(line 1355)** points one location beyond the key (first) block number of the active directory in RBUF. When ROUTEFLG equals zero, the volume menu is active. As each new directory level is reached, ROUTEFLG is incremented by one and indexes the location within RBUF where the new key directory block number is stored.

On returning from a command in the directory menu, SELFLG **(line 1356)** controls which menu line is selected. A positive value highlights the first menu entry. If SELFLG is negative, the current selection is maintained.

BAKUPFLG **(line 1357)** regulates which directory is exhibited. If the flag is set (negative), the prior directory level is accessed. On BAKUPFLG clear (positive), the current directory is sustained.

During the display of a directory, DIR files are printed before DEL files. SCANFLG **(line 1358)** orchestrates this partition as directories are scanned. On SCANFLG positive, DIR files are dug out. When the flag is negative, DEL files are procured.

WRIXFLG **(line 1359)** governs the writing of index blocks to disk. Writing is suppressed by a clear (positive) flag and enabled by a set (negative) flag.

Initialization

When this utility is started or restarted, 80-column mode is enabled, all files are closed, and several flags are cleared. The key block of a volume, block number 2, is always stuffed into the first location of RBUF.

Volume Menu

After printing the message box containing instructions, the ON__LINE call polls all disk devices and deposits the data in DBUF; up to 16 devices may be connected. For an explanation of the MLI ON__LINE command, glance at one of your three ever-faithful ProDOS companions[3-5].

The menu of on-line volumes is revealed by PRTVOL **(lines 320-370)**, which extracts the data from DBUF. User commands are processed by GETMKEY **(lines 691-729)**, which will be dissected later.

Directory Menu

On entering the directory menu from the volume menu, SETTXB **(lines 401-414)** places the name of the root directory into TXBUF2, the pathname buffer for the MLI OPEN call. As succeeding directories are reached, SETTXB concatenates their names with the pathname already in TXBUF2. In this manner, a full directory pathname is maintained in the buffer.

Deriving the key block number from the file entry of the directory to be secured, GETKYBLK **(lines 267-275)** places the block number into RBUF.

GETDIR **(lines 418-470)** opens the directory file, calculates its length, and determines if the file would fit between the start of DBUF and HIMEM. If the file size is satisfactory, the new directory is read into DBUF and all files are closed. If the directory header establishes that no more than 255 entries exist, the type of directory is checked. The subroutine ends if a subdirectory is being examined. On finding a volume directory, header information is stored in program memory, and the size of the VBM is calculated before GETDIR returns to its caller.

SCANDIR **(lines 487-587)** scans DBUF for DIR and DEL files, using SCANFLG to segment the two file types. As each pertinent entry is uncovered, a 2-byte pointer to its location is stored in the first and second pages of PBUF, and its file type code (e.g., DIR is $0F, DEL is $00) is stuffed into the final page of PBUF. When all files have been handled, the total number of menu entries is obtained and ROUTEFLG is incremented as long as BAKUPFLG is clear. If the directory contains neither DIR nor DEL files, DIRECTORY EMPTY appears in the message box. If none of the prior subroutines has generated an error, command instructions are printed, and SELFLG is tested to determine which menu line should be selected.

If all file blocks are free, the default is "Y," and pressing either Return or the Y key reincarnates the file.

PRTDIR **(lines 591-618)** employs several subroutines to display the contents of the current directory. PRTTXB **(lines 386-397)** outputs the full name of the directory on the top screen line. CKTOPARW **(lines 622-628)** ascertains whether the first file to be printed is the initial entry in the directory. If not, MouseText Up-Arrows are placed on the second line; if so, the second line is cleared. PRTENTRY **(lines 643-668)** withdraws the data from PBUF and prints the file type and filename. A simple loop allows as many as 64 entries to be shown. PRTDIR exits via CKBOTARW **(lines 632-639)**, which displays MouseText Down-Arrows if more entries exist.

GETMKEY **(lines 691-729)** fetches a menu keypress and routes certain commands to specific handler routines. On returning to its caller, the action taken by GETMKEY is indicated by the state of five processor status flags and the A-Register. A summary of these settings is presented in **lines 691-698**.

Escape and Q are processed forthwith. Whereas Return has a singular purpose in the volume menu, this key serves three functions in the directory menu: procuring a new directory, backing up to the prior directory, and recovering a deleted file. RTN (**lines 785-794**) interprets a carriage return and an Open-Apple keypress and ensures that the above-noted flags are conditioned properly. If file recovery is signaled, flow passes to the common undeletion code (see the Undeletion section). When GETMKEY returns to the main program loop, critical flags and registers are tested and appropriate action is taken.

Quitting

If the Q command has been issued, the directory menu allows for a change of heart via CKEXIT (**lines 125-130**). If exit is confirmed, the quit code returns control to BASIC.

Error Handler

Most errors demand the stripping of one file level from TXBUF2 by STRIPTXB (**lines 374-382**). If an error occurred during undeletion, the strip is unnecessary. MLI error codes are positive numbers, i.e., $00-$7F, which are decoded by BADCALL. The corresponding BASIC interpreter (BI) error message is output by PRINTERR. Both are BI global page subroutines. Negative values, e.g., $F0-$F3, are assigned to internal errors, which are handled directly by HANDLERR, the program error handler. After pausing to view the error message, ROUTEFLG indicates which menu should be displayed. If the directory menu is called for, ROUTEFLG is decremented and the selected line is preserved by setting SELFLG. Should the volume menu be required, PFR is restarted.

Undeletion

When file recovery is invoked, the common undeletion code calls SETDSPLY (**lines 886-971**) to fill the top line of the message box with the name of the file to be restored, the storage type of that file, and the fields for counting free and used file blocks. GETVBMBF (**lines 1004-1017**) calculates the beginning page of the VBM buffer and checks whether this buffer fits the memory constraints imposed by BASIC. RDVBM (**lines 1027-1047**) reads one or more VBM blocks into memory. After ZBLKS (**lines 1051-1056**) zeroes the count of free and used blocks and PRTBLKS (**lines 1071-1082**) prints the result in the message box, GETUKYBL (**lines 1086-1094**) obtains the key block number of the target file. FIXVBM (**lines 1098-1139**) finds the position of a given block number in the VBM, determines whether that block is free or used, prints its finding in the message box, and ensures that the target block is reserved. If the block number being processed exceeds maximum size on that volume, the fatal BAD BLOCK NUMBER message is generated.

If a seedling file is being exhumed, control passes directly to DOUNDEL (**lines 845-868**), the heart of the undeletion code. Before any other storage type undergoes specific treatment, SETKYBLK (**lines 1267-1277**) fixes parameters for the key file block.

A directory file is handled by DIRECTRY (**lines 839-841**), which calls DODIR (**lines 1281-1322**) to prepare for file reclamation by counting the free and used blocks, reserving file blocks in the VBM, and reconstructing the zeroed storage__type and name__length byte in the header.

In deleting a file, the MLI reverses the usual low order/high order sequence in all index blocks (see "D/L," Vol. 8/No. 11). DOIX (**lines 1182-1210**) readies the key index block for restoration by reading the block into KBUF and reversing the high- and low-order bytes of the block numbers so that a normal relationship is reestablished. In addition, DOIX calls FIXVBM to reserve within the VBM each block encountered. At this stage, the single index block of a sapling file or the master index block of a tree file has been repaired.

If a sapling file is under scrutiny, all is ready for DOUNDEL. Tree files are further treated by DOSUBIX (**lines 1214-1230**), which employs DOIX to restore order to each and every subindex block.

Regardless of storage type, control eventually passes to DOUNDEL where CKUNDEL (**lines 1143-1178**) allows the user to view the block usage fields and decide whether to proceed with the recovery. If the answer is no, undeletion is canceled and the directory menu is reprinted. An affirmative reply causes DOUNDEL to replace the zeroed first byte of the file entry with an accurate storage__type and name__length. WRKEYSIX (**lines 1234-1248**) writes the key block to disk, thus etching into magnetic or electronic memory the master index block of a tree file, the index block of a sapling file, or the header block of a directory file. After setting WRIXFLG, the latter subroutine exits through DOSUBIX, which writes subindex blocks of a tree file to disk. DOUNDEL completes its task by calling WRITVBM (**lines 1021-1047**) and WRITDIR (**lines 975-1000**) to write the VBM and directory, respectively, back to disk.

The undeletion process continues with **lines 863-864**, which issue a backup signal if file restoration has caused an empty directory, and **lines 865-868**, which adjust the menu selection if the last file in the directory has been deleted. Before returning to the main loop, **lines 872-880** condition the processor status register and accumulator so that the main loop will understand what has transpired during file restoration. *Note:* the decimal flag is turned on to signal an undeletion error. Because decimal mode can wreak havoc with a program, the flag is cleared immediately on returning to the main loop.

Debugging and Testing

Debugging a program as complex as PFR takes longer than writing the utility. During this testing process, I have annihilated several nests of termites. For the past two weeks, no glitches have occurred using hard disk and floppy media. Does this mean that PFR is perfect? Not on your life! Continue to back up all vital work so that unexpected destruction of a disk will not cause you undue hardship. As usual, if any aberrations surface, please drop me a line. I would like PFR to be the definitive program for file recovery, and I need your help.

REFERENCES

1. Mossberg, S., "Disassembly Lines: FLASH: ProDOS 8 Supports File Recovery," *Nibble*, November 1987, pp. 100-108.
2. Mossberg, S., "Disassembly Lines: The CAT and CATALOG Commands," *Nibble*, May 1986, pp. 114-128.
3. *ProDOS TECHNICAL REFERENCE MANUAL*. Reading, MA: Addison-Wesley Publishing Company, Inc.
4. Little, Gary. *APPLE ProDOS: ADVANCED FEATURES FOR PROGRAMMERS*. Bowie, MD: Brady Communications Company, Inc. (a Prentice-Hall Publishing Company), 1985.
5. Worth, Don and Pieter Lechner. *BENEATH APPLE ProDOS*. Chatsworth, CA: Quality Software, 1984.

LISTING 1: PFR.S

```
         LST    ON
*********************************
*                               *
*          P F R . S            *
*     ProDOS File Recovery      *
*      by  Sandy Mossberg       *
*                               * Merlin-Pro
*      Copyright (c) 1988       *
*      by MicroSPARC, Inc.      *
*      Concord, MA  01742       *
*                               *
*********************************
CV       =      $25        ;cursor row
INVFLG   =      $32        ;inverse flag
MEMSIZ   =      $73        ;HIMEM
DOSWRM   =      $3D0       ;warmstart ProDOS
OURCH    =      $57B       ;cursor column (80-column)
PRINTERR =      $BE0C      ;print error message
BADCALL  =      $BE8B      ;convert MLI error to BI error
MLI      =      $BF00      ;MLI call
KEY      =      $C000      ;keypress storage
STROBE   =      $C010      ;keyboard strobe
BUTN0    =      $C061      ;Open-Apple status
C3ROM    =      $C300      ;initialize 80-column mode
DEVNUM   =      $BF30      ;S/D of active device
PRBLNK   =      $F948      ;output 3 spaces
PRBL2    =      $F94A      ;output X-Register spaces
HOME     =      $FC58      ;home cursor, clear screen
CLREOP   =      $FC42      ;clear to end of page
CLREOL   =      $FC9C      ;clear to end of line
RDKEY    =      $FD0C      ;get keypress
CROUT    =      $FD8E      ;output CR
PRBYTE   =      $FDDA      ;print hex byte
COUT     =      $FDED      ;output char
SETINV   =      $FE80      ;set inverse text mode
SETNORM  =      $FE84      ;set normal text mode
BELL     =      $FF3A      ;output bell char
*-------------------------------
IXPTR    =      $EE        ;ptr to index block
DPTR     =      $FA        ;ptr to data and VBM buffers
SPTR     =      $FC        ;ptr to selected file
PPTR     =      $FE        ;ptr to text
TXBUF2   =      $280       ;pathname buffer
RBUF     =      $1500      ;buffer for key block numbers
PBUF     =      $1600      ;buffer for file pointers
KBUF     =      $1900      ;buffer for key block
SBUF     =      $1B00      ;buffer for subindex block
OBUF     =      $1D00      ;buffer for OPEN call
DBUF     =      $2100      ;buffer for READ & ON_LINE calls
DIRFILES =      DBUF+4     ;start of dir entries
ENTLEN   =      DBUF+$23   ;entry.length
ENTPBLK  =      DBUF+$24   ;entries.per.block
FILCNT   =      DBUF+$25   ;file.count (active files)
HVBMPTR  =      DBUF+$27   ;bit.map.pointer
HTOTBLKS =      DBUF+$29   ;total.blocks

*******************************

         ORG    $900
*===============================
* MAIN PROGRAM LOOP:
*===============================
RESTART  LDA    #$81       ;unprintable char
         JSR    C3ROM      ;turn on 80-columns
         JSR    CLOSALL    ;close all files
         LDA    #0
         STA    ROUTEFLG   ;flag volume menu
         STA    SELLIN     ;select 1st line in volume menu
         STA    TOPSCR     ;put 1st line at top of screen
         STA    SELFLG     ;clear select flag
         STA    TXBUF2     ;zero length byte in TXBUF2
         STA    RBUF+128   ;key block of
         LDA    #2         ; volume dir
         STA    RBUF       ; always $0002
*-------------------------------
* Volume Menu:
*-------------------------------
         JSR    PRTHELP1   ;print instructions
         JSR    MLI        ;execute ON_LINE call
         HEX    C5
         DA     OLPARM
         BCS    HANDLERR   ;MLI error
:1       JSR    PRTVOL     ;print on-line volumes
         BCS    HANDLERR   ;no volume on-line
         JSR    GETNKEY    ;get command keypress
         BCC    :1         ;Arrow key struck
         BNE    RESTART    ;Escape restarts volume search
         CMP    #"Q'
         BEQ    CKEXIT     ;QUIT if you agree
*-------------------------------
* Directory Menu:
*-------------------------------
PROCDIR  JSR    SETTXB     ;put filename in TXBUF2
PROCDIR1 JSR    GETKYBLK   ;get key block from file entry
PROCDIR2 JSR    GETDIR     ;read dir into memory
         BCS    HANDLERR   ;MLI error or dir too big
         JSR    SCANDIR    ;scan dir
         BCS    HANDLERR   ;dir empty or too large
         JSR    HOME
         JSR    PRTHELP2   ;print instructions
         BIT    SELFLG
         BMI    :1         ;selected line unchanged
         LDA    #0         ;select 1st line in dir
         STA    SELLIN     ; menu and start printing
         STA    TOPSCR     ; from top of dir
:1       LSR    SELFLG     ;clear select flag
:2       JSR    PRTDIR     ;print dir files
         JSR    GETNKEY    ;get command keypress
         BCC    :2         ;Arrow key changes selection
         BVS    PROCBACK   ;Open-Apple-Return gets prior dir
         BNE    TORSTART   ;Escape restarts volume search
         BPL    CKEXIT     ;QUIT
         CMP    #0
         BNE    PROCDIR    ;RTN gets next dir
         PHP               ;RTN handles UNDELETE
         PLA               ;get P-Register into Accumulator
         AND    #%00001000 ;check decimal mode
         BEQ    REPRINT    ;reprint dir after undelete
         CLD               ;clear decimal mode
         LDA    UERRCODE   ;get error code
         BNE    HANDLER1   ;always
*-------------------------------
* Check Exit From Program:
*-------------------------------
CKEXIT   JSR    CLRBOX     ;clear message box
         JSR    CKQUIT     ;check exit
         BCC    QUIT       ;YES means we're done
         LDA    ROUTEFLG
         BEQ    TORSTART   ;restart from volume menu
         BNE    REPRINT    ;always reprint dir menu
*===============================
* QUIT:
*===============================
QUIT     LDA    #22
         JSR    SETROW     ;24th row
         JMP    DOSWRM
*===============================
* MAIN LOOP ERROR HANDLER:
*===============================
HANDLERR JSR    STRIPTXB   ;strip 1 file level from TXBUF2
HANDLER1 JSR    CLRBOX     ;clear message box
         CMP    #$F0       ;error code in Accumulator
         BCC    :5         ;MLI error code
         BEQ    :1         ;DIRECTORY TOO LARGE error
         CMP    #$F1
         BEQ    :2         ;DIRECTORY EMPTY error
         CMP    #$F2
         BEQ    :3         ;BITMAP TOO LARGE error
         LDY    #TXTBLK    ;BAD BLOCK NUMBER error
         LDA    #>TXTBLK
         BNE    :4         ;always
:1       LDY    #TXTDBIG
         LDA    #>TXTDBIG
         BNE    :4         ;always
:2       LDY    #TXTDEMPT
         LDA    #>TXTDEMPT
         BNE    :4         ;always
:3       LDY    #TXTVBIG
         LDA    #>TXTVBIG
:4       JSR    PRTMSG
         BEQ    :6         ;always
:5       JSR    BADCALL    ;convert MLI error to BI error
         JSR    PRINTERR   ;print BI error message
:6       JSR    PAUSE      ;pause
         LDA    ROUTEFLG   ;check error direction
         BEQ    TORSTART   ;fatal error
REPRINT  LDA    #-1
         STA    SELFLG     ;preserve selected line
         DEC    ROUTEFLG   ;restore key block
         JSR    CLOSALL    ;close all files
         JMP    PROCDIR2   ;reprint dir menu
TORSTART JMP    RESTART    ;restart volume search
*-------------------------------
* Backup to Prior Directory:
*-------------------------------
PROCBACK LDA    ROUTEFLG
         BEQ    TORSTART   ;already in volume menu
         JSR    STRIPTXB   ;in dir menu so strip
         DEC    ROUTEFLG   ; one file level
         BEQ    TORSTART   ;now in volume menu
         LDA    #-1        ;still in dir menu so
         STA    BAKUPFLG   ; avoid bumping ROUTEFLG
         JMP    PROCDIR1   ;display prior dir
*===============================
* MAIN LOOP SUBROUTINES:
*===============================
* Print Message:
*-------------------------------
PRTMSG   STY    PPTR       ;save ptr
         STA    PPTR+1
         LDY    #0
:1       LDA    (PPTR),Y   ;get char
         BEQ    :2         ;end-of-message marker
         JSR    COUT       ;print char
         INY
         BNE    :1         ;loop back
         INC    PPTR+1     ;bump page
         BNE    :1         ;always loop back
:2       RTS
*-------------------------------
* Pause for Keypress:
*-------------------------------
```

LISTING 1: PFR.S *continued from page 22*

```
PAUSE     LDY     #TXTPAUSE
          LDA     #>TXTPAUSE
          JSR     PRTMSG          ;print "Hit a key" message
          JMP     RDKEY           ;pause for keypress
*-------------------------------
* Print Help Box:
*-------------------------------
PRTHELP1 LDX      #14             ;center volume help line
          HEX     2C              ;skip next 2-byte instruction
PRTHELP2 LDX      #1              ;center dir help line
          LDA     #19
          JSR     SETROW          ;start at 21st row
          LDA     #$1B            ;activate mouse chars
          JSR     COUT
          LDA     #$0F
          JSR     COUT
          LDA     #"S"            ;dash mouse char
          LDY     #80             ; printed 80 times
:1        JSR     COUT
          DEY
          BNE     :1
          LDA     #$0E            ;deactivate mouse chars
          JSR     COUT
          LDA     #$18
          JSR     COUT
          STX     OURCH           ;uses entry X-Register to center line
          LDY     #TXTHELP1
          LDA     #>TXTHELP1
          JSR     PRTMSG          ;print help line
          LDA     ROUTEFLG
          BEQ     :2              ;handling volume menu
          LDY     #TXTHELP2
          LDA     #>TXTHELP2
          JSR     PRTMSG          ;add to help line
:2        JSR     CROUT
          JSR     CROUT
          LDA     #29
          STA     OURCH           ;center title message
          LDY     #TXTCAT
          LDA     #>TXTCAT
          JMP     PRTMSG          ;print title message
*-------------------------------
* Close All Files:
*-------------------------------
CLOSALL   JSR     MLI             ;execute CLOSE call
          HEX     CC
          DA      CLPARN
          RTS                     ;no error expected
*-------------------------------
* Check for Selected Line:
*-------------------------------
CKSELLIN  LDA     CURLIN
          CMP     SELLIN
          BNE     :1              ;current line not selected
          LDA     DPTR            ;point SPTR at selected entry
          STA     SPTR
          LDA     DPTR+1
          STA     SPTR+1
          LDA     #$3F            ;set inverse
          STA     INVFLG          ; text mode
:1        RTS
*-------------------------------
* Get Key Block of Selected File:
*-------------------------------
GETKYBLK  LDX     ROUTEFLG
          BEQ     :1              ;no file read into DBUF
          LDY     #$11
          LDA     (SPTR),Y        ;get key subdirectory
          STA     RBUF,X          ; block LSB
          INY
          LDA     (SPTR),Y        ;get key subdirectory
          STA     RBUF+128,X      ; block MSB
:1        RTS
*-------------------------------
* Print Filename in Data Buffer:
*-------------------------------
PRTDBUF   LDY     #0
          LDX     #15             ;anticipate deleted file
          LDA     (DPTR),Y        ;get 1st byte
          AND     #$0F            ;isolate name length
          BEQ     :1              ;deleted file
          TAX                     ;X-Register is name-length counter
:1        INY                     ;Y-Register indexes DBUF
          LDA     (DPTR),Y
          BNE     :2              ;end of deleted filename
          DEY                     ;adjust index for deleted file
          BPL     :3              ;always
:2        ORA     #$80            ;convert to negative ASCII
          JSR     COUT
          DEX
          BNE     :1              ;loop back for another char
:3        CPY     #15
          BCS     :4              ;15 chars printed
          LDA     #" "            ;fill with spaces
          JSR     COUT
          INY                     ;bump char count
          BNE     :3              ;always
:4        LDA     #$FF            ;restore normal
          STA     INVFLG          ; text mode
          RTS
*-------------------------------
* Clear Instruction Box:
*-------------------------------
CLRBOX    PHA                     ;save Accumulator
          LDA     #20
          JSR     SETROW          ;22nd row
          JSR     CLREOP
          PLA                     ;restore entry Accumulator
          RTS
```

```
*-------------------------------
* Set Row:
*-------------------------------
SETROW   STA  CV
         JMP  CROUT
*-------------------------------
* Print Online Volumes:
*-------------------------------
PRTVOL   LDA  #$99        ;Control-Y
         JSR  COUT        ;home cursor with screen intact
         LDY  #TXTSLDRV
         LDA  #>TXTSLDRV
         JSR  PRTMSG      ;print column headers
         LDA  #$FF        ;prepare to zero
         STA  CURLIN      ; line count
         LDA  #DBUF       ;point at ON_LINE buffer
         STA  DPTR
         LDA  #>DBUF
         STA  DPTR+1
         LDA  #16         ;counter for 16 possible
         STA  TEMP        ; on-line volumes
:1       LDY  #0
         LDA  (DPTR),Y    ;get 1st byte in volume entry
         AND  #$0F        ;isolate length byte
         BEQ  :2          ;zero length indicates error
         INC  CURLIN      ;bump line count
         JSR  CKSELLIN    ;check if current line selected
         LDA  (DPTR),Y    ;get 1st entry byte again
         LSR              ;shift high nibble
         LSR              ; into low nibble
         LSR
         LSR
         AND  #7          ;isolate slot bits
         ORA  #"0"        ;convert to negative ASCII
         JSR  COUT        ;print slot number
         LDA  #"/"
         JSR  COUT        ;print slash
         LDA  (DPTR),Y    ;get 1st entry byte yet again
         ASL              ;drive 1=CC, drive 2=CS
         LDA  #"1"        ;default to drive 1
         ADC  #0          ;pick up carry
         JSR  COUT        ;print drive number
         JSR  PRBLNK
         JSR  PRTDBUF     ;print volume name
         JSR  CROUT
:2       CLC
         LDA  DPTR        ;bump ptr by
         ADC  #16         ; 16 chars per
         STA  DPTR        ; ON_LINE entry
         BCC  :3
         INC  DPTR+1
:3       DEC  TEMP        ;reduce volume counter
         BNE  :1          ;loop back for more volumes
         SEC              ;anticipate no on-line volume
         LDA  CURLIN
         BMI  :4          ;no on-line volume
         STA  TOTLIN      ;save total menu lines
         CLC              ;signal no error
:4       RTS              ;CC=on-line vol, CS=no on-line vol
*-------------------------------
* Strip One File Level from TXBUF2:
*-------------------------------
STRIPTXB PHA              ;save Accumulator
         LDX  TXBUF2      ;get length byte
:1       LDA  TXBUF2,X    ;get TXBUF2 char
         DEC  TXBUF2      ;reduce length
         DEX              ; until
         CMP  #"/"        ; slash found
         BNE  :1          ;no slash yet
         PLA              ;restore entry Accumulator
         RTS
*-------------------------------
* Print Directory Pathname in TXBUF2:
*-------------------------------
PRTTXB   LDX  #0
:1       LDA  TXBUF2+1,X  ;get char
         ORA  #$80        ;convert to negative ASCII
         JSR  COUT
         INX
         CPX  TXBUF2      ;check length byte
         BCC  :1          ;loop back for more
         LDA  #" "
         JSR  COUT        ;skip space
         LDA  ROUTEFLG
         JSR  PRBYTE      ;print dir level
         JMP  CROUT
*-------------------------------
* Put Directory Pathname into TXBUF2:
*-------------------------------
SETTXB   LDY  #0
         LDA  (SPTR),Y    ;get 1st byte
         AND  #$0F        ;isolate name length
         STA  TEMP        ; and save it
         LDA  #'/'        ;start with file separator
         LDX  TXBUF2
:1       INX              ;X-Register indexes TXBUF2
         INY              ;Y-Register indexes DBUF
         STA  TXBUF2,X    ;save name char in TXBUF2
         LDA  (SPTR),Y    ;get next name char from DBUF
         DEC  TEMP        ;reduce counter
         BPL  :1          ;loop back for another char
         STX  TXBUF2      ;length byte precedes name
         RTS
*-------------------------------
* Read Directory File into Memory:
*-------------------------------
GETDIR   JSR  MLI         ;execute OPEN call
         HEX  C8
         DA   OPPARM
         BCS  TOFATER1    ;MLI error
         LDA  OPREFNUM    ;stuff file ref.num
         STA  RWREFNUM    ; into R/W parmlist and
         STA  EOREFNUM    ; GET_EOF parmlist
         JSR  MLI         ;execute GET_EOF call
         HEX  D1
         DA   EOFPARM
TOFATER1 BCS  FATALERR    ;fatal error
         LDA  EOF+1       ;stuff dir file length
         STA  RWCOUNT+1   ; into R/W parmlist
         LDA  EOF
         STA  RWCOUNT
         LDA  MEMSIZ+1    ;calculate memory available
         SBC  #>DBUF-256  ; (compensate for clear carry)
         CMP  EOF+1
         BCC  DBIGERR     ;available memory < file length
         JSR  MLI         ;execute READ call
         HEX  CA
         DA   RWPARM
         BCC  :1          ;no MLI error
         CMP  #$4C
         BNE  FATALERR    ;not RANGE ERROR
:1       JSR  CLOSALL     ;close file
         LDA  FILCNT+1
         BNE  DBIGERR     ;> 255 files in dir
         LDA  DIRFILES
         CMP  #$F0
         BCC  :3          ;not in volume dir
         LDA  DEVNUM      ;stuff device number into
         STA  RWBUNUM     ; R/W_BLOCK parmlist
         LDA  HVBMPTR     ;save ptr to start of VBM
         STA  VBMPTR      ; in storage area
         LDA  HVBMPTR+1
         STA  VBMPTR+1
         LDY  HTOTBLKS+1  ;save total number of blocks
         STY  TOTBLKS+1   ; on disk device
         LDA  HTOTBLKS
         STA  TOTBLKS
         BNE  :2          ;calculate number of VBM blocks
         DEY
:2       TYA
         LSR              ;divide MSB by 16 to get
         LSR              ; number of VBM blocks-1
         LSR
         LSR
         TAY
         INY
         STY  VBMBLKS     ;save number of VBM blocks
:3       CLC              ;signal no error
         RTS              ;CC=no error, CS=error
*-------------------------------
* Main Loop Error Direction:
*-------------------------------
DBIGERR  LDA  #$F0        ;DIRECTORY TOO LARGE code
         BNE  SECRTS      ;always
DEMPTERR LDA  #$F1        ;DIRECTORY EMPTY code
         BNE  SECRTS      ;always
FATALERR LDY  #0
         STY  ROUTEFLG    ;restart volume search
SECRTS   SEC              ;flag error
         RTS
*-------------------------------
* Scan Directory:
*-------------------------------
* Set up shop:

SCANDIR  LDX  #0          ;zero index to PBUF
         STX  SCANFLG     ;get dir files on 1st pass
         LDA  FILCNT      ;get number of active
         STA  FCNTTEMP    ; files in dir
         BNE  SCANDIR1    ;active file(s) present
         DEC  SCANFLG     ;no active files, skip 1st pass

* Start at first entry in directory block:

SCANDIR1 LDA  #0          ;zero block offset
:1       STA  DIRPGOFS    ;save block offset
         LDA  ENTPBLK
         STA  EPBTEMP     ;number of file entries in block
         LDA  #DIRFILES   ;point at start of files
         STA  DPTR        ; in current block LSB
         CLC
         LDA  #>DIRFILES  ;point at start
         ADC  DIRPGOFS    ; of files in
         STA  DPTR+1      ; current block MSB

* Build table of pointers to file entries:

:2       LDA  DPTR        ;get file ptr LSB
         STA  PBUF,X      ; and save it
         LDA  DPTR+1      ;get file ptr MSB
         STA  PBUF+256,X  ; and save it
         LDY  #0
         LDA  (DPTR),Y    ;get storage.type/name.length
         TAY              ; and save it
         CMP  #$E0
         BCS  :6          ;skip header
         BIT  SCANFLG
         BMI  :5          ;scan for deleted files

* Handle directory file entries:

         TAY
         BEQ  :6          ;skip deleted file
         LDY  #$10
         LDA  (DPTR),Y    ;get file.type
         CMP  #$0F
         BNE  :4          ;skip nondirectory file
         STA  PBUF+512,X  ;save file.type code
         INX              ;bump entry count
:3       BEQ  DBIGERR     ;> 256 lines gives range error
:4       DEC  FCNTTEMP    ;reduce active file count
         BNE  :6          ;more active files to scan
         DEC  SCANFLG     ;get deleted files on 2nd pass
         BMI  SCANDIR1    ;always

* Handle deleted file entries:

:5       TAY
         BNE  :6          ;skip active file
         LDY  #1
```

LISTING 1: PFR.S *(continued)*

```
          LDA    (DPTR),Y      ;get 1st char in filename
          BEQ    :8            ;zero means end of entries
          LDA    #0
          STA    PBUF+512,X    ;save deleted 'file.type' code
          INX                  ;bump entry count
          BEQ    DBIGERR       ;>256 lines gives range error

* Advance to next entry in block:

:6        CLC
          LDA    DPTR          ;advance to next file
          ADC    ENTLEN        ; in current block
          STA    DPTR
          BCC    :7
          INC    DPTR+1
:7        DEC    EPBTEMP       ;decrement files per block count
          BNE    :2            ;not at end of block

* Advance to next block in directory:

          LDA    #DBUF         ;point at start
          STA    DPTR          ; of current
          DEC    DPTR+1        ; dir block
          LDY    #2
          LDA    (DPTR),Y      ;get forward link LSB
          INY
          ORA    (DPTR),Y      ;OR with forward link MSB
          BEQ    :8            ;end of dir
          CLC                  ;more dir blocks so
          LDA    DIRPGOFS      ; advance to next block
          ADC    #2            ; (2 pages per block)
          JMP    :1

* Final housekeeping:

:8        CPX    #0
          BEQ    :10           ;no DIR or DEL files
          DEX
          STX    TOTLIN        ;save total number of lines-1
          BIT    BAKUPFLG
          BMI    :9
          INC    ROUTEFLG      ;bump flag so key block secure
:9        LSR    BAKUPFLG      ;clear backup flag
          CLC                  ;signal no error
          RTS                  ;CC=no error, CS=error
:10       JMP    DEMPTERR      ;dir empty
*-----------------------------
* Print Directory:
*-----------------------------
PRTDIR    LDA    #$99          ;Control-Y
          JSR    COUT          ;home cursor with screen intact
          JSR    PRTTXB        ;print dir name
          JSR    CKTOPARW      ;check for more files at top
          LDA    #0            ;1st file column is at
          STA    COLUMN        ; left edge of screen
          LDA    #64           ;no more than 64
          STA    MAXSCR        ; files on screen
          LDX    TOPSCR        ;set 1st screen line
          DEX                  ;prepare to zero PBUF index
          STX    CURLIN        ; and line count
:1        LDA    #1
          JSR    SETROW        ;3rd row
          LDA    #16           ;no more than 16 lines
          STA    ROW           ; in one column
:2        INX                  ;bump PBUF index
          INC    CURLIN        ;bump line count
          JSR    PRTENTRY      ;print file entry
          CPX    TOTLIN
          BCS    CKBOTARW      ;last dir line printed
          DEC    MAXSCR        ;reduce max screen file count
          BEQ    CKBOTARW      ;check for more files at bottom
          DEC    ROW           ;reduce max row count
          BNE    :2            ;room for more files in column
          LDA    COLUMN        ;bump file column
          ADC    #20           ; (20 chars per column)
          STA    COLUMN
          BNE    :1            ;always
*-----------------------------
* Check/Flag Unprinted Files at Top:
*-----------------------------
CKTOPARW  LDA    TOPSCR
          BEQ    :1            ;printing 1st file in dir
          LDY    #TXTTARW
          LDA    #>TXTTARW
          JSR    PRTMSG        ;signal more files above
:1        JSR    CLREOL        ;clear arrows, if present
          JMP    CROUT
*-----------------------------
* Check/Flag Unprinted Files at Bottom:
*-----------------------------
CKBOTARW  LDA    COLUMN
          STA    OURCH         ;set file column
          LDA    MAXSCR
          BNE    :1            ;no more files in dir
          LDY    #TXTBARW
          LDA    #>TXTBARW
          JMP    PRTMSG        ;signal more files below
:1        JMP    CLREOL
*-----------------------------
* Print Directory Entry:
*-----------------------------
PRTENTRY  TXA
          PHA                  ;save X-Register
          LDA    PBUF,X        ;set ptr to file
          STA    DPTR
          LDA    PBUF+256,X
          STA    DPTR+1
          LDA    COLUMN
          STA    OURCH         ;set file column
          JSR    CKSELLIN      ;check if current line selected
          LDA    PBUF+512,X    ;get file type code
          TAY
          BEQ    :1            ;deleted file
          LDY    #3            ;dir file
:1        LDX    #3            ;count file type text
:2        LDA    FITYPTBL,Y    ;get file type char
          JSR    COUT          ; and print it
          INY                  ;bump table index
          DEX                  ;reduce counter
          BNE    :2            ;loop back for another char
          LDA    #" "
          JSR    COUT          ;print space
          JSR    PRTDBUF       ;print file entry
          JSR    CROUT
          PLA
          TAX                  ;restore entry X-Register
          RTS
*-----------------------------
* Check Quitting:
*-----------------------------
CKQUIT    LDY    #TXTQUIT
          LDA    #>TXTQUIT
          JSR    PRTMSG        ;print quit message
          BEQ    :2            ;always
:1        JSR    BELL
:2        JSR    RDKEY         ;get response
          AND    #$DF          ;upshift
          CMP    #"N"
          BEQ    PRTANSW       ;NO
          CMP    #"Y"
          BNE    :1
          CLC                  ;signal YES response
PRTANSW   PHP
          JSR    COUT          ;print response
          PLP
          RTS                  ;CS=NO, CC=YES
*-----------------------------
* Get Menu Keypress:
*-----------------------------
*Exit state of P-Reg.: N V - B D I Z C    Acc.   Menu Use
*               ARROW: - - - - 0 - - 0     -        VD
*           PRIOR.DIR: - 1 - - 0 - - 1     -        D
*                 ESC: - 0 - - 0 - 0 1     -        VD
*                QUIT: 0 0 - - 0 - 1 1     -        D
*            NEXT.DIR: 1 0 - - 0 - 1 1   not 0      D
*       GOOD.UNDELETE: 1 0 - - 0 - 1 1     0        D
*        BAD.UNDELETE: 1 0 - - 1 - 1 1     0        D

GMKEYERR  JSR    BELL
GETMKEY   BIT    BUTN0
          BPL    :1            ;Open-Apple key not down
          BIT    RTS1          ;Open-Apple key down so set V-flag
          BVS    :2            ;always
:1        CLV                  ;Open-Apple key not down so clear V-flag
:2        LDA    KEY           ;check keypress
          BPL    GETMKEY       ;no keypress
          STA    STROBE        ;got keypress so clear strobe
          LDX    SELLIN        ;X-Register=original selected line
          CMP    #$E0
          BCC    :3            ;upper case entered
          AND    #$DF          ;upshift
:3        CMP    #$8A          ;down arrow
          BEQ    DWNARW
          CMP    #$8B          ;up arrow
          BEQ    UPARW
          CMP    #$95          ;-->
          BEQ    FWDARW
          CMP    #$88          ;<--
          BEQ    BCKARW
          CMP    #"Q"          ;quit
          BEQ    :4
          CMP    #$8D          ;examine/undelete
          BEQ    RTN
          CMP    #$9B          ;Escape
          BNE    GMKEYERR      ;invalid keypress
          LDA    #$9B          ;clear Z-flag
:4        CLV
RTS1      RTS
*-----------------------------
* UP ARROW Handler:
*-----------------------------
UPARW     LDY    SELLIN
          BEQ    UPARW1        ;1st line currently selected
          CPY    TOPSCR
          BNE    :1            ;top of screen not selected
          DEC    TOPSCR        ;start displaying 1 higher line
:1        DEY                  ;select prior line
ENDARW    STY    SELLIN        ;set selected line
          CLC                  ;signal arrow key
          RTS
UPARW1    LDY    CURLIN        ;select bottom line on screen
          JMP    ENDARW
*-----------------------------
* DOWN ARROW Handler:
*-----------------------------
DWNARW    LDY    SELLIN
          CPY    TOTLIN
          BEQ    DWNARW1       ;last line currently selected
          CPY    CURLIN
          BNE    :1            ;bottom of screen not selected
          INC    TOPSCR        ;start displaying 1 lower line
:1        INY                  ;select next line
          BNE    ENDARW        ;always
DWNARW1   LDY    TOPSCR        ;select top line on screen
          JMP    ENDARW
*-----------------------------
* FORWARD ARROW Handler:
*-----------------------------
FWDARW    CLC
          LDA    SELLIN        ;get selected line
          TAY                  ;save selected line in Y-Register
          ADC    #16           ;16 lines per column
          CMP    CURLIN
```

LISTING 1: PFR.S *(continued)*

```
          BEQ   GOODARW    ;valid -->
          BCS   BADARW     ;invalid -->
GOODARW   TAY              ;select new line
BADARW    JMP   ENDARW
*-------------------------------
* BACK ARROW Handler:
*-------------------------------
BCKARW    SEC
          LDA   SELLIN     ;get selected line
          TAY
          SBC   #16        ;16 lines per column
          BCC   BADARW     ;invalid <--
          PHA
          SBC   TOPSCR
          PLA
          BCC   BADARW     ;invalid <--
          BCS   GOODARW    ;always valid <--
*-------------------------------
* RETURN Handler:
*-------------------------------
RTN       LDA   ROUTEFLG   ;in volume dir all RTNs
          BEQ   RTS1       ; mean get volume dir
          BVS   RTS1       ;BACKUP to prior dir
          LDA   #%10000011 ;becomes value of P-Register
          PHA              ;save on stack
          LDY   #0
          LDA   (SPTR),Y   ;get storage.type/name.length
          BEQ   UNDELETE   ;UNDELETE: A = 0 ; CS,EQ,VC,MI
          PLP              ;NEXT.DIR: A <> 0; CS,EQ,VC,MI
          RTS
*-------------------------------
* UNDELETE Error Direction:
*-------------------------------
UBLKERR   LDA   #$F3       ;BAD BLOCK code
          HEX   2C         ;skip next 2-byte instruction
UVBIGERR  LDA   #$F2       ;BITMAP TOO LARGE code
UFATERR   STA   UERRCODE   ;save error code
          JMP   UBADRTN
*===============================
* UNDELETE FILE:
*===============================
UNDELETE  PLA              ;reset stack
          LSR   WRIXFLG    ;clear write index block flag
          JSR   SETDSPLY   ;set display in message box
          JSR   GETVBMBF   ;put VBM buffer into parmlist
          BCS   UVBIGERR   ;not enough space for VBM
          JSR   RDVBM      ;read VBM into memory
          BCS   UFATERR    ;fatal error
          JSR   ZBLKS      ;zero block count fields
          JSR   PRTBLKS    ; and print result
          JSR   GETUKYBL   ;get key block of target file
          JSR   FIXVBM     ;reserve key block in VBM
          BCS   UBLKERR    ;invalid block number
          LDX   STORTYP
          DEX
          BEQ   DOUNDEL    ;seedling file
          JSR   SETKYBLK   ;set parms for key block
          CPX   #$0C       ;X-Register reduced by one
          BEQ   DIRECTRY   ;dir file
          JSR   DOIX       ;prepare key index block
          BVS   UBLKERR    ;invalid block number
          BCS   UFATERR    ;fatal error
          DEX
          BEQ   DOUNDEL    ;sapling file
*-------------------------------
* TREE File Handler:
*-------------------------------
          JSR   DOSUBIX    ;prepare subindex block(s)
          BVS   UBLKERR    ;invalid block number
          BCS   UFATERR    ;fatal error
          BCC   DOUNDEL    ;always
*-------------------------------
* DIRECTORY File Handler:
*-------------------------------
DIRECTRY  JSR   DODIR      ;prepare dir header
          BVS   UBLKERR    ;invalid block number
          BCS   UFATERR    ;fatal error
*-------------------------------
* Undelete File:
*-------------------------------
DOUNDEL   JSR   CKUNDEL    ;confirm undeletion
          BCS   UGOODRTN   ;negative response so depart
          LDY   #0
          LDA   STYPNLEN   ;put storage.type/name.length
          STA   (SPTR),Y   ; into file entry
          INC   FILCNT     ;bump file.count
          BNE   :1
          INC   FILCNT+1
:1        JSR   WRKEYSIX   ;write key/subindex block(s)
          BVS   UBLKERR    ;invalid block number
          BCS   UFATERR    ;fatal error
          JSR   WRITVBM    ;write VBM to disk
          BCS   UFATERR    ;fatal error
          JSR   WRITDIR    ;write altered dir to disk
          BCS   UFATERR    ;fatal error
*-------------------------------
* Reset Menu to Compensate for Deleted File:
*-------------------------------
          LDA   TOTLIN     ;if undeleted file was only file
          BEQ   UBAKRTN    ; in menu backup to prior dir
          LDA   SELLIN     ;if undeleted file not
          CMP   TOTLIN     ; at end of menu don't
          BNE   UGOODRTN   ; change selection else
          DEC   SELLIN     ; select next higher entry
*-------------------------------
* Return to Main Program Loop:
*-------------------------------
UGOODRTN  LDA   #%10000011 ;MI,VC,DC,EQ,CS
          HEX   2C         ;skip next 2-byte instruction
UBAKRTN   LDA   #%01000001 ;VS,CS
          HEX   2C         ;skip next 2-byte instruction
UBADRTN   LDA   #%10001011 ;MI,VC,DS,EQ,CS
          PHA              ;save on stack
          LDA   #0         ;signal undeletion
          PLP              ;set P-Register for return
          RTS
*===============================
* UNDELETE SUBROUTINES:
*===============================
* Set Display in Message Box
*-------------------------------
SETDSPLY  JSR   CLRBOX     ;clear message box
          LDA   SPTR       ;copy file location from
          STA   DPTR       ; SPTR to DPTR
          LDA   SPTR+1
          STA   DPTR+1
          JSR   PRTDBUF    ;print deleted filename

* Calculate storage type and name length of deleted file:

          TXA
          EOR   #$0F       ;get name.length
          STA   STYPNLEN   ; and save it
          JSR   PRBLNK
          LDX   #$0D       ;anticipate dir file
          LDY   #$10
          LDA   (SPTR),Y
          CMP   #$0F
          BEQ   :3         ;got dir file
          LDX   #3         ;anticipate tree file
          LDY   #$17
          LDA   (SPTR),Y   ;get high-order EOF byte
          BEQ   :1         ;not tree (EOF < $10000)
          CMP   #2
          BCC   :2         ;sapling (EOF $10000-$1FFFF)
          BNE   :3         ;tree (EOF >= $30000)
          DEY
          LDA   (SPTR),Y   ;get mid order EOF byte
          DEY
          ORA   (SPTR),Y   ;OR with low-order EOF byte
          BEQ   :2         ;sapling (EOF = $20000)
          BNE   :3         ;tree (EOF $20001-$2FFFF)
:1        LDY   #$15
          LDA   #0         ;compare $200 with EOF
          CMP   (SPTR),Y   ;(lo order EOF byte)
          INY
          LDA   #2
          SBC   (SPTR),Y   ;subtract mid order EOF byte
          BCC   :2         ;sapling (EOF > $200)
          DEX              ;seedling (EOF <= $200)
:2        DEX
:3        STX   STORTYP    ;save storage type
          TXA
          ASL              ;shift to high nibble
          ASL
          ASL
          ASL
          ORA   STYPNLEN
          STA   STYPNLEN   ;save storage.type/name.length

* Print storage type of deleted file:

          DEX
          BEQ   :6         ;seedling file
          DEX
          BEQ   :5         ;sapling file
          DEX
          BEQ   :4         ;tree file
          LDY   #TXTDIR    ;directory file
          LDA   #>TXTDIR
          BNE   :7         ;always
:4        LDY   #TXTTREE
          LDA   #>TXTTREE
          BNE   :7         ;always
:5        LDY   #TXTSAP
          LDA   #>TXTSAP
          BNE   :7         ;always
:6        LDY   #TXTSEED
          LDA   #>TXTSEED
:7        JSR   PRTMSG

* Setup to display status of deleted file blocks:

          JSR   PRBLNK
          LDY   #TXTFREE
          LDA   #>TXTFREE
          JSR   PRTMSG     ;print "Blocks Free"
          LDA   OURCH
          STA   FREEPOSN   ;save FREE field position
          LDX   #7
          JSR   PRBL2
          LDY   #TXTUSED
          LDA   #>TXTUSED
          JSR   PRTMSG     ;print "Blocks Used"
          LDA   OURCH
          STA   USEDPOSN   ;save USED field position
          RTS
*-------------------------------
* Write Directory to Disk:
*-------------------------------
WRITDIR   LDX   ROUTEFLG
          LDA   RBUF-1,X   ;put key dir block number
          STA   RWBLKNUM   ; into R/W_BLOCK parmlist
          LDA   RBUF-1+128,X
          STA   RWBLKNUM+1
          LDA   #DBUF      ;point at start of
          STA   DPTR       ; dir LSB
          STA   RWBLKBUF   ;set R/W_BLOCK buffer LSB
          LDA   #>DBUF     ;point at dir start MSB
```

```
:1        STA   DPTR+1      ;save dir ptr MSB
          STA   RWBLKBUF+1  ;set R/W_BLOCK buffer MSB
          JSR   WRITBLK     ;execute WRITE_BLOCK call
          BCS   :3          ;MLI error
          LDY   #2          ;index 1st forward link byte
          LDA   (DPTR),Y    ;get link LSB to next block
          STA   RWBLKNUM    ; and stuff it into parmlist
          INY
          ORA   (DPTR),Y    ;if both link bytes are zero,
          BEQ   :2          ; no more blocks in dir
          LDA   (DPTR),Y    ;get link MSB to next block
          STA   RWBLKNUM+1  ; and stuff it into parmlist
          LDA   DPTR+1
          ADC   #2          ;add 1 block to block offset
          BNE   :1          ;always
:2        CLC               ;signal no error
:3        RTS               ;CS=error, CC=no error
*-------------------------------
* Get Start of VBM Buffer:
*-------------------------------
GETVBMBF  LDA   #0          ;buffer always on
          STA   RWBLKBUF    ; page boundary
          CLC
          LDA   EOF+1       ;add dir pages to
          ADC   #>DBUF      ; start of dir to get
          STA   RWBLKBUF+1  ; start of buffer for VBM blocks
          STA   VBUF        ;save start of VBM buffer
          ADC   VBMBLKS     ;add pages used
          ADC   VBMBLKS     ; by VBM
          CMP   MEMSIZ+1
          BEQ   :1          ;VBM just below HIMEM
          BCS   :2          ;VBM pages > available memory
:1        CLC               ;signal no error
:2        RTS               ;CS=space ng, CC=space ok
*-------------------------------
* Read/Write VBM into Memory:
*-------------------------------
WRITVBM   LDA   #0          ;stuff VBM buffer into
          STA   RWBLKBUF    ; WRITE_BLOCK parmlist
          LDA   VBUF
          STA   RWBLKBUF+1
          LDA   #$81        ;WRITE_BLOCK code
          HEX   2C          ;skip next 2-byte instruction
RDVBM     LDA   #$80        ;READ_BLOCK code
          STA   RWBLKCMD
          LDX   VBMBLKS     ;count number of VBM blocks
          LDA   VBMPTR      ;stuff starting block number of
          STA   RWBLKNUM    ; VBM into R/W_BLOCK parmlist
          LDA   VBMPTR+1
          STA   RWBLKNUM+1
RWVBM     JSR   MLI         ;execute R/W_BLOCK call
RWBLKCMD  HEX   00
          DA    RWBLPARM
          BCS   :1          ;MLI error
          DEX
          BEQ   :1          ;all VBM blocks read into memory
          LDA   RWBLKBUF+1  ;bump buffer block
          ADC   #2
          STA   RWBLKBUF+1
          INC   RWBLKNUM    ;bump block number
          BNE   RWVBM
          INC   RWBLKNUM+1
          BNE   RWVBM       ;always
:1        RTS               ;CS=fatal error, CC=no error
*-------------------------------
* Zero Free/Used Block Count:
*-------------------------------
ZBLKS     LDA   #0
          STA   FREEBLKS    ;zero count of free and reserved
          STA   FREEBLKS+1  ; blocks in target deleted file
          STA   USEDBLKS
          STA   USEDBLKS+1
          RTS
*-------------------------------
* Increment Number of Free/Used Blocks in Deleted File:
*-------------------------------
INCDSPLY  BEQ   :1          ;reserved block found
          INC   FREEBLKS    ;bump free block count
          BNE   PRTBLKS
          INC   FREEBLKS+1
          BNE   PRTBLKS     ;always
:1        INC   USEDBLKS    ;bump used block count
          BNE   PRTBLKS
          INC   USEDBLKS+1  ;fall into print code
*-------------------------------
* Print Count of Free/Used Blocks in Deleted File:
*-------------------------------
PRTBLKS   LDA   FREEPOSN
          STA   OURCH       ;set FREE BLOCKS position
          LDA   FREEBLKS+1
          JSR   PRBYTE      ;print FREE BLOCKS count
          LDA   FREEBLKS
          JSR   PRBYTE
          LDA   USEDPOSN
          STA   OURCH       ;set USED BLOCKS position
          LDA   USEDBLKS+1
          JSR   PRBYTE      ;print USED BLOCKS count
          LDA   USEDBLKS
          JMP   PRBYTE
*-------------------------------
* Get Key Block of Deleted File:
*-------------------------------
GETUKYBL  LDY   #$11
          LDA   (SPTR),Y    ;get key pointer and stuff in
          STA   ACC+1       ; my accumulator (MSB/LSB)
          STA   KYFILBLK    ; and storage area
          INY
```

```
         LDA    (SPTR),Y
         STA    ACC
         STA    KYFILBLK+1
         RTS
*-------------------------------
* Reserve Block in VBM and Display Block Count:
*-------------------------------
FIXVBM   TYA
         PHA               ;preserve Y-Register
         BIT    WRIXFLG
         BMI    :1         ;VBM already changed
         LDA    TOTBLKS    ;check valid block number
         CMP    ACC+1
         LDA    TOTBLKS+1
         SBC    ACC
         BCC    :2         ;invalid block number
         LDA    ACC+1      ;get block number LSB
         AND    #7         ;isolate bits 0-2 and use as
         TAY               ; index to
         LDA    VBMSKTBL,Y ; lookup table of bit positions
         STA    VBMBIT     ;save bit position mask
         EOR    #$FF
         STA    VBMMSK     ;save mask to reserve target bit
         LSR    ACC        ;divide block number by 8
         ROR    ACC+1
         LSR    ACC
         ROR    ACC+1
         LSR    ACC        ;page offset into VBM buffer
         ROR    ACC+1      ;byte offset into VBM page
         LDY    ACC+1      ;index target byte
         CLC
         LDA    VBUF       ;add starting page of VBM buffer
         ADC    ACC        ; to page offset into VBM buffer
         STA    DPTR+1     ; and point at target page
         LDA    #0         ; which is always
         STA    DPTR       ; on page boundary
         LDA    (DPTR),Y   ;get target byte
         PHA               ;save it
         AND    VBMBIT     ;EQ=block used, NE=block free
         JSR    INCDSPLY   ;update block display
         PLA               ;recover target byte
         AND    VBMMSK     ;clear target bit to indicate
         STA    (DPTR),Y   ; that block is reserved
:1       CLC               ;signal no error
         HEX    24         ;skip next 1-byte instruction
:2       SEC               ;signal error
         PLA
         TAY               ;restore entry Y-Register
         RTS               ;CC=no error, CS=error
*-------------------------------
* Check Undeletion:
*-------------------------------
CKUNDEL  JSR    CROUT
         LDX    #"Y"       ;set affirmative default
         LDA    USEDBLKS
         ORA    USEDBLKS+1
         BEQ    :1         ;no used blocks in deleted file
         JSR    SETINV
         LDY    #TXTINUSE
         LDA    #>TXTINUSE
         JSR    PRTMSG     ;print "BLOCK(S) IN USE" message
         JSR    SETNORM
         LDX    #"N"       ;change to negative default
:1       JSR    CROUT
         LDY    #TXTUNDEL
         LDA    #>TXTUNDEL
         JSR    PRTMSG     ;print the critical question
         TXA
         JSR    COUT       ;output the default response
         LDA    #$88
         JSR    COUT       ;backspace
         JMP    :3
:2       JSR    BELL
:3       JSR    RDKEY
         CMP    #$E0
         BCC    :4         ;upper case entered
         AND    #$DF       ;get rid of lower case
:4       CMP    #"Y"
         BEQ    :5         ;YES keypress
         CMP    #"N"
         BEQ    :6         ;NO keypress
         CMP    #$8D
         BNE    :2         ;invalid keypress
         TXA
         CMP    #"N"       ;check default response
         BEQ    :6         ;NO
:5       CLC               ;signal YES response
:6       JMP    PRTANSW    ;CS=NO, CC=YES
*-------------------------------
* Prepare Index Block for Undeletion:
*-------------------------------
DOIX     STY    RWBLKNUM
         STA    RWBLKNUM+1
         JSR    RDBLK      ;execute READ_BLOCK call
         BCS    :3         ;MLI error
         LDY    #0         ;block index
:1       LDA    (IXPTR),Y  ;get block number MSB and
         STA    ACC        ; stuff in my accumulator
         INC    IXPTR+1    ;2nd page
         ORA    (IXPTR),Y
         BEQ    :2         ;no block number given
         LDA    (IXPTR),Y  ;get block number LSB and
         STA    ACC+1      ; stuff in my accumulator
         DEC    IXPTR+1    ;1st page
         STA    (IXPTR),Y  ;reverse MSB and LSB
         LDA    ACC
         INC    IXPTR+1    ;2nd page
         STA    (IXPTR),Y
         JSR    FIXVBM     ;reserve block in VBM
         BCS    :4         ;invalid block number
:2       DEC    IXPTR+1    ;1st page
         INY               ;bump index to index block
         BNE    :1         ;loop back for another block
         BIT    WRIXFLG
         BPL    :3         ;not writing block to disk
         JSR    WRITBLK    ;execute WRITE_BLOCK call
:3       CLV
         RTS               ;CC=no error, CS,VC=fatal error
:4       BIT    RTS1       ;set overflow flag
         RTS               ;CS,VS=bad block number error
*-------------------------------
* Prepare Subindex Block(s) for Undeletion:
*-------------------------------
DOSUBIX  LDY    #SBUF      ;set subindex block buffer
         LDA    #>SBUF     ; in R/W_BLOCK parmlist
         STY    RWBLKBUF
         STA    RWBLKBUF+1
         STY    IXPTR      ;point to subindex block buffer
         STA    IXPTR+1
         LDX    #0         ;zero block index
:1       STX    XSAV       ;save block index
         LDA    KBUF,X     ;get block number LSB
         TAY               ; and stuff in Y-Register
         LDA    KBUF+256,X ;get block number MSB in Accumulator
         JSR    DOIX       ;prepare & write subindex block
         BCS    RTS2       ;CS=fatal or bad block # error
         LDX    XSAV       ;restore block index
         INX
         BPL    :1         ;max of 128 subindex blocks
RTS2     RTS               ;CC=no error, CS/VS=error
*-------------------------------
* Write Key Block or Subindex Block(s) to Disk:
*-------------------------------
WRKEYSIX CLC               ;anticipate no error
         CLV
         LDX    STORTYP
         DEX
         BEQ    RTS2       ;seedling has no index block
         DEX
         BEQ    WRITBLK    ;sapling has 1 index block
         JSR    SETKYBLK   ;set parms for key block
         JSR    WRITBLK    ;write key block to disk
         BCS    RTS2       ;MLI error
         DEX
         BNE    RTS2       ;dir has no subindex block(s)
         LDA    #-1        ;tree has subindex block(s)
         STA    WRIXFLG    ;set write index block flag
         JMP    DOSUBIX    ;write subindex block(s)
*-------------------------------
* Execute WRITE_BLOCK Call:
*-------------------------------
WRITBLK  JSR    MLI
         HEX    81
         DA     RWBLPARM
         CLV               ;avoid bad block number error
         RTS               ;CC=no error, CS,VC=fatal error
*-------------------------------
* Execute READ_BLOCK Call:
*-------------------------------
RDBLK    JSR    MLI
         HEX    80
         DA     RWBLPARM
         RTS               ;CC=no error, CS=fatal error
*-------------------------------
* Set Parameters for Key Block:
*-------------------------------
SETKYBLK LDY    #KBUF      ;tree has 1 or more index blocks
         LDA    #>KBUF     ;set key buffer in
         STY    RWBLKBUF   ; R/W_BLOCK parmlist
         STA    RWBLKBUF+1
         STY    IXPTR      ;point to key index block buffer
         STA    IXPTR+1
         LDY    KYFILBLK   ;set key block number in
         LDA    KYFILBLK+1 ; R/W_BLOCK parmlist
         STY    RWBLKNUM
         STA    RWBLKNUM+1
         RTS
*-------------------------------
* Prepare Directory File for Undeletion:
*-------------------------------
DODIR    JSR    RDBLK      ;read key block into key buffer
         BCS    :3         ;MLI error
         LDA    #1
         STA    DBLKS      ;count dir blocks
         LDY    #SBUF      ;set buffer for nonkey blocks
         LDA    #>SBUF
         STY    RWBLKBUF
         STA    RWBLKBUF+1
         LDY    KBUF+2     ;get forward link LSB
         TYA
         ORA    KBUF+3     ;OR with forward link MSB
         BEQ    :2         ;done
         LDA    KBUF+3     ;get forward link MSB
:1       STY    ACC+1      ;stuff block number in my
         STY    RWBLKNUM   ; accumulator (MSB/LSB)
         STA    ACC        ; and R/W_BLOCK parmlist
         STA    RWBLKNUM+1
         JSR    FIXVBM     ;reserve block in VBM
         BCS    :4         ;invalid block number
         JSR    RDBLK      ;read next dir block
         BCS    :3         ;MLI error
         INC    DBLKS      ;bump block count
         LDY    SBUF+2     ;get forward link LSB
         TYA
         ORA    SBUF+3     ;OR with forward link MSB
         BEQ    :2         ;done
         LDA    SBUF+3     ;get forward link MSB
         BCC    :1         ;always loop back
:2       CLC
         LDA    STYPNLEN   ;put storage_type/name_length
         ADC    #$10       ; plus $10 (i.e. $EX)
         STA    KBUF+4     ; into subdir header
         LDY    #$13
         LDA    (SPTR),Y   ;get blocks_used in file entry
         CMP    DBLKS
         SEC               ;anticipate error
         BNE    :4         ;invalid key block
         CLC               ;flag no error
```

```
:3        CLV
          RTS                   ;CC=no error, CS,VC=fatal error
:4        BIT   RTS1            ;set overflow flag
          RTS                   ;CS,VS=bad block number error
*==============================
* PARAMETER LISTS:
*==============================
OLPARM    HEX   Ø2              ;ON_LINE PARMLIST
          HEX   ØØ              ;unit.num (all volumes)
          DA    DBUF            ;data.buffer

EOFPARM   HEX   Ø2              ;SET/GET_EOF PARMLIST
EOREFNUM  HEX   ØØ              ;ref.num
EOF       DS    3               ;EOF

OPPARM    HEX   Ø3              ;OPEN PARMLIST
          DA    TXBUF2          ;path.pointer
          DA    OBUF            ;io.buffer
OPREFNUM  HEX   ØØ              ;ref.num

RWPARM    HEX   Ø4              ;READ/WRITE PARMLIST
RWREFNUM  HEX   ØØ              ;ref.num
          DA    DBUF            ;data.buffer
RWCOUNT   DA    Ø               ;request.count
          DA    Ø               ;trans.count

CLPARM    HEX   Ø1              ;CLOSE PARMLIST
          HEX   ØØ              ;ref.num (all files)

RWBLPARM  HEX   Ø3              ;READ/WRITE_BLOCK PARMLIST
RWBUNUM   HEX   ØØ              ;unit.num
RWBLKBUF  DA    Ø               ;data.buffer
RWBLKNUM  DA    Ø               ;block.num
*==============================
* STORAGE:
*==============================
ROUTEFLG  HEX   ØØ              ;EQ=volume, NE=directory
SELFLG    HEX   ØØ              ;PL=select top, MI=select same
BAKUPFLG  HEX   ØØ              ;PL=no backup, MI=backup
SCANFLG   HEX   ØØ              ;PL=DIR file, MI=deleted file
WRIXFLG   HEX   ØØ              ;PL=no write, MI=write index
TEMP      HEX   ØØ              ;temporary storage
XSAV      HEX   ØØ              ;store X-Register
DIRPGOFS  HEX   ØØ              ;page offset into dir
FCNTTEMP  HEX   ØØ              ;active files remaining
EPBTEMP   HEX   ØØ              ;file entries per block
COLUMN    HEX   ØØ              ;column-1 of dir printout
ROW       HEX   ØØ              ;row of dir printout
SELLIN    HEX   ØØ              ;selected line-1 in file menu
CURLIN    HEX   ØØ              ;current/bottom line-1 in menu
TOPSCR    HEX   ØØ              ;top screen line-1 of dir
MAXSCR    HEX   ØØ              ;maximum lines on screen
TOTLIN    HEX   ØØ              ;total lines-1 in dir
UERRCODE  HEX   ØØ              ;error code during undeletion
*------------------------------
STORTYP   HEX   ØØ              ;storage type
STYPNLEN  HEX   ØØ              ;storage type/name.length
TOTBLKS   DS    2               ;total blocks on disk device
VBMPTR    DS    2               ;number of 1st VBM block
VBUF      HEX   ØØ              ;starting page of VBM buffer
VBMBLKS   HEX   ØØ              ;number of VBM blocks
VBMBIT    HEX   ØØ              ;bit position mask in VBM byte
VBMMSK    HEX   ØØ              ;mask to clear target bit
FREEBLKS  DS    2               ;number of free blocks
USEDBLKS  DS    2               ;number of used blocks
FREEPOSN  HEX   ØØ              ;position of free blocks display
USEDPOSN  HEX   ØØ              ;position of used blocks display
ACC       DS    2               ;my accumulator (MSB/LSB)
KYFILBLK  DS    2               ;key block number of target file
DBLKS     DS    2               ;count of dir blocks
*==============================
* TABLES:
*==============================
FITYPTBL  ASC   "DEL" ;file type table
          ASC   "DIR"

VBMSKTBL  HEX   8Ø,4Ø,2Ø,1Ø,Ø8,Ø4,Ø2,Ø1  ;VBM byte masks
*==============================
* TEXT:
*==============================
TXTHELP1  ASC   "ESC-restart   Q-quit   "
          ASC   "ARROWS-select   RTN-next.dir",ØØ
TXTHELP2  ASC   "/undelete    "
          HEX   1B,ØF
          ASC   "A"
          HEX   ØE,18
          ASC   "RTN-prior.dir",ØØ
TXTCAT    ASC   "ProDOS FILE RECOVERY",ØØ
TXTSLDRV  ASC   "S/D   VOLUME NAME",8D,8D,ØØ
TXTPAUSE  ASC   "Hit a key ",ØØ
TXTTARW   HEX   1B,ØF
          HEX   D2,1C,D2,1C,D2,1C,D2,1C,D2
          HEX   ØE,18,ØØ
TXTBARW   HEX   1B,ØF
          HEX   D1,1C,D1,1C,D1,1C,D1,1C,D1
          HEX   ØE,18,ØØ
TXTQUIT   ASC   "Quit (Y/N)? ",ØØ
TXTDIR    ASC   "(Directory)",ØØ
TXTTREE   ASC   "(Tree)",ØØ
TXTSAP    ASC   "(Sapling)",ØØ
TXTSEED   ASC   "(Seedling)",ØØ
TXTFREE   ASC   "Free Blocks: $",ØØ
TXTUSED   ASC   "Used Blocks: $",ØØ
TXTINUSE  ASC   "BLOCK(S) IN USE",87,ØØ
TXTUNDEL  ASC   "Undelete the file (Y/N)? ",ØØ
TXTDBIG   HEX   8D
          ASC   "DIRECTORY TOO LARGE",87,8D,ØØ
TXTDEMPT  HEX   8D
          ASC   "DIRECTORY EMPTY",87,8D,ØØ
TXTVBIG   HEX   8D
          ASC   "BITMAP TOO LARGE",87,8D,ØØ
TXTBLK    HEX   8D
          ASC   "BAD BLOCK NUMBER",87,8D,ØØ
END
          PAG
```

END OF LISTING 1

```
               KEY PERFECT 5.Ø
                    RUN ON
                     PFR
==========================================
   CODE-5.Ø    ADDR# - ADDR#    CODE-4.Ø
   --------    -------------    --------
   B497611C    Ø9ØØ - Ø94F        293C
   BC3544BD    Ø95Ø - Ø99F        2838
   81679A53    Ø9AØ - Ø9EF        2774
   ADB2BDF3    Ø9FØ - ØA3F        29A1
   2DC2A286    ØA4Ø - ØA8F        2847
   864E6FF2    ØA9Ø - ØADF        2CC4
   A36ØD87F    ØAEØ - ØB2F        269Ø
   7123B6F9    ØB3Ø - ØB7F        27F4
   C8C1B3AØ    ØB8Ø - ØBCF        2A7E
   DAØD18DA    ØBDØ - ØC1F        2878
   69CEBD41    ØC2Ø - ØC6F        2A9Ø
   F1743EBØ    ØC7Ø - ØCBF        255E
   61BB471C    ØCCØ - ØDØF        2818
   F2B7D6D4    ØD1Ø - ØD5F        27BC
   98DB53Ø6    ØD6Ø - ØDAF        2863
   27Ø7114A    ØDBØ - ØDFF        289Ø
   3ED3BE5Ø    ØEØØ - ØE4F        2433
   1CB4F2C8    ØE5Ø - ØE9F        29F2
   8BB24955    ØEAØ - ØEEF        27BC
   EØ87785D    ØEFØ - ØF3F        28AB
   7AE31E36    ØF4Ø - ØF8F        2141
   7ACA2FD3    ØF9Ø - ØFDF        283F
   D6C6FDA3    ØFEØ - 1Ø2F        22FB
   833BØ35Ø    1Ø3Ø - 1Ø7F        26BD
   27D1665B    1Ø8Ø - 1ØCF        2633
   FA8C62A2    1ØDØ - 111F        29CF
   534BØ28F    112Ø - 116F        28D7
   FB2CB7AA    117Ø - 11BF        25B8
   4F7ØFA9D    11CØ - 12ØF        2586
   2ØØ46226    121Ø - 125F        2289
   133236BF    126Ø - 12AF        27A3
   Ø8B7CC3A    12BØ - 12FF        28B8
   67F48B27    13ØØ - 134F        25A1
   BE5FB53C    135Ø - 139F        241E
   C93BA859    13AØ - 13D2        1B2F
   16ØBA42E = PROGRAM TOTAL =     ØAD3
```

LISTING 2: PFR

```
START: 900                LENGTH: AD3

BE |0900:A9 81 20 00 C3 20 69 0A
0E |0908:A9 00 8D 2A 12 8D 36 12
5E |0910:8D 38 12 8D 2B 12 8D 80
D6 |0918:02 8D 80 15 A9 02 8D 00
D9 |0920:15 20 19 0A 20 00 BF C5
2D |0928:0B 12 B0 6B 20 D4 0A B0
F0 |0930:66 20 85 0D 90 F6 D0 C8
56 |0938:C9 D1 F0 44 20 6F 0B 20
ED |0940:85 0A 20 8D 0B B0 50 20
CC |0948:12 0C B0 4B 20 58 FC 20
C6 |0950:1C 0A 2C 2B 12 30 08 A9
CE |0958:00 8D 36 12 8D 38 12 4E
08 |0960:2B 12 20 BE 0C 20 85 0D
FA |0968:90 F8 70 79 D0 74 10 10
84 |0970:C9 00 D0 C8 08 68 29 08
B3 |0978:F0 5A D8 AD 3B 12 D0 1A
4E |0980:20 C4 0A 20 62 0D 90 07
84 |0988:AD 2A 12 F0 55 D0 45 A9
78 |0990:16 20 CF 0A 4C D0 03 20
E5 |0998:40 0B 20 C4 0A C9 F0 90
BF |09A0:25 F0 0E C9 F1 F0 10 C9
70 |09A8:F2 F0 12 A0 BF A9 13 D0
FD |09B0:10 A0 81 A9 13 D0 0A A0
DB |09B8:98 A9 13 D0 04 A0 AB A9
E6 |09C0:13 20 FA 09 F0 06 20 8B
BB |09C8:BE 20 0C BE 20 0F 0A AD
DE |09D0:2A 12 F0 0E A9 FF 8D 2B
37 |09D8:12 CE 2A 12 20 69 0A 4C
5A |09E0:42 09 4C 00 09 AD 2A 12
05 |09E8:F0 F8 20 40 0B CE 2A 12
32 |09F0:F0 F0 A9 FF 8D 2C 12 4C
C0 |09F8:3F 09 84 FE 85 FF A0 00
37 |0A00:B1 FE F0 0A 20 ED FD C8
CF |0A08:D0 F6 E6 FF D0 F2 60 A0
71 |0A10:DC A9 12 20 FA 09 4C 0C
EE |0A18:FD A2 0E 2C A2 01 A9 13
8F |0A20:20 CF 0A A9 1B 20 ED FD
BB |0A28:A9 0F 20 ED FD A9 D3 A0
B8 |0A30:50 20 ED FD 88 D0 FA A9
78 |0A38:0E 20 ED FD A9 18 20 ED
B6 |0A40:FD 8E 7B 05 A0 60 A9 12
AD |0A48:20 FA 09 AD 2A 12 F0 07
7B |0A50:A0 94 A9 12 20 FA 09 20
A3 |0A58:8E FD 20 8E FD A9 1D 8D
9C |0A60:7B 05 A0 B3 A9 12 4C FA
92 |0A68:09 20 00 BF CC 22 12 60
8A |0A70:AD 37 12 CD 36 12 D0 0C
6C |0A78:A5 FA 85 FC A5 FB 85 FD
7A |0A80:A9 3F 85 32 60 AE 2A 12
ED |0A88:F0 0D A0 11 B1 FC 9D 00
59 |0A90:15 C8 B1 FC 9D 80 15 60
06 |0A98:A0 00 A2 0F B1 FA 29 0F
D8 |0AA0:F0 01 AA C8 B1 FA D0 03
3B |0AA8:88 10 08 09 80 20 ED FD
E3 |0AB0:CA D0 F0 C0 0F B0 08 A9
BE |0AB8:A0 20 ED FD C8 D0 F4 A9
C3 |0AC0:FF 85 32 60 48 A9 14 20
9D |0AC8:CF 0A 20 42 FC 68 60 85
3B |0AD0:25 4C 8E FD A9 99 20 ED
EC |0AD8:FD A0 C8 A9 12 20 FA 09
42 |0AE0:A9 FF 8D 37 12 A9 00 85
7E |0AE8:FA A9 21 85 FB A9 10 8D
03 |0AF0:2F 12 A0 00 B1 FA 29 0F
AE |0AF8:F0 2B EE 37 12 20 70 0A
A9 |0B00:B1 FA 4A 4A 4A 4A 29 07
B6 |0B08:09 B0 20 ED FD A9 AF 20
DF |0B10:ED FD B1 FA 0A A9 B1 69
E0 |0B18:00 20 ED FD 20 48 F9 20
57 |0B20:98 0A 20 8E FD 18 A5 FA
16 |0B28:69 10 85 FA 90 02 E6 FB
6F |0B30:CE 2F 12 D0 BD 38 AD 37
6E |0B38:12 30 04 8D 3A 12 18 60
EA |0B40:48 AE 80 02 BD 80 02 CE
24 |0B48:80 02 CA C9 2F D0 F5 68
66 |0B50:60 A2 00 BD 81 02 09 80
8A |0B58:20 ED FD E8 EC 80 02 90
12 |0B60:F2 A9 A0 20 ED FD AD 2A
A1 |0B68:12 20 DA FD 4C 8E FD A0
5E |0B70:00 B1 FC 29 0F 8D 2F 12

4E |0B78:A9 2F AE 80 02 E8 C8 9D
05 |0B80:80 02 B1 FC CE 2F 12 10
3B |0B88:F4 8E 80 02 60 20 00 BF
DA |0B90:C8 14 12 B0 0F AD 19 12
01 |0B98:8D 1B 12 8D 10 12 20 00
79 |0BA0:BF D1 0F 12 B0 65 AD 12
08 |0BA8:12 8D 1F 12 AD 11 12 8D
11 |0BB0:1E 12 A5 74 E9 20 CD 12
42 |0BB8:12 90 48 20 00 BF CA 1A
C6 |0BC0:12 90 04 C9 4C D0 44 20
35 |0BC8:69 0A AD 26 21 D0 34 AD
02 |0BD0:04 21 C9 F0 90 2B AD 30
3B |0BD8:BF 8D 25 12 AD 27 21 8D
86 |0BE0:40 12 AD 28 21 8D 41 12
CE |0BE8:AC 2A 21 8C 3F 12 AD 29
FB |0BF0:21 8D 3E 12 D0 01 88 98
D6 |0BF8:4A 4A 4A 4A A8 C8 8C 43
7A |0C00:12 18 60 A9 F0 D0 09 A9
09 |0C08:F1 D0 05 A0 00 8C 2A 12
0A |0C10:38 60 A2 00 8E 2D 12 AD
D5 |0C18:25 21 8D 32 12 D0 03 CE
CB |0C20:2D 12 A9 00 8D 31 12 AD
83 |0C28:24 21 8D 33 12 A9 04 85
10 |0C30:FA 18 A9 21 6D 31 12 85
69 |0C38:FB A5 FA 9D 00 16 A5 FB
75 |0C40:9D 00 17 A0 00 B1 FA A8
DB |0C48:C9 E0 B0 31 2C 2D 12 30
39 |0C50:1B A8 F0 29 A0 10 B1 FA
07 |0C58:C9 0F D0 06 9D 00 18 E8
BA |0C60:F0 A1 CE 32 12 D0 16 CE
C2 |0C68:2D 12 30 B6 A8 D0 0E A0
63 |0C70:01 B1 FA F0 31 A9 00 9D
B6 |0C78:00 18 E8 F0 86 18 A5 FA
46 |0C80:6D 23 21 85 FA 90 02 E6
DF |0C88:FB CE 33 12 D0 AB A9 00
99 |0C90:85 FA C6 FB A0 02 B1 FA
97 |0C98:C8 11 FA F0 09 18 AD 31
FF |0CA0:12 69 02 4C 24 0C E0 00
CE |0CA8:F0 11 CA 8E 3A 12 2C 2C
41 |0CB0:12 30 03 EE 2A 12 4E 2C
FE |0CB8:12 18 60 4C 07 0C A9 99
B3 |0CC0:20 ED FD 20 51 0B 20 04
3B |0CC8:0D A9 00 8D 34 12 A9 40
1B |0CD0:8D 39 12 AE 38 12 CA 8E
97 |0CD8:37 12 A9 01 20 CF 0A A9
DE |0CE0:10 8D 35 12 E8 EE 37 12
27 |0CE8:20 2B 0D EC 3A 12 D0 26
AF |0CF0:CE 39 12 F0 21 CE 35 12
5D |0CF8:D0 EA AD 34 12 69 14 8D
09 |0D00:34 12 D0 D6 AD 38 12 F0
41 |0D08:07 A0 E7 A9 12 20 FA 09
C4 |0D10:20 9C FC 4C 8E FD AD 34
62 |0D18:12 8D 7B 05 AD 39 12 D0
6D |0D20:07 A0 F5 A9 12 4C FA 09
77 |0D28:4C 9C FC 8A 48 BD 00 16
45 |0D30:85 FA BD 00 17 85 FB AD
22 |0D38:34 12 8D 7B 05 20 70 0A
62 |0D40:BD 00 18 A8 F0 02 A0 03
A7 |0D48:A2 03 B9 52 12 20 ED FD
CC |0D50:C8 CA D0 F6 A9 A0 20 ED
87 |0D58:FD 20 98 0A 20 8E FD 68
A0 |0D60:AA 60 A0 03 A9 13 20 FA
29 |0D68:09 F0 03 20 3A FF 20 0C
14 |0D70:FD 29 DF C9 CE F0 05 C9
74 |0D78:D9 D0 F0 18 08 20 ED FD
14 |0D80:28 60 20 3A FF 2C 61 C0
7B |0D88:10 05 2C C0 0D 70 01 B8
10 |0D90:AD 00 C0 10 F0 8D 10 C0
9F |0D98:AE 36 12 C9 E0 90 02 29
1C |0DA0:DF C9 8A F0 35 C9 8B F0
C4 |0DA8:18 C9 95 F0 46 C9 88 F0
C7 |0DB0:54 C9 D1 F0 0A C9 8D F0
B7 |0DB8:5E C9 9B D0 C5 A9 9B B8
55 |0DC0:60 AC 36 12 F0 0E CC 38
1F |0DC8:12 D0 03 CE 38 12 88 8C
77 |0DD0:36 12 18 60 AC 37 12 4C
DD |0DD8:CF 0D AC 36 12 CC 3A 12
E6 |0DE0:F0 0B CC 37 12 D0 03 EE
52 |0DE8:38 12 C8 D0 E2 AC 38 12
DC |0DF0:4C CF 0D 18 AD 36 12 A8
3D |0DF8:69 10 CD 37 12 F0 02 B0
6B |0E00:01 A8 4C CF 0D 38 AD 36
D6 |0E08:12 A8 E9 10 90 F4 48 ED
05 |0E10:38 12 68 90 ED B0 EA AD

2B |0E18:2A 12 F0 A4 70 A2 A9 83
F0 |0E20:48 A0 00 B1 FC F0 0D 28
D8 |0E28:60 A9 F3 2C A9 F2 8D 3B
CF |0E30:12 4C B5 0E 68 4E 2E 12
38 |0E38:20 BC 0E 20 92 0F B0 EC
56 |0E40:20 BF 0F B0 E9 20 F1 0F
DE |0E48:20 14 10 20 38 10 20 4C
28 |0E50:10 B0 D6 AE 3C 12 CA F0
E8 |0E58:21 20 8F 11 E0 0C F0 13
A0 |0E60:20 FC 10 70 C4 B0 C7 CA
39 |0E68:F0 10 20 3C 11 70 BA B0
23 |0E70:BD 90 07 20 AA 11 70 B1
65 |0E78:B0 B4 20 A9 10 B0 30 A0
98 |0E80:00 AD 3D 12 91 FC EE 25
93 |0E88:21 D0 03 EE 26 21 20 62
A0 |0E90:11 70 96 B0 99 20 B1 0F
38 |0E98:B0 94 20 57 0F B0 8F AD
BB |0EA0:3A 12 F0 0E AD 36 12 CD
94 |0EA8:3A 12 D0 03 CE 36 12 A9
EF |0EB0:83 2C A9 41 2C A9 8B 48
CA |0EB8:A9 00 28 60 20 C4 0A A5
8C |0EC0:FC 85 FA A5 FD 85 FB 20
3B |0EC8:98 0A 8A 49 0F 8D 3D 12
89 |0ED0:20 48 F9 A2 0D A0 10 B1
87 |0ED8:FC C9 0F F0 27 A2 03 A0
3E |0EE0:17 B1 FC F0 10 C9 02 90
5F |0EE8:1A D0 19 88 B1 FC 88 11
A8 |0EF0:FC F0 10 D0 0F A0 15 A9
18 |0EF8:00 D1 FC C8 A9 02 F1 FC
E8 |0F00:90 01 CA CA 8E 3C 12 8A
20 |0F08:0A 0A 0A 0A 0D 3D 12 8D
70 |0F10:3D 12 CA F0 18 CA F0 0F
1C |0F18:CA F0 06 A0 10 A9 13 D0
58 |0F20:10 A0 1C A9 13 D0 0A A0
7D |0F28:23 A9 13 D0 04 A0 2D A9
40 |0F30:13 20 FA 09 20 48 F9 A0
C0 |0F38:38 A9 13 20 FA 09 AD 7B
80 |0F40:05 8D 4A 12 A2 07 20 4A
99 |0F48:F9 A0 47 A9 13 20 FA 09
A0 |0F50:AD 7B 05 8D 4B 12 60 AE
F1 |0F58:2A 12 BD FF 14 8D 28 12
04 |0F60:BD 7F 15 8D 29 12 A9 00
62 |0F68:85 FA 8D 26 12 A9 21 85
FF |0F70:FB 8D 27 12 20 80 11 B0
53 |0F78:18 A0 02 B1 FA 8D 28 12
99 |0F80:C8 11 FA F0 0B B1 FA 8D
70 |0F88:29 12 A5 FB 69 02 D0 DF
25 |0F90:18 60 A9 00 8D 26 12 18
1C |0F98:AD 12 12 69 21 8D 27 12
FC |0FA0:8D 42 12 6D 43 12 6D 43
C2 |0FA8:12 C5 74 F0 02 B0 01 18
64 |0FB0:60 A9 00 8D 26 12 AD 42
E2 |0FB8:12 8D 27 12 A9 81 2C A9
30 |0FC0:80 8D D6 0F AE 43 12 AD
DB |0FC8:40 12 8D 28 12 AD 41 12
FB |0FD0:8D 29 12 20 00 BF 00 24
55 |0FD8:12 B0 15 CA F0 12 AD 27
C0 |0FE0:12 69 02 8D 27 12 EE 28
C1 |0FE8:12 D0 E8 EE 29 12 D0 E3
3C |0FF0:60 A9 00 8D 46 12 8D 47
66 |0FF8:12 8D 48 12 8D 49 12 60
DD |1000:F0 0A EE 46 12 D0 0D EE
9B |1008:47 12 D0 08 EE 48 12 D0
44 |1010:03 EE 49 12 AD 4A 12 8D
61 |1018:7B 05 AD 47 12 20 DA FD
E6 |1020:AD 46 12 20 DA FD AD 4B
D8 |1028:12 8D 7B 05 AD 49 12 20
C9 |1030:DA FD AD 48 12 4C DA FD
7F |1038:A0 11 B1 FC 8D 4D 12 8D
FB |1040:4E 12 C8 B1 FC 8D 4C 12
17 |1048:8D 4F 12 60 98 48 2C 2E
6D |1050:12 30 50 AD 3E 12 CD 4D
20 |1058:12 AD 3F 12 ED 4C 12 90
5B |1060:44 AD 4D 12 29 07 A8 B9
45 |1068:58 12 8D 44 12 49 FF 8D
58 |1070:45 12 4E 4C 12 6E 4D 12
AE |1078:4E 4C 12 6E 4D 12 4E 4C
DE |1080:12 6E 4D 12 AC 4D 12 18
09 |1088:AD 42 12 6D 4C 12 85 FB
E3 |1090:A9 00 85 FA B1 FA 48 2D
AF |1098:44 12 20 00 10 68 2D 45
17 |10A0:12 91 FA 18 24 38 68 A8
EB |10A8:60 20 8E FD A2 D9 AD 48
BE |10B0:12 0D 49 12 F0 0F 20 80
```

```
7E | 10B8:FE A0 56 A9 13 20 FA 09
0E | 10C0:20 84 FE A2 CE 20 8E FD
F2 | 10C8:A0 67 A9 13 20 FA 09 8A
84 | 10D0:20 ED FD A9 88 20 ED FD
F9 | 10D8:4C DE 10 20 3A FF 20 0C
9A | 10E0:FD C9 E0 90 02 29 DF C9
DE | 10E8:D9 F0 0D C9 CE F0 0A C9
28 | 10F0:8D D0 E8 8A C9 CE F0 01
33 | 10F8:18 4C 7C 0D 8C 28 12 8D
5C | 1100:29 12 20 88 11 B0 2F A0
6E | 1108:00 B1 EE 8D 4C 12 E6 EF
17 | 1110:11 EE F0 15 B1 EE 8D 4D
F6 | 1118:12 C6 EF 91 EE AD 4C 12
0E | 1120:E6 EF 91 EE 20 4C 10 B0
53 | 1128:0F C6 EF C8 D0 DB 2C 2E
B6 | 1130:12 10 03 20 80 11 B8 60
84 | 1138:2C C0 0D 60 A0 00 A9 1B
A6 | 1140:8C 26 12 8D 27 12 84 EE
C4 | 1148:85 EF A2 00 8E 30 12 BD
8A | 1150:00 19 A8 BD 00 1A 20 FC
CA | 1158:10 B0 06 AE 30 12 E8 10
E9 | 1160:EB 60 18 B8 AE 3C 12 CA
AE | 1168:F0 F7 CA F0 13 20 8F 11
12 | 1170:20 80 11 B0 EC CA D0 E9
27 | 1178:A9 FF 8D 2E 12 4C 3C 11
8A | 1180:20 00 BF 81 24 12 B8 60
85 | 1188:20 00 BF 80 24 12 60 A0
03 | 1190:00 A9 19 8C 26 12 8D 27
7F | 1198:12 84 EE 85 EF AC 4E 12
F8 | 11A0:AD 4F 12 8C 28 12 8D 29
8B | 11A8:12 60 20 88 11 B0 56 A9
84 | 11B0:01 8D 50 12 A0 00 A9 1B
06 | 11B8:8C 26 12 8D 27 12 AC 02
2D | 11C0:19 98 0D 03 19 F0 2A AD
E9 | 11C8:03 19 8C 4D 12 8C 28 12
34 | 11D0:8D 4C 12 8D 29 12 20 4C
0A | 11D8:10 B0 2C 20 88 11 B0 25
F3 | 11E0:EE 50 12 AC 02 1B 98 0D
E2 | 11E8:03 1B F0 05 AD 03 1B 90
5E | 11F0:D9 18 AD 3D 12 69 10 8D
65 | 11F8:04 19 A0 13 B1 FC CD 50
BB | 1200:12 38 D0 03 18 B8 60 2C
BF | 1208:C0 0D 60 02 00 00 21 02
55 | 1210:00 00 00 00 03 80 02 00
B9 | 1218:1D 00 04 00 00 21 00 00
D2 | 1220:00 00 01 00 03 00 00 00
52 | 1228:00 00 00 00 00 00 00 00
3D | 1230:00 00 00 00 00 00 00 00
1A | 1238:00 00 00 00 00 00 00 00
B7 | 1240:00 00 00 00 00 00 00 00
A6 | 1248:00 00 00 00 00 00 00 00
49 | 1250:00 00 C4 C5 CC C4 C9 D2
94 | 1258:80 40 20 10 08 04 02 01
DA | 1260:C5 D3 C3 AD F2 E5 F3 F4
9F | 1268:E1 F2 F4 A0 A0 A0 D1 AD
02 | 1270:F1 F5 E9 F4 A0 A0 A0 C1
22 | 1278:D2 D2 CF D7 D3 AD F3 E5
19 | 1280:EC E5 E3 F4 A0 A0 A0 D2
AF | 1288:D4 CE AD EE E5 F8 F4 AE
E9 | 1290:E4 E9 F2 00 AF F5 EE E4
FC | 1298:E5 EC E5 F4 E5 A0 A0 A0
BA | 12A0:1B 0F C1 0E 18 D2 D4 CE
97 | 12A8:AD F0 F2 E9 EF F2 AE E4
25 | 12B0:E9 F2 00 D0 F2 EF C4 CF
87 | 12B8:D3 A0 C6 C9 CC C5 A0 D2
08 | 12C0:C5 C3 CF D6 C5 D2 D9 00
DE | 12C8:D3 AF C4 A0 A0 A0 D6 CF
E1 | 12D0:CC D5 CD C5 A0 CE C1 CD
F6 | 12D8:C5 8D 8D 00 C8 E9 F4 A0
60 | 12E0:E1 A0 EB E5 F9 A0 00 1B
CA | 12E8:0F D2 1C D2 1C D2 1C D2
49 | 12F0:1C D2 0E 18 00 1B 0F D1
81 | 12F8:1C D1 1C D1 1C D1 1C D1
1A | 1300:0E 18 00 D1 F5 E9 F4 A0
34 | 1308:A8 D9 AF CE A9 BF A0 00
2B | 1310:A8 C4 E9 F2 E5 E3 F4 EF
79 | 1318:F2 F9 A9 00 A8 D4 F2 E5
28 | 1320:E5 A9 00 A8 D3 E1 F0 EC
42 | 1328:E9 EE E7 A9 00 A8 D3 E5
5F | 1330:E5 E4 EC E9 EE E7 A9 00
33 | 1338:C6 F2 E5 E5 A0 C2 EC EF
DA | 1340:E3 EB F3 BA A0 A4 00 D5
53 | 1348:F3 E5 E4 A0 C2 EC EF E3
F4 | 1350:EB F3 BA A0 A4 00 C2 CC
4B | 1358:CF C3 CB A8 D3 A9 A0 C9
F3 | 1360:CE A0 D5 D3 C5 87 00 D5
D6 | 1368:EE E4 E5 EC E5 F4 E5 A0
C8 | 1370:F4 E8 E5 A0 E6 E9 EC E5
11 | 1378:A0 A8 D9 AF CE A9 BF A0
6C | 1380:00 8D C4 C9 D2 C5 C3 D4
C6 | 1388:CF D2 D9 A0 D4 CF CF A0
2B | 1390:CC C1 D2 C7 C5 87 8D 00
86 | 1398:8D C4 C9 D2 C5 C3 D4 CF
B8 | 13A0:D2 D9 A0 C5 CD D0 D4 D9
C6 | 13A8:87 8D 00 8D C2 C9 D4 CD
34 | 13B0:C1 D0 A0 D4 CF CF A0 CC
89 | 13B8:C1 D2 C7 C5 87 8D 00 8D
C8 | 13C0:C2 C1 C4 A0 C2 CC CF C3
DF | 13C8:CB A0 CE D5 CD C2 C5 D2
74 | 13D0:87 8D 00

TOTAL: B299
```

END OF LISTING 2

```
             KEY PERFECT 5.0
                 RUN ON
              PRACTICE.PFR
==========================================
 CODE-5.0    LINE# - LINE#     CODE-4.0
 --------    -------------     --------
 0C8A5543        1 -    20         8502
 DF3EDDE8       30 -   120         3DDA
 5B750FD5      130 -   220         7676
 77FCB294      230 -   320         35B6
 4DCF3FDA      330 -   420         3D05
 741FD429      430 -   520         4326
 B289EBC3 = PROGRAM TOTAL =        0443
```

LISTING 3: PRACTICE.PFR

```
CB | 1  REM *************************
47 | 2  REM * PRACTICE.PFR          *
47 | 3  REM * PRODOS FILE RECOVERY  *
3F | 4  REM * BY SANDY MOSSBERG     *
3B | 5  REM * COPYRIGHT (C) 1988    *
BF | 6  REM * MICROSPARC, INC       *
BF | 7  REM * CONCORD, MA 01742     *
9D | 8  REM *************************
37 | 10  REM Volume name of disk is in line 20. Chan
   |     ge this name to the name of your test disk.
56 | 20 V$ = "/RAM"
0B | 30 D$ =  CHR$ (4)
78 | 40  PRINT D$"PREFIX"
30 | 50  INPUT P$
9A | 60  PRINT D$"PREFIX"V$
45 | 70  REM  *
E9 | 80  FOR I = 1 TO 4
97 | 90  PRINT D$"CREATE DIR"I
DD | 100  NEXT I
D7 | 110  REM  *
79 | 120  PRINT D$"BSAVE BIGTREE,A$2000,L1,B$FFFFFF"
58 | 130  PRINT D$"BSAVE TINYTREE,A$2000,L1,B$20000"
83 | 140  PRINT D$"BSAVE BIGSAP,A$2000,L1,B$1FFFF"
BF | 150  PRINT D$"BSAVE TINYSAP,A$2000,L1,B$200"
4B | 160  PRINT D$"BSAVE BIGSEED,A$2000,L1,B$1FF"
F6 | 170  PRINT D$"BSAVE TINYSEED,A$2000,L0"
7E | 180  PRINT D$"DELETE BIGTREE"
1D | 190  REM  *
24 | 200  FOR I = 1 TO 15
2D | 210  PRINT D$"CREATE DIR1/DIRECTORY.FIL"I
CA | 220  NEXT I
5E | 230  REM  *
83 | 240  FOR J = 1 TO 10
5C | 250  PRINT D$"SAVE DIR1/FILE"J
46 | 260  NEXT J
D2 | 270  REM  *
84 | 280  PRINT D$"SAVE DIR2/FILE1"
DC | 290  REM  *
84 | 300  FOR J = 1 TO 10
CD | 310  PRINT D$"SAVE DIR4/FILE"J
E4 | 320  NEXT J
60 | 330  REM  *
04 | 340  FOR J = 1 TO 10
9B | 350  PRINT D$"DELETE DIR4/FILE"J
61 | 360  NEXT J
77 | 370  PRINT D$"DELETE DIR4"
9F | 380  REM  *
AD | 390  PRINT D$"DELETE DIR2/FILE1"
2B | 400  REM  *
F6 | 410  FOR J = 1 TO 10
3A | 420  PRINT D$"DELETE DIR1/FILE"J
52 | 430  NEXT J
B7 | 440  REM  *
4E | 450  PRINT D$"DELETE TINYTREE"
3B | 460  PRINT D$"DELETE BIGSAP"
38 | 470  PRINT D$"DELETE TINYSAP"
04 | 480  PRINT D$"DELETE BIGSEED"
E3 | 490  PRINT D$"DELETE TINYSEED"
11 | 500  REM  *
27 | 510  PRINT D$"PREFIX"P$
3A | 520  PRINT  CHR$ (7)

TOTAL: AED1
```

END OF LISTING 3