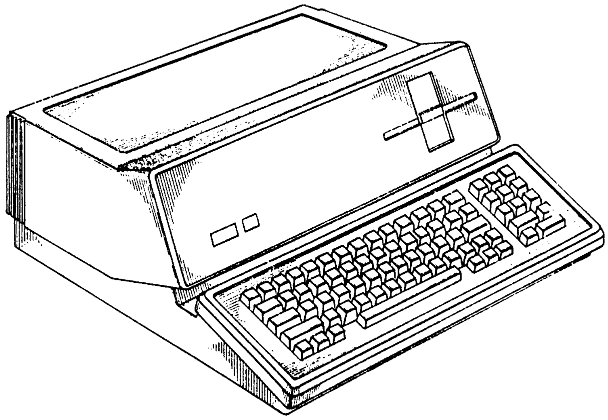
Apple /// Computer Technical Information

Apple /// SOS Operating System Source Code Listing

Version 1.3
Apple Computer -- 1982



Assembly Listing Produced by
Paul R. Santa-Maria
August 2006

Apple III SOS 1.3 Source Code Assembly Commentary

Apple3SOS13List.pdf is where I started from.

The six disk images contain the source code. The name contains the slot/drive where it should be mounted; for example, SOS13-42.DSK should be in slot 4 drive 2.

I put a DOS 3.3 boot disk in slot 5, drive 1, and the DOS Tool Kit assembler disk in slot 5 drive 2. This means eight 140KB drives have to be connected to the Apple II. This is hard for a real Apple II, but trivial in an Apple II emulator.

SOS13-62.DSK has no source code; it got the OBJ files.

I modified the source code as little as possible. There was some bit rot on SOS13-41.DSK. Search for BITROT in the listing to see where I had to modify the source so it would assemble. Compare it to the PDF file to see the differences. I changed slot/drive assignments in the source code to match the disk assignments here. I deleted volume numbers.

SOS13-D.LST was created using DOS EDASM, while SOS13-P.LST used ProDOS EDASM. SOS13-D.LST ran out of memory while assembling the last file, PRINT. SOS13-P.LST shows the full assembly. SOS13-D.LST shows twelve warnings in INIT but it assembles okay. The warnings are "EXTRN USED AS ZXTRN IN LINE" and are about the label FCBZPP. SOS13-P.LST shows four warnings and fourteen errors in INIT. It seems that the rules have changed between the DOS and ProDOS versions I used. That is why I include listings for both versions of EDASM.

I never compared the output files from the assembler with the SOS.KERNEL file, but I have included SOSKERNEL.BIN. if you want to compare.

--- Paul R. Santa-Maria
--- Temperance, Michigan USA
--- August 2006

```
SOURCE   FILE #01 =>SOSLDR.SRC
SOURCE   FILE #02 =>SOSLDR.A.SRC
SOURCE   FILE #03 =>SOSLDR.B.SRC
SOURCE   FILE #04 =>SOSLDR.C.SRC
SOURCE   FILE #05 =>SOSLDR.D.SRC
SOURCE   FILE #06 =>SOSLDR.E.SRC
SOURCE   FILE #07 =>SOSLDR.F.SRC

***** UNDEFINED IDENTIFIER ERROR IN LINE  571
```

```
0000:                                 REL
1E00:        1E00                     ORG   $1E00
1E00:        1E00      ZZORG          EQU   *
1E00:                                 MSB   OFF
1E00:
****************************************************************************************************
1E00:                  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
1E00:                  *                    ALL RIGHTS RESERVED
1E00:
****************************************************************************************************
1E00:                  *
1E00:                  *        SOS KERNEL LOAD & MEMORY POINTS
1E00:                  *
1E00:                  *  MODULE      START  END   I/O  ROM  SOS BLOAD    SIZE
1E00:                  *------------------------------------------------------
1E00:                  *  SOSLDR      1E00 - 28F7            2000         0CF8
1E00:                  *  INIT        28F8 - 2AA9            2AF8        [01B2]
1E00:                  *  SYSGLOB     18FC - 1A03            2CF8
1E00:                  *
1E00:                  *  BFM.INIT2 + BITMAPS
1E00:                  *              B800 - BBFF            2E00         03FF
1E00:                  *  BFM         BC00 - DE62            3200         2263
1E00:                  *  <PATCH>     DE63 - DE6A            5463         0008
1E00:                  *
1E00:                  *  OPRMSG      DE6B - E48A   X        546B         015A
1E00:                  *  IPL         DFC5 - E48F   X    X   55C5         04CB
1E00:                  *  UMGR        E490 - E89D   X    X   5A8B         040E
1E00:                  *
1E00:                  *  DISK3       E899 - EE03   X    X   5E99         056B
1E00:                  *  SYSERR      EE04 - EED8   X        64D9         00D5
1E00:                  *  DEVMGR      EED9 - F05D            64D9         0185
1E00:                  *
1E00:                  *  SCMGR       F05E - F2F3            665E         0296
1E00:                  *  FMGR        F2F4 - F354            68F4         0061
1E00:                  *  CFMGGR      F355 - F551            6955         01FD
1E00:                  *
1E00:                  *  BUFMGR      F552 - F86D            6B52         031C
1E00:                  *  MEMMGR      F86E - FFBE            6E6E         0751
1E00:                  *  <END>       FFBE
1E00:                  *
1E00:
****************************************************************************************************
1E00:                  * SOS LOADER  (VERSION = 1.1O   )
1E00:                  *             (DATE    = 8/04/81)
1E00:                  *
1E00:                  * SOURCE FILES:  SOSLDR.SRC,   SOSLDR.A.SRC, SOSLDR.B.SRC, SOSLDR.C.SRC,
1E00:                  *                              SOSLDR.D.SRC, SOSLDR.E.SRC, SOSLDR.F.SRC
1E00:                  *
1E00:                  * FUNCTION:
1E00:                  *    MOVES AND INITIALIZES SOS KERNEL, READS INTERPRETER FROM DISK, READS CHARACTER SET TABLE,
1E00:                  *    KEYBOARD TABLE AND DRIVERS FROM DISK, INITIALIZES ALL DRIVERS AND THEN JUMPS TO INTERPRETER
1E00:                  *    ENTRY POINT.
1E00:                  *
1E00:                  * CALLED BY:
1E00:                  *    SOSBOOT 7.0 WITH KERNEL FILE LOADED AT $I:1E00.9FFF(MAX)
1E00:                  *    WHERE: $I=INTERPRETER BANK (HIGHEST BANK IN SYSTEM)
1E00:                  *
1E00:                  * CALLS:
1E00:                  *    INTERPRETER ENTRY POINT (FIRST BYTE OF INTERPRETER CODE)
```

```
1E00:                  *
1E00:                  * DOCUMENTS:
1E00:                  *    SOS ERS APPENDICES - XX/XX/81
1E00:                  *    APPLE III I/O SYSTEM PROGRAMMERS GUIDE - DEC-15-80
1E00:                  *
1E00:                  * CONSTRAINTS:
1E00:                  *    INTERPRETER FILE:  READ INTO BANK 0 BEGINNING AT $80:LDREND+$400(=BUFSIZE).
1E00:                  *                       INTERPRETER CODE DOES NOT CONTAIN RELOCATION INFORMATION.
1E00:                  *                       MAX = 38K  ($I:2000..B7FF)
1E00:                  *                       MIN = .25K ($I:B700..B7FF)
1E00:                  *
1E00:                  *    DRIVER FILE:  READ INTO BANK 0 BEGINNING AT $80:LDREND+$400(=BUFSIZE).
1E00:                  *                  DRIVER MODULES ARE RELOCATED AND MOVED TO THE HIGHEST AVAILABLE 32K BANK USING
1E00:                  *                  A "FIRST FIT" ALGORITHM.  MODULES ARE REMOVED FROM THE FILE BEGINNING AT THE
BACK
1E00:                  *                  AND WORKING TOWARD THE FRONT.  A DRIVER MODULE CANNOT SPAN A BANK BOUNDARY.
1E00:                  *
1E00:                  *                  DRIVER FILE:  MAX = 60K  (APPROX)       DRIVER MODULE:  MAX = 32K-1
1E00:                  *                                MIN = .25K                                MIN < .25K
1E00:                  *
1E00:                  *
1E00:                  * DATA STRUCTURES:
1E00:                  *    SOS.KERNEL FILE FORMAT
1E00:                  *    SOS.INTERP FILE FORMAT
1E00:                  *    SOS.DRIVER FILE FORMAT
1E00:                  *
1E00:
****************************************************************************************************
```

```
1E00:
****************************************************************************************************
1E00:                  *
1E00:                  * NOTATION:
1E00:                  *
1E00:                  *    A, X, Y           ::= 6502 REGISTERS
1E00:                  *
1E00:                  *    C, OV             ::= CARRY, OVERFLOW FLAGS IN 6502 STATUS (P) REGISTER
1E00:                  *    E, Z, B           ::= ENVIRONMENT, ZERO PAGE, BANK REGISTERS (SYSTEM CONTROL REGISTERS)
1E00:                  *
1E00:                  *    (1.I.S.R:W.P.R.R) ::= ENVIRONMENT REGISTER FLAGS.  FROM LEFT TO RIGHT BITS 7..0
1E00:                  *                          (1MHZ, I/O ENABLE, SCREEN ENABLE, RESET ENABLE,
1E00:                  *                           WRITE PROTECT, PRIMARY STACK, ROM1, ROM ENABLE)
1E00:                  *
1E00:                  *    "POSITIVE LOGIC"  ::= ALL LOGIC USED IS POSITIVE LOGIC.  FOR EXAMPLE, C="NO DRIVERS LEFT"
1E00:                  *                          INDICATES THAT NO DRIVERS ARE LEFT WHEN CARRY = SET, AND THAT ONE OR
1E00:                  *                          MORE DRIVERS ARE LEFT WHEN CARRY = CLEAR.
1E00:                  *
1E00:                  *    TRUE,FALSE        ::= TRUE = SET = ON, WHILE FALSE = CLEAR = OFF.
1E00:                  *
1E00:
****************************************************************************************************
1E00:                  *
1E00:                  * ABBREVIATIONS:
1E00:                  *
1E00:                  *    DIB               ::= DEVICE INFORMATION BLOCK.  DEFINES A UNIQUE DEVICE THAT CAN BE LINKED
1E00:                  *                          INTO THE SYSTEM DEVICE TABLE.  EACH DRIVER MODULE CONTAINS ONE OR MORE
1E00:                  *                          DIBS (DEVICES) EACH OF WHICH CAN BE "ACTIVE" OR "INACTIVE".
1E00:                  *
1E00:                  *    ADIB              ::= "ACTIVE DIB"
1E00:                  *
1E00:                  *   <VARNAME>.P        ::=  POINTER.  A 3 BYTE ZERO PAGE POINTER.  DON'T FORGET THE X BYTE!
1E00:                  *
1E00:                  *    SDT               ::= SYSTEM DEVICE TABLE.  CONTAINS THE ENTRY POINT AND DIB ADDRESS OF EACH
1E00:                  *                          DEVICE CONFIGURED INTO THE SYSTEM, (USED BY THE DEVICE MANAGER).
1E00:
****************************************************************************************************
1E00:                  *
1E00:                             CHN   SOSLDR.A.SRC
```

```
1E00:
**************************************************************************************************************
1E00:                  *
1E00:                  *   $1E00 +---------------+
1E00:                  *         !    SOSLDR     !<-ENTRY        SOS MEMORY MAP
1E00:                  *   $1FFF +---------------+  -----       (128K APPLE ///)
1E00:                  *
1E00:                  *              BANK 0               BANK 1               BANK 2
1E00:                  *   $2000 +---------------+    +---------------+    +---------------+
1E00:                  *         !               !    !               !    !               !
1E00:                  *         !               !    !               !    !    SOSLDR     !
1E00:                  *         !               !    !               !    !      &        !
1E00:                  *         !               !    !               !    !  INIT MODULE  !
1E00:                  *         !               !    !               !    !               !
1E00:                  *         !               !    !               !    ! - - - - - - - !
1E00:                  *         !               !    !               !    !    GLOBALS    !
1E00:                  *         !               !    !               !    ! - - - - - - - !
1E00:                  *         !               !    !               !    !               !
1E00:                  *         !               !    !               !    !               !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !  KERNEL !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !-- EOF --!
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *         !               !    !               !    !               !         !
1E00:                  *   $9FFF +---------------+    +---------------+    +---------------+
1E00:                  *
1E00:                  *
1E00:                  *   $A000 +---------------+
1E00:                  *     .   !    SOSBOOT    !
1E00:                  *     .   +---------------+
1E00:                  *
1E00:                  *
1E00:                  *  FIGURE 1.  SOS KERNEL FILE READ INTO $2:1E00..9FFF BY SOS BOOT IN BLOCKS 0,1.
1E00:                  *             SOS LOADER BEGINS EXECUTION AT THIS POINT.
1E00:                  *
1E00:                  *
1E00:                  *
1E00:                  *
1E00:
**************************************************************************************************************
```

```
1E00:
*********************************************************************************************************************
1E00:                   *
1E00:                   *   $1E00 +---------------+
1E00:                   *         !    SOSLDR     !          SOS MEMORY MAP
1E00:                   *   $1FFF +---------------+          (128K APPLE ///)
1E00:                   *
1E00:                   *               BANK 0               BANK 1               BANK 2
1E00:                   *   $2000 +---------------+    +---------------+    +---------------+
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !    SOSLDR     !    !               !    !               !
1E00:                   *         !      &        !    !               !    !               !
1E00:                   *         !  INIT MODULE  !    !               !    !               !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *  LDREND ! - - - - - - - !    !               !    !               !
1E00:                   *         !  FILE BUFFER  !    !               !    !               !
1E00:                   *         ! - - - - - - - !    !               !    !               !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !  INTERPRETER  !    !  INTERPRETER  !    !               !
1E00:                   *         !     FILE      !    !     FILE      !    !               !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !               !    !               !         !
1E00:                   *         !               !    !- - - EOF - - -!    !               !
1E00:                   *   $9FFF +---------------+    +---------------+    +---------------+
1E00:                   *
1E00:                   *
1E00:                   *
1E00:                   *
1E00:                   *  FIGURE 2.  SOS INTERPRETER FILE READ INTO BANKS 0 AND 1
1E00:                   *             USING EXTENDED ADDRESSING (X=$80).
1E00:                   *
1E00:                   *
1E00:                   *
1E00:                   *
1E00:                   *
1E00:
*********************************************************************************************************************
1E00:                   *
1E00:                              CHN   SOSLDR.B.SRC
```

```
1E00:                  *
1E00:                  ;*******************************************************************************
1E00:                  ;
1E00:                  ;   $1E00 +---------------+
1E00:                  ;         !    SOSLDR     !       SOS MEMORY MAP
1E00:                  ;   $1FFF +---------------+       (128K APPLE ///)
1E00:                  ;
1E00:                  ;              BANK 0               BANK 1              BANK 2
1E00:                  ;   $2000 +---------------+    +---------------+    +---------------+
1E00:                  ;         !               !    !               !    !               !
1E00:                  ;         !    SOSLDR     !    !               !    !               !
1E00:                  ;         !      &        !    !               !    !               !
1E00:                  ;         !  INIT MODULE  !    !               !    !               !
1E00:                  ;         !               !    !               !    !               !
1E00:                  ;         ! - - - - - - - !    !               !    !               !
1E00:                  ;         !  FILE BUFFER  !    !               !    !               !
1E00:                  ;         ! - - - - - - - !    !               !    !               !
1E00:                  ;         !               !    !               !    !               !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               ! - - - - - - !
1E00:                  ;         !    DRIVER     !    !    DRIVER     !    !               !
1E00:                  ;         !     FILE      !    !     FILE      !    !               !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               ! INTERPRETER !
1E00:                  ;         !               !    !               !    !               !     CODE    !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !               !    !               !             !
1E00:                  ;         !               !    !- - - EOF - - -!    !               !
1E00:                  ;   $9FFF +---------------+    +---------------+    +---------------+
1E00:                  ;
1E00:                  ;
1E00:                  ;
1E00:                  ;
1E00:                  ;               FIGURE 3.  SOS DRIVER FILE READ INTO BANKS 0 AND 1
1E00:                  ;                          USING EXTENDED ADDRESSING (X=$80).
1E00:                  ;
1E00:                  ;
1E00:                  ;
1E00:                  ;
1E00:                  ;*******************************************************************************
1E00:                  ;!BITROT
```

```
1E00:    ;!BITROT
1E00:    ;         !               !    !               !    !        !               !
1E00:    ;   $9FFF +---------------+    +---------------+    !        +---------------+
1E00:    ;                                                   !
1E00:    ;                                                   !
1E00:    ;                                                   !   (SYSTEM DEVICE TABLE)
1E00:    ;                                                   !
1E00:    ;  FIGURE 4. SOS LOADER FINISHED.  JUMP TO        DIB   ADR   BANK  UNIT
1E00:    ;            FIRST BYTE OF INTERPRETER'S CODE.   !-----!-----!-----!-----!
1E00:    ;                                                !     !     !     !     !
1E00:    ;                                                !     !     !     !     !
1E00:    ;                                                !     !     !     !     !
1E00:    ;                                                !     !     !     !     !
1E00:    ;                                                !-----!-----!-----!-----!
1E00:    ;
1E00:    ;
1E00:    ;
1E00:    ;*******************************************************************************
1E00:               CHN   SOSLDR.C.SRC     ;BITROT
```

```
1E00:
****************************************************************************************************
1E00:                  *
1E00:                  * SUBROUTINES:
1E00:                  *
1E00:                  * SOSLDR                "MAIN PROGRAM"
1E00:                  *
1E00:                  *    SOSLDR1            "PROCESSES KERNEL/INTERPRETER/DRIVER FILES"
1E00:                  *
1E00:                  * (1)   MOVE            "MOVES SRC.P..SRC.P+CNT-1 TO DST.P..DST.P+CNT-1"
1E00:                  *
1E00:                  *       INIT.KRNL       "CALLS KERNEL INITIALIZATION MODULES"
1E00:                  *
1E00:                  *       WELCOME         "PRINTS WELCOME MESSAGE ("APPLE ///", VERSION, DATE/TIME, COPYRIGHT)
1E00:                  *
1E00:                  *       ADVANCE         "ADVANCES WRK.PTR TO NEXT INTERP/KERNEL MODULE.  INITS SRC.P, DST.P, CNT
FOR MOVE"
1E00:                  *
1E00:                  *       REVERSE         "REVERSES TITLE/CODE/RELOC COUNTS TO ALLOW DRIVER FILE TO BE PROCESSED FM
BACK TO FRONT"
1E00:                  *
1E00:                  *       DADVANCE        "ADVANCES WORK.P TO NEXT DRIVER MODULE.  INITS SRC.P, CNT, REL.P FOR MOVE"
1E00:                  *
1E00:                  *          DADD         "ADVANCES WORK.P TO NEXT DRIVER FIELD"
1E00:                  *
1E00:                  *       FLAGS           "PROCESSES "INACTIVE" & "PAGE ALIGN" FLAGS IN DRIVER MODULE'S DIBS"
1E00:                  *
1E00:                  *          NEXT.DIB     "ADVANCES TO NEXT DIB IN DRIVER MODULE"
1E00:                  *
1E00:                  *       GETMEM          "COMPUTES DESTINATION BASE ADDRESS FOR NEXT DRIVER MODULE"
1E00:                  *
1E00:                  *          NEWDST       "COMPUTES DESTINATION BASE ADDRESS, ALIGNING ON PAGE BOUNDARY IF
REQUESTED"
1E00:                  *
1E00:                  *          BUILD.DSEG   "COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW
SEGMENT"
1E00:                  *
1E00:                  *       RELOC           "RELOCATES DRIVER MODULE'S CODE FIELD USING RELOCATION FIELD"
1E00:                  *
1E00:                  * (1)   LINK            "LINKS FIRST DIB TO PREVIOUS DRIVER'S LAST "ACTIVE" DIB, AND ADDS SDT
ENTRY"
1E00:                  *
1E00:                  *          SET.DRIVES   "INITIALIZES DIB LINKS IN KERNEL'S FLOPPY DRIVER"
1E00:                  *
1E00:                  * (1)      ALLOC.DEV    "ADDS A NEW ENTRY TO THE DEVICE MANAGER'S SYSTEM DEVICE TABLE (SDT)"
1E00:                  *
1E00:                  *       ALLOC.SEG       "ALLOCATES SEGMENTS FOR KERNEL, INTERPRETER AND SYSTEM WORK AREA"
1E00:                  *
1E00:                  *          RSEG         "CALLS MEMORY MANAGER TO ALLOCATE SEGMENTS FOR THE KERNEL AND INTERPRTER"
1E00:                  *
1E00:                  *       ALLOC.DSEG      "ALLOCATES SEGMENTS FOR DRIVER MODULES"
1E00:                  *
1E00:                  *    ERROR              "DISPLAYS ERROR MESSAGE, SOUNDS BELL AND LOOPS UNTIL CONTROL/RESET
PRESSED"
1E00:                  *
1E00:                  * (1) - INDICATES THAT THE ROUTINE PERFORMS BANK SWITCHING AND MUST(!) BE OUTSIDE THE 32K RAM
BANKS.
1E00:
****************************************************************************************************
```

```
1E00:
****************************************************************************************************
1E00:                  *
1E00:                  * SOS.KERNEL FILE FORMAT
1E00:                  *
1E00:                  *  (8)   LABEL                    <---+
1E00:                  *          = "SOS KRNL"               !
1E00:                  *                                     !
1E00:                  *  (2)   HEADER COUNT                 !
1E00:                  *        HEADER                       !
1E00:                  *          = # OF FLOPPY DRIVES       !    CONTAINED IN THIS LISTING
1E00:                  *          = INTERPRETER PATHNAME     !
1E00:                  *          = DRIVER PATHNAME          !
1E00:                  *                                     !
1E00:                  *  (4)   ADR & COUNT                  !
1E00:                  *        SOSLDR CODE              <---+
1E00:                  *
1E00:                  *  (4)   ADR & COUNT
1E00:                  *        GLOBALS
1E00:                  *
1E00:                  *  (4)   ADR & COUNT
1E00:                  *        KERNEL CODE
1E00:                  *
1E00:
****************************************************************************************************
1E00:                  *
1E00:                  * SOS.INTERP FILE FORMAT
1E00:                  *
1E00:                  *  (8)   LABEL
1E00:                  *           = "SOS NTRP"
1E00:                  *
1E00:                  *  (2)   HEADER COUNT
1E00:                  *
1E00:                  *  (4)   ADR & COUNT
1E00:                  *        INTERPRETER CODE
1E00:                  *
1E00:
****************************************************************************************************
1E00:                  *
1E00:                  * SOS.DRIVER FILE FORMAT
1E00:                  *
1E00:                  *  (8)   LABEL
1E00:                  *           = "SOS DRVR"
1E00:                  *
1E00:                  *  (2)   HEADER COUNT
1E00:                  *           = # OF FLOPPY DRIVES
1E00:                  *           = CHARACTER SET TABLE
1E00:                  *           = KEYBOARD TABLE
1E00:                  *        ...
1E00:                  *                                                               +---------------------------------
------+
1E00:                  *  (2)   DM #N TITLE COUNT        <---+                         !       RELOCATION FIELD FORMAT
!
1E00:                  *              TITLE FIELD            !                         !       -----------------------
!
1E00:                  *  (2)   DM #N CODE  COUNT            !    DRIVER MODULE #N     ! CONSISTS OF A LIST OF 2 BYTE
POINTERS !
1E00:                  *              CODE  FIELD            !                         ! WHICH POINT TO THE LOW BYTE OF A
TWO  !
1E00:                  *  (2)   DM #N RELOC COUNT            !                         ! BYTE QUANTITY TO BE RELOCATED.
!
1E00:                  *              RELOC FIELD        <---+                         +---------------------------------
------+
1E00:                  *        ...
1E00:                  *
1E00:                  *        $FFFF = THE END
```

```
1E00:                  *
1E00:
****************************************************************************************************
```

```
1E00:
****************************************************************************************************
1E00:                   *
1E00:                   * SOSLDR - EXTERNAL DECLARATIONS
1E00:                   *
1E00:
****************************************************************************************************
1E00:        0000                  EXTRN SYSBANK
1E00:        0000                  EXTRN MEMSIZE
1E00:        0000                  EXTRN SCRNMODE
1E00:        0000                  EXTRN SOSVER
1E00:        0000                  EXTRN SOSVERL
1E00:                   *
1E00:        0000                  EXTRN INT.INIT         ; (IPL) INTERRUPT INIT
1E00:        0000                  EXTRN EVQ.INIT         ; (IPL) EVENT QUEUE INIT
1E00:        0000                  EXTRN DMGR.INIT        ; DEVICE MANAGER INIT
1E00:        0000                  EXTRN MAX.DNUM         ;        "
1E00:        0000                  EXTRN SDT.SIZE
1E00:        0000                  EXTRN SDT.DIBL
1E00:        0000                  EXTRN SDT.DIBH
1E00:        0000                  EXTRN SDT.ADRL
1E00:        0000                  EXTRN SDT.ADRH
1E00:        0000                  EXTRN SDT.BANK
1E00:        0000                  EXTRN SDT.UNIT
1E00:        0000                  EXTRN BLKD.SIZE
1E00:        0000                  EXTRN BLKDLST
1E00:        0000                  EXTRN CFMGR.INIT       ; CHAR FILE MANAGER INIT
1E00:        0000                  EXTRN MMGR.INIT        ; MEMORY MANAGER INIT
1E00:        0000                  EXTRN BMGR.INIT        ; BUFFER FILE MANAGER INIT
1E00:        0000                  EXTRN BFM.INIT         ; BLOCK FILE MANAGER INIT
1E00:        0000                  EXTRN BFM.INIT2        ; BLOCK FILE MANAGER INIT2
1E00:        0000                  EXTRN CLK.INIT         ; CLOCK SYSTEM CALL INIT
1E00:                   *
1E00:        0000                  EXTRN DIB1             ; ON BOARD DISK DRIVER'S DIBS (1-4)
1E00:        0000                  EXTRN DIB2
1E00:        0000                  EXTRN DIB3
1E00:        0000                  EXTRN DIB4
1E00:                   *
1E00:                   *ENTRY I.BASE.P ; USED BY BFM.INIT2  (HARDWIRED!)
```

```
1E00:
****************************************************************************************************
1E00:              *
1E00:              * FILE DATA DECLARATIONS
1E00:              *
1E00:
****************************************************************************************************
1E00:              * KERNEL FILE
1E00:
****************************************************************************************************
1E00:53 4F 53 20   K.FILE     ASC   "SOS             KRNL"
1E08:62 00         K.HDR.CNT  DW    LDR.ADR-K.DRIVES
1E0A:01            K.DRIVES   DFB   $1
1E0B:00            K.FLAGS    DFB   $0               ; RESERVED FOR FUTURE USE
1E0C:0E            I.PATH     DFB   $E
1E0D:2E 44 31 2F              ASC   ".D1/SOS.INTERP"
1E1B:        0021             DS    $30-$F
1E3C:0E            D.PATH     DFB   $E
1E3D:2E 44 31 2F              ASC   ".D1/SOS.DRIVER"
1E4B:        0021             DS    $30-$F
1E6C:00 00         LDR.ADR    DW    $0
1E6E:88 0C         LDR.CNT    DW    ZZEND-SOSLDR
1E70:
****************************************************************************************************
1E70:              * INTERPRETER/DRIVER FILES    <--+
1E70:              * ERROR MESSAGES                 !    DEFINED IN BACK OF THIS LISTING
1E70:              * WELCOME MESSAGES            <--+
1E70:
****************************************************************************************************
```

```
1E70:
****************************************************************************************************
1E70:                  *
1E70:                  * SOSLDR - DATA DECLARATIONS (1)
1E70:                  *
1E70:
****************************************************************************************************
1E70:        0080      TRUE       EQU   $80
1E70:        0000      FALSE      EQU   $0
1E70:                  *
1E70:        FFD0      Z.REG      EQU   $FFD0
1E70:        FFDF      E.REG      EQU   $FFDF
1E70:        FFEF      B.REG      EQU   $FFEF
1E70:                  *
1E70:        1A00      CZPAGE     EQU   $1A00
1E70:        1B00      CSPAGE     EQU   $1B00
1E70:        1600      CXPAGE     EQU   $1600
1E70:        1800      SZPAGE     EQU   $1800
1E70:        1400      SXPAGE     EQU   $1400
1E70:        0100      SSPAGE     EQU   $0100
1E70:                  *
1E70:        F1B9      ROM.ADR    EQU   $F1B9
1E70:        00A0      ROM.ID     EQU   $A0
```

```
1E70:
****************************************************************************************************
1E70:              *
1E70:              * SOSLDR - DATA DECLARATIONS (2)
1E70:              *
1E70:
****************************************************************************************************
1E70:        0000  ZPAGE      EQU   $00
1E70:              *
1E70:        0000  K.BASE     EQU   ZPAGE+$0         ; SOSLDR1 SUBROUTINE    +--------------------------------------+
1E70:        0002  I.BASE.P   EQU   ZPAGE+$2         ;                       ! <VARNAME>.P ::= 3 BYTE ZPAGE POINTER !
1E70:        0004  RDBUF.P    EQU   ZPAGE+$4         ;                       +--------------------------------------+
1E70:        0006  SYSBUF.P   EQU   ZPAGE+$6
1E70:        0008  TEMP.BANK  EQU   ZPAGE+$8
1E70:        0009  TEMP.ADRH  EQU   ZPAGE+$9
1E70:        000A  WORK.P     EQU   ZPAGE+$A
1E70:              *
1E70:        000C  REV.SAVE   EQU   ZPAGE+$C         ; REVERSE SUBROUTINE
1E70:              *
1E70:        0010  FIRST.ADIB EQU   ZPAGE+$10        ; FLAGS SUBROUTINE
1E70:        0012  PREV.ADIB.P EQU  ZPAGE+$12
1E70:        0014  DIB.P      EQU   ZPAGE+$14
1E70:        0016  PG.ALIGN   EQU   ZPAGE+$16
1E70:        0014  DIB.FLAGS  EQU   $14
1E70:        0020  DIB.DCB    EQU   $20
1E70:              *
1E70:        0018  PREVBANK   EQU   ZPAGE+$18        ; GETMEM SUBROUTINE
1E70:        0019  PREVDST    EQU   ZPAGE+$19
1E70:              *
1E70:        001C  CODE.P     EQU   ZPAGE+$1C        ; RELOCATION SUBROUTINE
1E70:        001E  REL.P      EQU   ZPAGE+$1E
1E70:        0020  REL.END    EQU   ZPAGE+$20
1E70:              *
1E70:        0022  SRC.P      EQU   ZPAGE+$22        ; MOVE SUBROUTINE
1E70:        0024  DST.P      EQU   ZPAGE+$24
1E70:        0026  CNT        EQU   ZPAGE+$26
1E70:              *
1E70:        002A  DSTBANK    EQU   ZPAGE+$2A        ; LINK SUBROUTINE
1E70:        002C  LINK.P     EQU   ZPAGE+$2C
1E70:              *
1E70:        0002  DIB.ENTRY  EQU   2                ; ALLOC.DEV SUBROUTINE
1E70:        0016  DIB.UNIT   EQU   4+16+2
1E70:        0017  DIB.DTYPE  EQU   4+16+3
1E70:              *
1E70:        002E  ETEMP      EQU   ZPAGE+$2E        ; ERROR SUBROUTINE
1E70:              *
1E70:        002F  WTEMP      EQU   ZPAGE+$2F        ; WELCOME SUBROUTINE
1E70:                         CHN   SOSLDR.D.SRC     ;!BITROT
```

```
1E70:
****************************************************************************************************
1E70:                  *
1E70:                  * SOS LOADER -
1E70:                  *
1E70:                  * (MAIN PROGRAM)
1E70:
****************************************************************************************************
1E70:        1E70      SOSLDR     EQU   *                ;                                          +---------------+
1E70:A9 00                        LDA   #0               ; ZERO SOS/USER X, Z AND STACK PAGES       ! SEE FIGURE 1. !
1E72:AA                           TAX                    ;                                          +---------------+
1E73:9D 00 1A          SLDR010    STA   CZPAGE,X
1E76:9D 00 16                     STA   CXPAGE,X
1E79:9D 00 1B                     STA   CSPAGE,X
1E7C:9D 00 18                     STA   SZPAGE,X
1E7F:9D 00 14                     STA   SXPAGE,X
1E82:9D 00 01                     STA   SSPAGE,X
1E85:CA                           DEX
1E86:D0 EB   1E73                 BNE   SLDR010
1E88:                  *                                       ; SETUP SOS CALL ENVIRONMENT (WRITE PROTECT=OFF)
1E88:A9 30                        LDA   #$30             ; E:=( 0.0.1.1:0.0.0.0 )
1E8A:8D DF FF                     STA   E.REG            ;    ( 1.I.S.R:W.P.R.R )
1E8D:                  *
1E8D:A2 FB                        LDX   #$FB             ; CONSOLE 1.0 MODIFIES STACK DURING D.INIT CALL
1E8F:9A                           TXS
1E90:A9 1A                        LDA   #<CZPAGE         ; ZREG:=CALLER'S Z PAGE
1E92:8D D0 FF                     STA   Z.REG
1E95:                  *                                       ; +-------------------------------+
1E95:20 D4 1F                     JSR   SOSLDR1          ; ! PROCESS KRNL/INTERP/DRVR FILES !
1E98:                  *                                       ; +-------------------------------+
1E98:AD DF FF                     LDA   E.REG
1E9B:29 10                        AND   #$10             ; SETUP SOS CALL ENVIRONMENT (WRITE PROTECT=ON)
1E9D:09 28                        ORA   #$28             ; E:=( 0.0.1.X:1.0.0.0 )
1E9F:8D DF FF                     STA   E.REG            ;    ( 1.I.S.R:W.P.R.R )
1EA2:                  *
1EA2:A2 FF                        LDX   #$FF             ; STACK.REG:=$FF
1EA4:9A                           TXS
1EA5:A9 1A                        LDA   #<CZPAGE         ; ZREG:=CALLER'S Z PAGE
1EA7:8D D0 FF                     STA   Z.REG
1EAA:                  *                                                                                  +--------------
-+
1EAA:AD 00 00                     LDA   SYSBANK          ; BREG:=SYSBANK                            ! SEE FIGURE 4. !
1EAD:8D EF FF                     STA   B.REG            ;                                          +---------------+
1EB0:6C 02 00                     JMP   (I.BASE.P)       ; SOS LOAD COMPLETE - JMP TO INTERPRETER
1EB3:                  *
1EB3:                  *THE END.
1EB3:
****************************************************************************************************
```

```
1EB3:
****************************************************************************************************
1EB3:                  *
1EB3:                  * MOVE ( IN:   SRC.P
1EB3:                  *        IN:   DST.P
1EB3:                  *        IN:   A="BANK"
1EB3:                  *        IN:   CNT      )
1EB3:                  *
1EB3:                  *        LOCAL:  END
1EB3:                  * (MOVES SRC.P..SRC.P+CNT-1 TO DST.P..DST.P+CNT-1)                "CNT PARM IS DESTROYED"
1EB3:
****************************************************************************************************
1EB3:        1EB3      MOVE       EQU   *
1EB3:AA                           TAX
1EB4:AD EF FF                     LDA   B.REG            ; SAVE BANK REGISTER
1EB7:48                           PHA
1EB8:8E EF FF                     STX   B.REG            ; BREG:=A
1EBB:A5 27                        LDA   CNT+1            ; IF CNT <> 0
1EBD:05 26                        ORA   CNT              ;    THEN
1EBF:F0 33   1EF4                 BEQ   MOVE.EXIT
1EC1:A5 26                        LDA   CNT              ;       CNT:=CNT-1
1EC3:D0 02   1EC7                 BNE   MOVE010
1EC5:C6 27                        DEC   CNT+1
1EC7:C6 26                 MOVE010    DEC   CNT
1EC9:18                           CLC                    ;       SRC.P:=SRC.P+PAGE.CNT
1ECA:A5 23                        LDA   SRC.P+1
1ECC:65 27                        ADC   CNT+1
1ECE:85 23                        STA   SRC.P+1
1ED0:A5 25                        LDA   DST.P+1          ;       DST.P:=DST.P+PAGE.CNT
1ED2:65 27                        ADC   CNT+1
1ED4:85 25                        STA   DST.P+1
1ED6:E6 27                        INC   CNT+1            ;       PAGE.CNT:=PAGE.CNT+1
1ED8:A4 26                        LDY   CNT              ;       Y:=BYTE.CNT
1EDA:F0 07   1EE3                 BEQ   MOVE020          ;       IF Y=0 THEN M2
1EDC:                  *
1EDC:B1 22                 MOVE.PAGE  LDA   (SRC.P),Y        ;M1:    DO
1EDE:91 24                        STA   (DST.P),Y        ;          (DST.P),Y:=(SRC.P),Y
1EE0:88                           DEY                    ;          Y:=Y-1
1EE1:D0 F9   1EDC                 BNE   MOVE.PAGE        ;       UNTIL  Y=0
1EE3:B1 22                 MOVE020    LDA   (SRC.P),Y        ;M2:    (DST.P),Y:=(SRC.P),Y
1EE5:91 24                        STA   (DST.P),Y
1EE7:88                           DEY                    ;       Y:=Y-1
1EE8:C6 23                        DEC   SRC.P+1          ;       SRC.P:=SRC.P-256
1EEA:C6 25                        DEC   DST.P+1          ;       DST.P:=DST.P-256
1EEC:C6 27                        DEC   CNT+1            ;       PAGE.CNT:=PAGE.CNT-1
1EEE:D0 EC   1EDC                 BNE   MOVE.PAGE        ;       IF PAGE.CNT <> 0 THEN M1
1EF0:                  *
1EF0:E6 23                        INC   SRC.P+1          ; RESTORE SRC.P
1EF2:E6 25                        INC   DST.P+1          ;    "    DST.P
1EF4:                  *
1EF4:68                    MOVE.EXIT  PLA                    ; RESTORE BANK REGISTER
1EF5:8D EF FF                     STA   B.REG
1EF8:60                           RTS
```

```
1EF9:
****************************************************************************************************
1EF9:              *
1EF9:              * LINK ( IN:   DST.P
1EF9:              *        IN:   DSTBANK
1EF9:              *        IN:   PREVBANK
1EF9:              *        IN:   FIRST.ADIB
1EF9:              *        I/O:  SDT.TBL
1EF9:              *        I/O:  BLKDLST
1EF9:              *        OUT:  LINKED DRIVER MODULE )
1EF9:              *
1EF9:              *        OWN:  LINK.P
1EF9:              * (LINKS FIRST DIB TO PREVIOUS DRIVER'S LAST "ACTIVE" DIB, AND ADDS SDT ENTRY)
1EF9:
****************************************************************************************************
1EF9:        1EF9  LINK       EQU   *
1EF9:18                       CLC                    ; FIRST.ADIB:=0:DST.P+FIRST.ADIB
1EFA:A5 24                    LDA   DST.P
1EFC:65 10                    ADC   FIRST.ADIB
1EFE:85 10                    STA   FIRST.ADIB
1F00:A5 25                    LDA   DST.P+1
1F02:65 11                    ADC   FIRST.ADIB+1
1F04:85 11                    STA   FIRST.ADIB+1
1F06:A9 00                    LDA   #0
1F08:8D 11 16                 STA   CXPAGE+FIRST.ADIB+1
1F0B:A5 18                    LDA   PREVBANK         ; BREG:=PREVBANK
1F0D:8D EF FF                 STA   B.REG
1F10:A0 00                    LDY   #0               ; (LINK.P):=FIRST.ADIB
1F12:A5 10                    LDA   FIRST.ADIB
1F14:91 2C                    STA   (LINK.P),Y
1F16:C8                       INY
1F17:A5 11                    LDA   FIRST.ADIB+1
1F19:91 2C                    STA   (LINK.P),Y
1F1B:A5 2A                    LDA   DSTBANK          ; BREG:=DSTBANK
1F1D:8D EF FF                 STA   B.REG
1F20:A5 10                    LDA   FIRST.ADIB       ; LINK.P:=FIRST.ADIB
1F22:85 2C                    STA   LINK.P
1F24:A5 11                    LDA   FIRST.ADIB+1
1F26:85 2D                    STA   LINK.P+1
1F28:20 79 1F      WALKLINKS  JSR   ALLOC.DEV        ; ALLOC.DEV(LINK.P BREG.IN, SDT.TBL BLKDLST.IO)
1F2B:A0 00         LINK010    LDY   #0               ; WHILE (LINK.P) <> 0 AND (LINK.P) <> LINK.P
1F2D:B1 2C                    LDA   (LINK.P),Y
1F2F:C8                       INY
1F30:11 2C                    ORA   (LINK.P),Y
1F32:F0 1F   1F53             BEQ   LINK100
1F34:B1 2C                    LDA   (LINK.P),Y
1F36:C5 2D                    CMP   LINK.P+1
1F38:D0 07   1F41             BNE   LINK030
1F3A:88                       DEY
1F3B:B1 2C                    LDA   (LINK.P),Y
1F3D:C5 2C                    CMP   LINK.P
1F3F:F0 12   1F53             BEQ   LINK100
1F41:A0 00         LINK030    LDY   #0               ;    DO  LINK.P:=(LINK.P)
1F43:B1 2C                    LDA   (LINK.P),Y
1F45:AA                       TAX
1F46:C8                       INY
1F47:B1 2C                    LDA   (LINK.P),Y
1F49:86 2C                    STX   LINK.P
```

```
1F4B:85 2D                    STA   LINK.P+1
1F4D:20 79 1F                 JSR   ALLOC.DEV        ;     "  ALLOC.DEV(LINK.P BREG.IN, SDT.TBL BLKDLST.IO)
1F50:4C 2B 1F                 JMP   LINK010
1F53:              *
1F53:A0 00         LINK100    LDY   #0               ; (LINK.P):=0
1F55:98                       TYA
1F56:91 2C                    STA   (LINK.P),Y
1F58:C8                       INY
1F59:91 2C                    STA   (LINK.P),Y
1F5B:88                       DEY                    ; BREG:=0
1F5C:8C EF FF                 STY   B.REG
1F5F:60                       RTS
1F60:              *
1F60:              *
1F60:              *
1F60:              *
1F60:              * LINK.INIT ( IN:   A=# DRIVES
1F60:              *             IN:   DIB1..4
1F60:              *             I/O:  SDT.TBL
1F60:              *             I/O:  BLKDLST    )
1F60:              *
1F60:        1F60  LINK.INIT  EQU   *
1F60:20 32 22                 JSR   SET.DRIVES       ; SET.DRIVES(A=#DRIVES.IN, DIB1..4.IN)
1F63:A9 00                    LDA   #0
1F65:8D 00 00                 STA   MAX.DNUM         ; MAXDNUM:=0
1F68:8D 00 00                 STA   BLKDLST          ; BLKDLST:=0
1F6B:8D 2D 16                 STA   CXPAGE+LINK.P+1  ; LINK.P:=0:DIB1
1F6E:A9 00                    LDA   #>DIB1
1F70:85 2C                    STA   LINK.P
1F72:A9 00                    LDA   #<DIB1
1F74:85 2D                    STA   LINK.P+1
1F76:4C 28 1F                 JMP   WALKLINKS
```

```
1F79:
****************************************************************************************************
1F79:              *
1F79:              * ALLOC.DEV ( IN:   LINK.P
1F79:              *             IN:   B.REG
1F79:              *             I/O:  SDT.TBL                                                   (SYSTEM DEVICE
TABLE)
1F79:              *                      IN:   SDT.SIZE  = CONSTANT
1F79:              *                      IN:   DIB.ENTRY = CONSTANT                       DEV   DIB   ADR  BANK
UNIT
1F79:              *                      IN:   DIB.UNIT  = CONSTANT                           !-----!-----!-----!---
--!
1F79:              *                      IN:   DIB.DTYPE = CONSTANT                        1  !     !     !     !
!
1F79:              *                      I/O:  MAX.DNUM                                    2  !     !     !     !
!
1F79:              *                      OUT:  SDT.BANK                                    .  !     !     !     !
!
1F79:              *                      OUT:  SDT.DIB                                     .  !     !     !     !
!
1F79:              *                      OUT:  SDT.ADR                                     .  !-----!-----!-----!---
--!
1F79:              *                      OUT:  SDT.UNIT                                 MAX.DNUM
1F79:              *             I/O:  BLKDLST
1F79:              *                      IN:   BLKD.SIZE = CONSTANT
1F79:              * (ADDS A NEW ENTRY TO THE DEVICE MANAGER'S SYSTEM DEVICE TABLE (SDT))
1F79:
****************************************************************************************************
1F79:        1F79  ALLOC.DEV  EQU   *
1F79:EE 00 00                 INC   MAX.DNUM         ; MAX.DNUM:=MAX.DNUM+1
1F7C:AE 00 00                 LDX   MAX.DNUM         ; IF MAX.DNUM >= SDT.SIZE
1F7F:E0 00                    CPX   #>SDT.SIZE       ;    THEN
1F81:90 07   1F8A             BCC   ADEV010
1F83:A2 C4                    LDX   #ERR8X           ;       ERROR("TOO MANY DEVICES")
1F85:A0 10                    LDY   #ERR8L
1F87:20 E2 25                 JSR   ERROR
1F8A:AD EF FF      ADEV010    LDA   B.REG            ; SDT.BANK,X:=BREG
1F8D:9D 00 00                 STA   SDT.BANK,X
1F90:18                       CLC                    ; SDT.DIB,X:=LINK.P+4
1F91:A5 2C                    LDA   LINK.P
1F93:69 04                    ADC   #4
1F95:9D 00 00                 STA   SDT.DIBL,X
1F98:A5 2D                    LDA   LINK.P+1
1F9A:69 00                    ADC   #0
1F9C:9D 00 00                 STA   SDT.DIBH,X
1F9F:38                       SEC                    ; SDT.ADR,X:=(LINK.P),DIB.ENTRY-1
1FA0:A0 02                    LDY   #DIB.ENTRY
1FA2:B1 2C                    LDA   (LINK.P),Y
1FA4:E9 01                    SBC   #1
1FA6:9D 00 00                 STA   SDT.ADRL,X
1FA9:C8                       INY
1FAA:B1 2C                    LDA   (LINK.P),Y
1FAC:E9 00                    SBC   #0
1FAE:9D 00 00                 STA   SDT.ADRH,X
1FB1:A0 16                    LDY   #DIB.UNIT        ; SDT.UNIT,X:=(LINK.P),DIB.UNIT
1FB3:B1 2C                    LDA   (LINK.P),Y
1FB5:9D 00 00                 STA   SDT.UNIT,X
1FB8:A0 17                    LDY   #DIB.DTYPE       ; IF (LINK.P),DIB.DTYPE = "BLOCK DEVICE"
1FBA:B1 2C                    LDA   (LINK.P),Y
1FBC:10 15   1FD3             BPL   ADEV.EXIT
1FBE:8A                       TXA                    ;    THEN
1FBF:EE 00 00                 INC   BLKDLST          ;       BLKDLST:=BLKDLST+1
1FC2:AE 00 00                 LDX   BLKDLST          ;       IF BLKDLST >= BLKD.SIZE
1FC5:E0 00                    CPX   #>BLKD.SIZE      ;          THEN
1FC7:90 07   1FD0             BCC   ADEV020
1FC9:A2 DA                    LDX   #ERR9X           ;             ERROR("TOO MANY BLOCK DEVICES")
```

```
1FCB:A0 16                    LDY   #ERR9L
1FCD:20 E2 25                 JSR   ERROR
1FD0:9D 00 00      ADEV020    STA   BLKDLST,X        ;       BLKDLST,X:=MAX.DNUM
1FD3:60            ADEV.EXIT  RTS                    ; RETURN
```

```
1FD4:
****************************************************************************************************
1FD4:              *
1FD4:              * SOSLDR1 ()
1FD4:              *
1FD4:              * (PROCESSES KERNEL/INTERPRETER/DRIVER FILES)
1FD4:
****************************************************************************************************
1FD4:        1FD4  SOSLDR1    EQU   *
1FD4:A2 1F                    LDX   #$1F             ; COPY ROM'S DISK CORE ROUTINE ZPAGE VARS TO SOS ZPAGE
1FD6:BD 80 03      LDR010     LDA   $380,X
1FD9:9D 00 18                 STA   SZPAGE,X
1FDC:CA                       DEX
1FDD:10 F7   1FD6             BPL   LDR010
1FDF:
****************************************************************************************************
1FDF:              * PROCESS KERNEL FILE
1FDF:
****************************************************************************************************
1FDF:              *
1FDF:              * MOVE AND INITIALIZE SOS GLOBALS
1FDF:              *
1FDF:A9 6C                    LDA   #>LDR.ADR        ; WORK.P:=0:LDR.ADR
1FE1:85 0A                    STA   WORK.P
1FE3:A9 1E                    LDA   #<LDR.ADR
1FE5:85 0B                    STA   WORK.P+1
1FE7:20 BA 22                 JSR   ADVANCE          ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
1FEA:              *
1FEA:AD EF FF                 LDA   B.REG            ; MOVE(SRC.P DST.P A=BREG CNT.IN)
1FED:20 B3 1E                 JSR   MOVE
1FF0:              *
1FF0:AD EF FF                 LDA   B.REG            ; SYSBANK:=BREG
1FF3:29 0F                    AND   #$0F
1FF5:8D 00 00                 STA   SYSBANK
1FF8:0A                       ASL   A                ; MEMSIZ:=SYSBANK*2+4 "16K CHUNKS"
1FF9:18                       CLC
1FFA:69 04                    ADC   #4
1FFC:8D 00 00                 STA   MEMSIZE          ; AND, MEMSIZE (SIZE IN 16K BYTE "CHUNKS")
1FFF:              *
1FFF:              * MOVE KERNAL CODE
1FFF:              *
1FFF:20 BA 22                 JSR   ADVANCE          ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
2002:              *
2002:A5 24                    LDA   DST.P            ; K.BASE:=DST.P
2004:85 00                    STA   K.BASE
2006:A5 25                    LDA   DST.P+1
2008:85 01                    STA   K.BASE+1
200A:AD EF FF                 LDA   B.REG            ; MOVE(SRC.P DST.P A=BREG CNT.IN)
200D:20 B3 1E                 JSR   MOVE
2010:              *
2010:              * MOVE LOADER TO BANK 0 AND SWITCH FROM SYSTEM BANK TO BANK 0
2010:              *
2010:A9 00                    LDA   #>$2000          ; MOVE(SRC.P=0:2000 DST.P=8F:2000 A=BREG CNT=LDR.END-$2000)
2012:85 22                    STA   SRC.P
2014:85 24                    STA   DST.P
2016:A9 20                    LDA   #<$2000
2018:85 23                    STA   SRC.P+1
201A:85 25                    STA   DST.P+1
201C:A9 8F                    LDA   #$8F
201E:8D 25 16                 STA   CXPAGE+DST.P+1
```

```
2021:A9 F8                    LDA   #>LDREND-$2000
2023:85 26                    STA   CNT
2025:A9 0A                    LDA   #<LDREND-$2000
2027:85 27                    STA   CNT+1
2029:AD EF FF                 LDA   B.REG
202C:20 B3 1E                 JSR   MOVE
202F:A9 00                    LDA   #0               ; BREG:=0
2031:8D EF FF                 STA   B.REG
2034:              *
2034:              * INITIALIZE SDT TABLE, KERNEL AND PRINT WELCOME MESSAGE
2034:              *
2034:AD 0A 1E                 LDA   K.DRIVES         ; LINK.INIT(A=K.DRIVES DIB1..4.IN, SDT.TBL BLKDLST.IO)
2037:20 60 1F                 JSR   LINK.INIT
203A:20 7B 22                 JSR   INIT.KRNL        ; INIT.KRNL()
203D:20 F1 26                 JSR   WELCOME          ; WELCOME()
2040:              *
2040:AD DF FF                 LDA   E.REG            ; ENABLE ROM BANK
2043:09 03                    ORA   #$03
2045:8D DF FF                 STA   E.REG
2048:AD B9 F1                 LDA   ROM.ADR          ; IF MONITOR ROM <> NEW
204B:C9 A0                    CMP   #ROM.ID          ;    THEN
204D:F0 07   2056             BEQ   LDR020
204F:A2 B4                    LDX   #ERR7X           ;       ERROR("ROM ERROR:  PLEASE NOTIFY YOUR DEALER")
2051:A0 25                    LDY   #ERR7L
2053:20 E2 25                 JSR   ERROR
2056:AD DF FF      LDR020     LDA   E.REG            ; DISABLE ROM BANK
2059:29 F6                    AND   #$F6
205B:8D DF FF                 STA   E.REG
205E:
****************************************************************************************************
205E:              * PROCESS INTERPRETER FILE
205E:
****************************************************************************************************
205E:              *
205E:              * OPEN SOS INTERPRETER FILE (DEFAULT='SOS.INTERP')
205E:              *
205E:AC 0C 1E                 LDY   I.PATH           ; OPEN(PATHNAME:=I.PATH
2061:B9 0C 1E      LDR030     LDA   I.PATH,Y         ;      REFNUM=OPEN.REF
2064:99 21 28                 STA   PATH,Y           ;      SYSBUF.P:=80:LDREND-2000 )
2067:88                       DEY
2068:10 F7   2061             BPL   LDR030
206A:              *
206A:A9 F8                    LDA   #>LDREND-$2000
206C:85 06                    STA   SYSBUF.P
206E:A9 0A                    LDA   #<LDREND-$2000
2070:85 07                    STA   SYSBUF.P+1
2072:A9 80                    LDA   #$80
2074:8D 07 16                 STA   CXPAGE+SYSBUF.P+1
2077:              *
2077:              *
2077:00                       BRK
2078:C8                       DFB   OPEN
2079:16 28                    DW    OPEN.PARMS
207B:F0 07   2084             BEQ   LDR040
207D:A2 22                    LDX   #ERR1X           ; ERROR("INTERPRETER FILE NOT FOUND")
207F:A0 1A                    LDY   #ERR1L
2081:20 E2 25                 JSR   ERROR
2084:AD 19 28      LDR040     LDA   OPEN.REF
```

```
2087:8D 72 28                 STA   READ.REF
208A:8D 7A 28                 STA   CLOSE.REF
208D:              *
208D:              * READ IN ENTIRE INTERPRETER FILE
208D:              *
208D:A9 80                    LDA   #$80             ; READ(REFNUM=READ.REF
208F:8D 05 16                 STA   CXPAGE+RDBUF.P+1 ;      RDBUF.P:=80:FILE
2092:A9 F8                    LDA   #>FILE           ;      BYTES=$FFFF-FILE+1
2094:85 04                    STA   RDBUF.P          ;      BYTESRD=I.BYTESRD )
2096:A9 0E                    LDA   #<FILE
2098:85 05                    STA   RDBUF.P+1
209A:              *
209A:00                       BRK
209B:CA                       DFB   READ
209C:71 28                    DW    READ.PARMS
209E:F0 07   20A7             BEQ   LDR050
20A0:A2 08                    LDX   #ERR0X           ; ERROR("I/O ERROR")
20A2:A0 09                    LDY   #ERR0L
20A4:20 E2 25                 JSR   ERROR
20A7:              *                                                                  +--------------
-+
20A7:              * CLOSE INTERPRETER FILE AND CHECK LABEL                           ! SEE FIGURE 2.
!
20A7:              *                                                                  +--------------
-+
20A7:00            LDR050     BRK                    ; CLOSE(REFNUM=CLOSE.REF)
20A8:CC                       DFB   CLOSE
20A9:79 28                    DW    CLOSE.PARMS
20AB:A0 07                    LDY   #7               ; CHECK LABEL
20AD:B1 04         LDR051     LDA   (RDBUF.P),Y
20AF:D9 61 28                 CMP   I.LABEL,Y
20B2:D0 05   20B9             BNE   LDR052
20B4:88                       DEY
20B5:10 F6   20AD             BPL   LDR051
20B7:30 07   20C0             BMI   LDR053
20B9:A2 3A         LDR052     LDX   #ERR2X           ; ERROR("INVALID INTERPRETER FILE")
20BB:A0 18                    LDY   #ERR2L
20BD:20 E2 25                 JSR   ERROR
20C0:              *
20C0:              * MOVE INTERPRETER CODE
20C0:              *
20C0:A9 FE         LDR053     LDA   #>I.HDR.CNT-2    ; WORK.P:=80:I.HDR.CNT-2
20C2:85 0A                    STA   WORK.P
20C4:A9 0E                    LDA   #<I.HDR.CNT-2
20C6:85 0B                    STA   WORK.P+1
20C8:A9 80                    LDA   #$80
20CA:8D 0B 16                 STA   CXPAGE+WORK.P+1
20CD:              *
20CD:20 BA 22                 JSR   ADVANCE          ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
20D0:              *
20D0:A5 24                    LDA   DST.P            ; I.BASE.P:=0:DST.P
20D2:85 02                    STA   I.BASE.P
20D4:A5 25                    LDA   DST.P+1
20D6:85 03                    STA   I.BASE.P+1
20D8:A9 00                    LDA   #0
20DA:8D 03 16                 STA   CXPAGE+I.BASE.P+1
20DD:              *
20DD:18                       CLC                    ; IF DST.P+CNT > K.BASE THEN ERROR
20DE:A5 26                    LDA   CNT
```

```
20E0:65 24                    ADC   DST.P
20E2:AA                       TAX
20E3:A5 27                    LDA   CNT+1
20E5:65 25                    ADC   DST.P+1
20E7:E4 00                    CPX   K.BASE
20E9:E5 01                    SBC   K.BASE+1
20EB:F0 09   20F6             BEQ   LDR070
20ED:90 07   20F6             BCC   LDR070
20EF:A2 52                    LDX   #ERR3X           ; ERROR("INCOMPATIBLE INTERPRETER")
20F1:A0 18                    LDY   #ERR3L
20F3:20 E2 25                 JSR   ERROR
20F6:              *
20F6:AD 00 00      LDR070     LDA   SYSBANK          ; MOVE(SRC.P=RDBUF.P DST.P A=SYSBANK CNT.IN)
20F9:20 B3 1E                 JSR   MOVE
20FC:
****************************************************************************************************
20FC:              * PROCESS DRIVER FILE
20FC:
****************************************************************************************************
20FC:              *
20FC:              * OPEN SOS DRIVER FILE (DEFAULT='SOS.DRIVER')
20FC:              *
20FC:AC 3C 1E                 LDY   D.PATH           ; OPEN(PATHNAME:=D.PATH
20FF:B9 3C 1E      LDR080     LDA   D.PATH,Y         ;      REFNUM=OPEN.REF
2102:99 21 28                 STA   PATH,Y           ;      SYSBUF.P:=80:LDREND-2000 )
2105:88                       DEY
2106:10 F7   20FF             BPL   LDR080
2108:              *
2108:00                       BRK
2109:C8                       DFB   OPEN
210A:16 28                    DW    OPEN.PARMS
210C:F0 07   2115             BEQ   LDR090
210E:A2 67                    LDX   #ERR4X           ; ERROR("DRIVER FILE NOT FOUND")
2110:A0 15                    LDY   #ERR4L
2112:20 E2 25                 JSR   ERROR
2115:AD 19 28      LDR090     LDA   OPEN.REF
2118:8D 72 28                 STA   READ.REF
211B:8D 7A 28                 STA   CLOSE.REF
211E:              *
211E:              * READ IN ENTIRE DRIVER FILE INTO BANK 0
211E:              *
211E:00                       BRK                    ; READ(REFNUM=READ.REF
211F:CA                       DFB   READ             ;      RDBUF.P:=80:FILE
2120:71 28                    DW    READ.PARMS       ;      BYTES=$FFFF-FILE+1
2122:              *                                       ;      BYTESRD=D.BYTESRD )
2122:F0 07   212B             BEQ   LDR100
2124:A2 08                    LDX   #ERR0X           ; ERROR("I/O ERROR")
2126:A0 09                    LDY   #ERR0L
2128:20 E2 25                 JSR   ERROR
212B:              *                                                                                  +--------------
-+
212B:              * CLOSE THE DRIVER FILE AND CHECK LABEL                                            ! SEE FIGURE 3.
!
212B:              *                                                                                  +--------------
-+
212B:00            LDR100     BRK                    ; CLOSE(REFNUM=CLOSE.REF)
212C:CC                       DFB   CLOSE
212D:79 28                    DW    CLOSE.PARMS
212F:A0 07                    LDY   #$7              ; CHECK LABEL
2131:B1 04         LDR101     LDA   (RDBUF.P),Y
2133:D9 69 28                 CMP   D.LABEL,Y
```

```
2136:D0 05   213D             BNE   LDR102
2138:88                       DEY
2139:10 F6   2131             BPL   LDR101
213B:30 07   2144             BMI   LDR103
213D:A2 7A         LDR102     LDX   #ERR5X            ; ERROR("INVALID DRIVER FILE")
213F:A0 13                    LDY   #ERR5L
2141:20 E2 25                 JSR   ERROR
2144:              *
2144:              * MOVE CHARACTER SET TABLE
2144:              *
2144:A9 14         LDR103     LDA   #>D.CHRSET        ; MOVE(SRC.P=D.CHRSET DST.P=$C00 A=0 CNT=$400)
2146:85 22                    STA   SRC.P
2148:A9 0F                    LDA   #<D.CHRSET
214A:85 23                    STA   SRC.P+1
214C:A9 00                    LDA   #>$C00
214E:85 24                    STA   DST.P
2150:A9 0C                    LDA   #<$C00
2152:85 25                    STA   DST.P+1
2154:A9 00                    LDA   #>$400
2156:85 26                    STA   CNT
2158:A9 04                    LDA   #<$400
215A:85 27                    STA   CNT+1
215C:A9 00                    LDA   #0
215E:20 B3 1E                 JSR   MOVE
2161:              *
2161:              * MOVE KEYBOARD TABLE
2161:              *
2161:A9 24                    LDA   #>D.KYBD          ; MOVE(SRC.P=D.KYBD DST.P=$1700 A=0 CNT=$100.IN)
2163:85 22                    STA   SRC.P
2165:A9 13                    LDA   #<D.KYBD
2167:85 23                    STA   SRC.P+1
2169:A9 00                    LDA   #>$1700
216B:85 24                    STA   DST.P
216D:A9 17                    LDA   #<$1700
216F:85 25                    STA   DST.P+1
2171:A9 00                    LDA   #>$100
2173:85 26                    STA   CNT
2175:A9 01                    LDA   #<$100
2177:85 27                    STA   CNT+1
2179:A9 00                    LDA   #0
217B:20 B3 1E                 JSR   MOVE
217E:              *
217E:              * RE-INITIALIZE SDT TABLE
217E:              *
217E:A0 0A                    LDY   #>D.DRIVES-D.FILE ; LINK.INIT(A=D.DRIVES DIB1..4.IN, SDT.TBL BLKDLST.IO)
2180:B1 04                    LDA   (RDBUF.P),Y
2182:20 60 1F                 JSR   LINK.INIT
2185:              *
2185:A9 00                    LDA   #0                ; DST.P:=0:I.BASE.P/256*256
2187:8D 25 16                 STA   CXPAGE+DST.P+1
218A:85 24                    STA   DST.P
218C:A5 03                    LDA   I.BASE.P+1
218E:85 25                    STA   DST.P+1
2190:C9 A0                    CMP   #$A0              ; IF DST.P>=$A000 THEN DST.P:=$A000
2192:90 04   2198             BCC   LDR105
2194:A9 A0                    LDA   #$A0
```

```
2196:85 25                   STA   DST.P+1
2198:AD 00 00     LDR105     LDA   SYSBANK           ; DSTBANK:=SYSBANK
219B:85 2A                   STA   DSTBANK
219D:20 FE 22                JSR   REVERSE           ; REVERSE(D.HDR.CNT.IN, WORK.P.OUT)
21A0:             *
21A0:             * RELOCATE AND MOVE DRIVERS
21A0:             *
21A0:20 7E 23     NEXTDRIVER JSR   DADVANCE          ; "NO DRIVERS LEFT":=DADVANCE(WORK.P.IO SRC.P CNT REL.P.OUT)
21A3:B0 43   21E8            BCS   LDR140
21A5:20 DB 23                JSR   FLAGS             ; "INACTIVE":=FLAGS(SRC.P.IN, PG.ALIGN FIRST.ADIB.OUT)
21A8:70 F6   21A0            BVS   NEXTDRIVER
21AA:20 8C 24                JSR   GETMEM            ; GETMEM(PG.ALIGN CNT.IN, DST.P DSTBANK DSEGLIST.IO,
PREVBANK.OUT)
21AD:20 19 25                JSR   RELOC             ; RELOC(SRC.P REL.P DST.P.IN)
21B0:             *
21B0:A5 2A                   LDA   DSTBANK           ; IF DSTBANK < 0 OR DST.P < SRC.P THEN ERROR
21B2:30 22   21D6            BMI   LDR120
21B4:AD 23 16                LDA   CXPAGE+SRC.P+1    ;    (CONVERT SRC.P TO BANK SWITCHED ADDRESS)
21B7:29 7F                   AND   #$7F
21B9:85 08                   STA   TEMP.BANK
21BB:A5 23                   LDA   SRC.P+1
21BD:10 02   21C1            BPL   LDR110
21BF:E6 08                   INC   TEMP.BANK
21C1:29 7F        LDR110     AND   #$7F
21C3:18                      CLC
21C4:69 20                   ADC   #<$2000
21C6:85 09                   STA   TEMP.ADRH
21C8:A5 24                   LDA   DST.P             ;    (NOW COMPARE)
21CA:C5 22                   CMP   SRC.P
21CC:A5 25                   LDA   DST.P+1
21CE:E5 09                   SBC   TEMP.ADRH
21D0:A5 2A                   LDA   DSTBANK
21D2:E5 08                   SBC   TEMP.BANK
21D4:B0 07   21DD            BCS   LDR130
21D6:A2 8F        LDR120     LDX   #ERR6X            ;    ERROR("DRIVER FILE TOO LARGE")
21D8:A0 15                   LDY   #ERR6L
21DA:20 E2 25                JSR   ERROR
21DD:             *
21DD:A5 2A        LDR130     LDA   DSTBANK           ; MOVE(SRC.P DST.P A=DSTBANK CNT.IN)
21DF:20 B3 1E                JSR   MOVE
21E2:20 F9 1E                JSR   LINK              ; LINK(DST.P DSTBANK PREVBANK FIRST.ADIB.IN, SDT.TBL BLKDLST.IO)
21E5:4C A0 21                JMP   NEXTDRIVER
21E8:
****************************************************************************************************
21E8:             * SETUP USER ENVIRONMENT
21E8:
****************************************************************************************************
21E8:             *
21E8:             * RE-INITIALIZE KERNEL/DRIVERS, ALLOCATE SYSTEM SEGMENTS
21E8:             *
21E8:20 7B 22     LDR140     JSR   INIT.KRNL         ; INIT.KRNL()
21EB:20 6A 25                JSR   ALLOC.SEG         ; ALLOC.SEG(K.BASE I.BASE.P SYSBANK.IN)
21EE:20 C0 25                JSR   ALLOC.DSEG        ; ALLOC.DSEG(DSEGLIST.IN)
21F1:             *
21F1:             * SET PREFIX TO THE BOOT VOLUME
21F1:             *
21F1:A9 00                   LDA   #0                ; TURN VIDEO OFF - PREVENTS CHAR "GROWTH" DURING DOWNLOAD
21F3:8D 00 00                STA   SCRNMODE
21F6:00                      BRK                     ; SET.PREFIX(PREFIXPATH=".D1")
```

```
21F7:C6                       DFB   SETPREFIX
21F8:8C 28                    DW    PREFX.PARMS
21FA:              *
21FA:              * LAUNCH CHARACTER SET DOWNLOAD (CONSOLE) AND CLEAR SCREEN
21FA:              *
21FA:58                       CLI                    ; BEGIN CHARACTER SET DOWNLOAD (CONSOLE)
21FB:              *
21FB:A9 00                    LDA   #0               ; CLEAR TEXT SCREENS
21FD:8D 23 16                 STA   CXPAGE+SRC.P+1
2200:8D 25 16                 STA   CXPAGE+DST.P+1
2203:A9 04                    LDA   #$04
2205:85 23                    STA   SRC.P+1
2207:85 25                    STA   DST.P+1
2209:A9 00                    LDA   #$00
220B:85 22                    STA   SRC.P
220D:A9 80                    LDA   #$80
220F:85 24                    STA   DST.P
2211:A9 A0                    LDA   #$A0
2213:A2 08                    LDX   #8
2215:A0 77         CLEAR0     LDY   #$77
2217:91 22         CLEAR1     STA   (SRC.P),Y
2219:91 24                    STA   (DST.P),Y
221B:88                       DEY
221C:10 F9   2217             BPL   CLEAR1
221E:E6 23                    INC   SRC.P+1          ; NEXT PAGE
2220:E6 25                    INC   DST.P+1          ; NEXT PAGE
2222:CA                       DEX
2223:D0 F0   2215             BNE   CLEAR0
2225:              *
2225:E6 22         WAIT       INC   SRC.P            ; WAIT FOR DOWNLOAD TO COMPLETE
2227:D0 FC   2225             BNE   WAIT
2229:E8                       INX
222A:D0 F9   2225             BNE   WAIT
222C:              *
222C:A9 80                    LDA   #$80             ; TURN VIDEO ON
222E:8D 00 00                 STA   SCRNMODE
2231:60                       RTS
2232:
****************************************************************************************************
2232:                         CHN   SOSLDR.E.SRC
```

```
2232:
****************************************************************************************************
2232:                  *
2232:                  * SET.DRIVES ( IN:   A=# DRIVES
2232:                  *              IN:   DIB1..4    )
2232:                  * (INITIALIZES DIB LINKS IN KERNEL'S FLOPPY DRIVER)
2232:
****************************************************************************************************
2232:                  *
2232:        2232      SET.DRIVES EQU   *
2232:A8                           TAY                    ; SAVE # OF DRIVES
2233:A9 00                        LDA   #>DIB2           ; DIB1:=ADR(DIB2)
2235:8D 00 00                     STA   DIB1
2238:A9 00                        LDA   #<DIB2
223A:8D 01 00                     STA   DIB1+1
223D:A9 00                        LDA   #>DIB3           ; DIB2:=ADR(DIB3)
223F:8D 00 00                     STA   DIB2
2242:A9 00                        LDA   #<DIB3
2244:8D 01 00                     STA   DIB2+1
2247:A9 00                        LDA   #>DIB4           ; DIB3:=ADR(DIB4)
2249:8D 00 00                     STA   DIB3
224C:A9 00                        LDA   #<DIB4
224E:8D 01 00                     STA   DIB3+1
2251:                  *
2251:A9 00                        LDA   #0               ; CASE (Y=# OF DRIVES)
2253:C0 02                        CPY   #2
2255:90 08   225F                 BCC   STDR010
2257:F0 0D   2266                 BEQ   STDR020
2259:C0 04                        CPY   #4
225B:90 10   226D                 BCC   STDR030
225D:B0 15   2274                 BCS   STDR040
225F:                  *
225F:8D 00 00          STDR010    STA   DIB1             ;    1:  DIB1:=0
2262:8D 01 00                     STA   DIB1+1
2265:60                           RTS
2266:                  *
2266:8D 00 00          STDR020    STA   DIB2             ;    2:  DIB2:=0
2269:8D 01 00                     STA   DIB2+1
226C:60                           RTS
226D:                  *
226D:8D 00 00          STDR030    STA   DIB3             ;    3:  DIB3:=0
2270:8D 01 00                     STA   DIB3+1
2273:60                           RTS
2274:                  *
2274:8D 00 00          STDR040    STA   DIB4             ;    4:  DIB4:=0
2277:8D 01 00                     STA   DIB4+1
227A:60                           RTS                    ; RETURN
```

```
227B:
****************************************************************************************************
227B:               *
227B:               * INIT.KRNL ()
227B:               *
227B:               * (CALLS KERNEL INITIALIZATION MODULES)
227B:
****************************************************************************************************
227B:               *
227B:        227B   INIT.KRNL  EQU   *
227B:AD DF FF                  LDA   E.REG            ; SWITCH IN I/O BANK AND SELECT PRIMARY STACK
227E:09 44                     ORA   #$44             ; E:=( 0.1.1.X:0.1.0.0 )
2280:8D DF FF                  STA   E.REG            ;    ( 1.I.S.R:W.P.R.R )
2283:               *
2283:A9 18                     LDA   #<SZPAGE         ; SWITCH TO SOS ZPAGE
2285:8D D0 FF                  STA   Z.REG
2288:               *
2288:20 00 00                  JSR   INT.INIT         ; CALL KERNEL INITIALIZATION ROUTINES
228B:20 00 00                  JSR   EVQ.INIT
228E:20 00 00                  JSR   BFM.INIT2
2291:B0 20   22B3              BCS   INITK.ERR
2293:20 00 00                  JSR   DMGR.INIT
2296:20 00 00                  JSR   CFMGR.INIT
2299:20 00 00                  JSR   MMGR.INIT
229C:20 00 00                  JSR   BMGR.INIT
229F:20 00 00                  JSR   BFM.INIT
22A2:20 00 00                  JSR   CLK.INIT
22A5:               *
22A5:AD DF FF                  LDA   E.REG            ; SWITCH OUT I/O BANK AND RETURN TO ALTERNATE STACK
22A8:29 BB                     AND   #$BB             ; E:=( 0.0.1.X:0.0.0.0 )
22AA:8D DF FF                  STA   E.REG            ;    ( 1.I.S.R:W.P.R.R )
22AD:               *
22AD:A9 1A                     LDA   #<CZPAGE         ; SWITCH BACK TO USER ZPAGE
22AF:8D D0 FF                  STA   Z.REG
22B2:               *
22B2:60                        RTS                    ; RETURN
22B3:               *
22B3:               *
22B3:A2 08          INITK.ERR  LDX   #ERR0X           ; ERROR("I/O ERROR")
22B5:A0 09                     LDY   #ERR0L
22B7:4C E2 25                  JMP   ERROR
```

```
22BA:
****************************************************************************************************
22BA:                  *
22BA:                  * ADVANCE ( I/O:  WORK.P
22BA:                  *           OUT:  SRC.P
22BA:                  *           OUT:  DST.P
22BA:                  *           OUT:  CNT     )
22BA:                  * (ADVANCES WORK.P TO NEXT INTERP.KERNEL MODULE.  INITS SRC.P, DST.P, CNT FOR MOVE)
22BA:
****************************************************************************************************
22BA:                  *
22BA:        22BA      ADVANCE    EQU   *
22BA:18                           CLC
22BB:A0 02                        LDY   #2               ; Y:=0
22BD:A5 0A                        LDA   WORK.P           ; WORK.P:=WORK.P+(WORK.P),Y + 4
22BF:71 0A                        ADC   (WORK.P),Y
22C1:AA                           TAX
22C2:C8                           INY
22C3:A5 0B                        LDA   WORK.P+1
22C5:71 0A                        ADC   (WORK.P),Y
22C7:48                           PHA
22C8:8A                           TXA
22C9:69 04                        ADC   #4
22CB:85 0A                        STA   WORK.P
22CD:68                           PLA
22CE:69 00                        ADC   #0
22D0:85 0B                        STA   WORK.P+1
22D2:18                           CLC                    ; SRC.P:=X:WORK.P+4
22D3:A5 0A                        LDA   WORK.P
22D5:69 04                        ADC   #>$0004
22D7:85 22                        STA   SRC.P
22D9:A5 0B                        LDA   WORK.P+1
22DB:69 00                        ADC   #<$0004
22DD:85 23                        STA   SRC.P+1
22DF:AD 0B 16                     LDA   CXPAGE+WORK.P+1
22E2:8D 23 16                     STA   CXPAGE+SRC.P+1
22E5:A0 00                        LDY   #0               ; DST.P:=0:(WORK.P)
22E7:8C 25 16                     STY   CXPAGE+DST.P+1
22EA:B1 0A                        LDA   (WORK.P),Y
22EC:85 24                        STA   DST.P
22EE:C8                           INY
22EF:B1 0A                        LDA   (WORK.P),Y
22F1:85 25                        STA   DST.P+1
22F3:C8                           INY                    ; Y:=2
22F4:B1 0A                        LDA   (WORK.P),Y       ; CNT:=(WORK.P),Y
22F6:85 26                        STA   CNT
22F8:C8                           INY
22F9:B1 0A                        LDA   (WORK.P),Y
22FB:85 27                        STA   CNT+1
22FD:60                           RTS                    ; RETURN
```

```
22FE:
****************************************************************************************************
22FE:              *
22FE:              * REVERSE ( IN:   D.HDR.CNT
22FE:              *           IN:   SDT.SIZE = CONSTANT
22FE:              *           I/O:  DRIVER FILE,
22FE:              *           OUT:  WORK.P      )      )
22FE:              *
22FE:              *            LOCAL:  REV.SAVE, REV.TEMP
22FE:              * (REVERSES TITLE/CODE/RELOC COUNTS TO ALLOW DRIVER FILE TO BE PROCESSED FROM BACK TO FRONT)
22FE:
****************************************************************************************************
22FE:        22FE  REVERSE    EQU   *
22FE:A9 00                    LDA   #>D.HDR.CNT      ; WORK.P:=80:D.HDR.CNT
2300:85 0A                    STA   WORK.P
2302:A9 0F                    LDA   #<D.HDR.CNT
2304:85 0B                    STA   WORK.P+1
2306:A9 80                    LDA   #$80
2308:8D 0B 16                 STA   CXPAGE+WORK.P+1
230B:18                       CLC                    ; WORK.P:=WORK.P+(WORK.P)+2
230C:A0 00                    LDY   #0
230E:A5 0A                    LDA   WORK.P
2310:71 0A                    ADC   (WORK.P),Y
2312:AA                       TAX
2313:C8                       INY
2314:A5 0B                    LDA   WORK.P+1
2316:71 0A                    ADC   (WORK.P),Y
2318:48                       PHA
2319:8A                       TXA
231A:69 02                    ADC   #2
231C:85 0A                    STA   WORK.P
231E:68                       PLA
231F:69 00                    ADC   #0
2321:85 0B                    STA   WORK.P+1
2323:B1 0A                    LDA   (WORK.P),Y       ; IF (WORK.P)=$FFFF
2325:88                       DEY
2326:31 0A                    AND   (WORK.P),Y       ;    THEN
2328:C9 FF                    CMP   #$FF
232A:D0 07   2333             BNE   REV010
232C:A2 EB                    LDX   #ERR10X          ;       ERROR("EMPTY DRIVER FILE")
232E:A0 11                    LDY   #ERR10L
2330:20 E2 25                 JSR   ERROR
2333:A9 FF         REV010     LDA   #$FF
2335:85 0C                    STA   REV.SAVE
2337:85 0D                    STA   REV.SAVE+1
2339:              *
2339:A5 0C         REV020     LDA   REV.SAVE         ;R1: STACK:=REV.SAVE
233B:48                       PHA
233C:A5 0D                    LDA   REV.SAVE+1
233E:48                       PHA
233F:A0 00                    LDY   #0               ;    REV.SAVE:=(WORK.P)
2341:B1 0A                    LDA   (WORK.P),Y
2343:85 0C                    STA   REV.SAVE
2345:C8                       INY
2346:B1 0A                    LDA   (WORK.P),Y
2348:85 0D                    STA   REV.SAVE+1
234A:68                       PLA                    ;    (WORK.P):=STACK
234B:91 0A                    STA   (WORK.P),Y
```

```
234D:88                       DEY
234E:68                       PLA
234F:91 0A                    STA   (WORK.P),Y
2351:A5 0C                    LDA   REV.SAVE         ;    IF REV.SAVE = $FFFF THEN EXIT
2353:25 0D                    AND   REV.SAVE+1
2355:C9 FF                    CMP   #$FF
2357:F0 24   237D             BEQ   REV.EXIT
2359:24 0D         REV030     BIT   REV.SAVE+1       ;    IF REV.SAVE >= $8000 THEN ERROR
235B:30 19   2376             BMI   REV040
235D:18                       CLC                    ;    WORK.P:=WORK.P+REV.SAVE+2
235E:A5 0A                    LDA   WORK.P
2360:65 0C                    ADC   REV.SAVE
2362:AA                       TAX
2363:A5 0B                    LDA   WORK.P+1
2365:65 0D                    ADC   REV.SAVE+1
2367:48                       PHA
2368:B0 0C   2376             BCS   REV040
236A:8A                       TXA
236B:69 02                    ADC   #2
236D:85 0A                    STA   WORK.P
236F:68                       PLA
2370:69 00                    ADC   #0
2372:85 0B                    STA   WORK.P+1
2374:90 C3   2339             BCC   REV020           ;    IF C=FALSE THEN R1
2376:A2 7A         REV040     LDX   #ERR5X           ;                ELSE ERROR("INVALID DRIVER FILE")
2378:A0 13                    LDY   #ERR5L
237A:20 E2 25                 JSR   ERROR
237D:              *
237D:60            REV.EXIT   RTS                    ; RETURN
```

```
237E:
****************************************************************************************************
237E:              *
237E:              * DADVANCE ( I/O:  WORK.P
237E:              *            OUT:  C="NO DRIVERS LEFT"
237E:              *            OUT:  SRC.P
237E:              *            OUT:  CNT
237E:              *            OUT:  REL.P )
237E:              * (ADVANCES WORK.P TO NEXT DRIVER MODULE.  INITS SRC.P, CNT, REL.P FOR RELOCATION AND MOVE)
237E:
****************************************************************************************************
237E:        237E  DADVANCE   EQU   *
237E:A0 00                    LDY   #0               ; IF (WORK.P)=$FFFF THEN EXIT "NO DRIVERS LEFT IN FILE"
2380:B1 0A                    LDA   (WORK.P),Y
2382:C8                       INY
2383:31 0A                    AND   (WORK.P),Y
2385:C9 FF                    CMP   #$FF
2387:D0 02   238B             BNE   DADV010
2389:38                       SEC                    ; C:="NO DRIVERS LEFT"
238A:60                       RTS                    ; RETURN
238B:              *
238B:              *
238B:A5 0A         DADV010    LDA   WORK.P           ; REL.P:=X:WORK.P
238D:85 1E                    STA   REL.P
238F:A5 0B                    LDA   WORK.P+1
2391:85 1F                    STA   REL.P+1
2393:AD 0B 16                 LDA   CXPAGE+WORK.P+1
2396:8D 1F 16                 STA   CXPAGE+REL.P+1
2399:              *
2399:20 C2 23                 JSR   DADD             ; ADVANCE TO CODE COUNT FIELD
239C:              *
239C:A0 00                    LDY   #0               ; CNT:=(WORK.P)
239E:B1 0A                    LDA   (WORK.P),Y
23A0:85 26                    STA   CNT
23A2:C8                       INY
23A3:B1 0A                    LDA   (WORK.P),Y
23A5:85 27                    STA   CNT+1
23A7:              *
23A7:20 C2 23                 JSR   DADD             ; ADVANCE TO TITLE CNT FIELD
23AA:              *
23AA:18                       CLC                    ; SRC.P:=X:WORK.P+2
23AB:A5 0A                    LDA   WORK.P
23AD:69 02                    ADC   #2
23AF:85 22                    STA   SRC.P
23B1:A5 0B                    LDA   WORK.P+1
23B3:69 00                    ADC   #0
23B5:85 23                    STA   SRC.P+1
23B7:AD 0B 16                 LDA   CXPAGE+WORK.P+1
23BA:8D 23 16                 STA   CXPAGE+SRC.P+1
23BD:              *
23BD:20 C2 23                 JSR   DADD             ; ADVANCE TO RELOC FIELD OF NEXT DRIVER
23C0:18                       CLC                    ; C:="DRIVERS LEFT"
23C1:60                       RTS                    ; RETURN
```

```
23C2:
****************************************************************************************************
23C2:              *
23C2:              * DADD ( I/O:   WORK.P )
23C2:              *
23C2:              * (ADVANCES WORK.P TO NEXT FIELD IN DRIVER MODULE)
23C2:
****************************************************************************************************
23C2:        23C2  DADD       EQU   *
23C2:38                       SEC                    ; WORK.P:=WORK.P-(WORK.P)-2
23C3:A0 00                    LDY   #0
23C5:A5 0A                    LDA   WORK.P
23C7:F1 0A                    SBC   (WORK.P),Y
23C9:AA                       TAX
23CA:C8                       INY
23CB:A5 0B                    LDA   WORK.P+1
23CD:F1 0A                    SBC   (WORK.P),Y
23CF:48                       PHA
23D0:8A                       TXA
23D1:E9 02                    SBC   #2
23D3:85 0A                    STA   WORK.P
23D5:68                       PLA
23D6:E9 00                    SBC   #0
23D8:85 0B                    STA   WORK.P+1
23DA:60                       RTS                    ; RETURN
```

```
23DB:              299
****************************************************************************************************
23DB:              300 *
23DB:              301 * FLAGS ( IN:   SRC.P
23DB:              302 *         OUT:  PG.ALIGN
23DB:              303 *         OUT:  FIRST.ADIB
23DB:              304 *         OUT:  OV="ALL DIBS INACTIVE" )
23DB:              305 *
23DB:              306 *         LOCAL:  PREV.ADIB.P, DIB.P
23DB:              307 * (PROCESSES "INACTIVE" & "PAGE ALIGN" FLAGS IN DRIVER MODULE'S DIBS"
23DB:              308
****************************************************************************************************
23DB:        23DB  309 FLAGS      EQU   *
23DB:38            310            SEC                     ; C="FIRST DIB"
23DC:20 36 24      311 FLAG010    JSR   NEXT.DIB          ; NEXT.DIB(SRC.P.IN, DIB.P PG.ALIGN C OV.OUT)
23DF:50 03   23E4  312            BVC   FLAG015           ; IF OV <> "INACTIVE" THEN ACTIVE DIB FOUND
23E1:90 F9   23DC  313            BCC   FLAG010           ; IF C <> "LAST DIB" THEN CHECK NEXT DIB
23E3:60            314            RTS                     ; RETURN  (OV:="ALL DIBS INACTIVE")
23E4:              315 *
23E4:08            316 FLAG015    PHP                     ; PUSH STATUS
23E5:38            317            SEC                     ; FIRST.ADIB:=DIB.P-SRC.P
23E6:A5 14         318            LDA   DIB.P
23E8:E5 22         319            SBC   SRC.P
23EA:85 10         320            STA   FIRST.ADIB
23EC:A5 15         321            LDA   DIB.P+1
23EE:E5 23         322            SBC   SRC.P+1
23F0:85 11         323            STA   FIRST.ADIB+1
23F2:A5 14         324            LDA   DIB.P             ; PREV.ADIB.P:=X:DIB.P
23F4:85 12         325            STA   PREV.ADIB.P
23F6:A5 15         326            LDA   DIB.P+1
23F8:85 13         327            STA   PREV.ADIB.P+1
23FA:AD 15 16      328            LDA   CXPAGE+DIB.P+1
23FD:8D 13 16      329            STA   CXPAGE+PREV.ADIB.P+1
2400:28            330            PLP                     ; PULL STATUS
2401:B0 31   2434  331            BCS   FLAG100           ; IF C="LAST DIB" THEN EXIT
2403:              332 *
2403:20 36 24      333 FLAG020    JSR   NEXT.DIB          ; NEXT.DIB(SRC.P.IN, DIB.P PG.ALIGN C OV.OUT)
2406:08            334            PHP                     ; PUSH STATUS
2407:A0 00         335            LDY   #0                ; IF OV="INACTIVE DIB"
2409:50 11   241C  336            BVC   FLAG025
240B:38            337            SEC                     ;    THEN
240C:A5 12         338            LDA   PREV.ADIB.P       ;       (PREV.ADIB.P):=PREV.ADIB.P-SRC.P
240E:E5 22         339            SBC   SRC.P
2410:91 12         340            STA   (PREV.ADIB.P),Y
2412:C8            341            INY
2413:A5 13         342            LDA   PREV.ADIB.P+1
2415:E5 23         343            SBC   SRC.P+1
2417:91 12         344            STA   (PREV.ADIB.P),Y
2419:4C 31 24      345            JMP   FLAG050
241C:              346 *
241C:38            347 FLAG025    SEC                     ;    ELSE
241D:A5 14         348            LDA   DIB.P             ;       (PREV.ADIB.P):=DIB.P-SRC.P
241F:E5 22         349            SBC   SRC.P
2421:91 12         350            STA   (PREV.ADIB.P),Y
2423:C8            351            INY
2424:A5 15         352            LDA   DIB.P+1
2426:AA            353            TAX
2427:E5 23         354            SBC   SRC.P+1
```

```
2429:91 12                    STA   (PREV.ADIB.P),Y
242B:86 13                    STX   PREV.ADIB.P+1    ;       PREV.ADIB.P:=DIB.P
242D:A5 14                    LDA   DIB.P
242F:85 12                    STA   PREV.ADIB.P
2431:28             FLAG050   PLP                    ; PULL STATUS
2432:90 CF   2403             BCC   FLAG020          ; IF C <> "LAST DIB" THEN PROCESS NEXT DIB
2434:               *
2434:B8             FLAG100   CLV                    ; OV:="ACTIVE DIBS"
2435:60                       RTS                    ; RETURN
```

```
2436:              
****************************************************************************************************
2436:              *
2436:              * NEXT.DIB ( IN:   C="FIRST DIB"
2436:              *            IN:   SRC.P
2436:              *            OUT:  DIB.P
2436:              *            OUT:  PG.ALIGN
2436:              *            OUT:  C="LAST DIB"
2436:              *            OUT:  OV="INACTIVE DIB" )
2436:              *
2436:              *            LOCAL:  DIB.FLAGS, DIB.DCB = CONSTANT
2436:              * (ADVANCES TO NEXT DIB IN DRIVER MODULE)
2436:              
****************************************************************************************************
2436:        2436  NEXT.DIB   EQU   *
2436:A0 00                    LDY   #0
2438:90 15   244F             BCC   NXTD010          ; IF C = "FIRST DIB"
243A:84 16                    STY   PG.ALIGN         ;    THEN
243C:84 17                    STY   PG.ALIGN+1       ;       PG.ALIGN:=0
243E:A5 22                    LDA   SRC.P            ;       DIB.P:=X:SRC.P
2440:85 14                    STA   DIB.P
2442:A5 23                    LDA   SRC.P+1
2444:85 15                    STA   DIB.P+1
2446:AD 23 16                 LDA   CXPAGE+SRC.P+1
2449:8D 15 16                 STA   CXPAGE+DIB.P+1
244C:4C 5D 24                 JMP   NXTD020
244F:A5 22         NXTD010    LDA   SRC.P            ;    ELSE
2451:71 14                    ADC   (DIB.P),Y        ;       DIB.P:=SRC.P+(DIB.P)
2453:AA                       TAX
2454:C8                       INY
2455:A5 23                    LDA   SRC.P+1
2457:71 14                    ADC   (DIB.P),Y
2459:85 15                    STA   DIB.P+1
245B:86 14                    STX   DIB.P
245D:              *
245D:A0 14         NXTD020    LDY   #DIB.FLAGS       ; IF (DIB.P),DIB.FLAGS.BIT7 = "INACTIVE"
245F:B1 14                    LDA   (DIB.P),Y
2461:30 05   2468             BMI   NXTD030
2463:2C 8B 24                 BIT   NXTD999          ;    THEN
2466:70 16   247E             BVS   NXTD040          ;       OV:="INACTIVE"
2468:              *                                            ELSE
2468:29 40         NXTD030    AND   #$40             ;       IF (DIB.P),DIB.FLAGS.BIT6 = "PAGE ALIGN"
246A:F0 12   247E             BEQ   NXTD040
246C:18                       CLC                    ;          THEN
246D:A9 22                    LDA   #DIB.DCB+2       ;             PAGE.ALIGN:=DIB.DCB+2+(SRC.P),DIB.DCB
246F:A8                       TAY
2470:88                       DEY
2471:88                       DEY
2472:71 22                    ADC   (SRC.P),Y
2474:85 16                    STA   PG.ALIGN
2476:C8                       INY
2477:A9 00                    LDA   #0
2479:71 22                    ADC   (SRC.P),Y
247B:85 17                    STA   PG.ALIGN+1
247D:B8                       CLV                    ;       OV:="ACTIVE"
247E:              *
247E:A0 00         NXTD040    LDY   #0               ; IF (DIB.P) = 0
2480:B1 14                    LDA   (DIB.P),Y
```

```
2482:C8                            INY
2483:11 14                         ORA   (DIB.P),Y
2485:D0 03   248A                  BNE   NXTD998
2487:38                            SEC                     ;    THEN  C:="LAST DIB"
2488:B0 01   248B                  BCS   NXTD999
248A:18                 NXTD998    CLC                     ;    ELSE  C:=NOT "LAST DIB"
248B:60                 NXTD999    RTS                     ; RETURN
248C:
****************************************************************************************************
248C:                              CHN   SOSLDR.F.SRC
```

```
248C:
****************************************************************************************************
248C:                  *
248C:                  * GETMEM ( IN:   PG.ALIGN
248C:                  *          IN:   CNT
248C:                  *          I/O:  DST.P
248C:                  *          I/O:  DSTBANK
248C:                  *          I/O:  DSEGLIST
248C:                  *          OUT:  PREVBANK )
248C:                  *
248C:                  *          LOCAL:  PREVDST
248C:                  * (COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT)
248C:
****************************************************************************************************
248C:        248C      GETMEM     EQU   *
248C:A5 2A                        LDA   DSTBANK          ; PREVBANK:=DSTBANK
248E:85 18                        STA   PREVBANK
2490:A5 24                        LDA   DST.P            ; PREVDST:=DST.P
2492:85 19                        STA   PREVDST
2494:A5 25                        LDA   DST.P+1
2496:85 1A                        STA   PREVDST+1
2498:20 C4 24                     JSR   NEWDST           ; NEWDST(PG.ALIGN.IN, PREVDST.IN, CNT.IN, DST.P.OUT)
249B:                  *
249B:A5 25                        LDA   DST.P+1          ; IF DST.P >= $2000
249D:C9 20                        CMP   #$20
249F:90 0C   24AD                 BCC   GETM010
24A1:38                           SEC                    ;    THEN
24A2:A5 1A                        LDA   PREVDST+1        ;       A=PAGES:=PREVDST-DST.P
24A4:E5 25                        SBC   DST.P+1
24A6:18                           CLC
24A7:20 FD 24                     JSR   BUILD.DSEG       ;       BUILD.DSEG(C="NEXT BANK".IN, A=PAGES.IN, DSEGLIST.IO)
24AA:4C C3 24                     JMP   GETM.EXIT
24AD:                  *                                             ELSE
24AD:C6 2A             GETM010    DEC   DSTBANK          ;       DSTBANK:=DSTBANK-1
24AF:A9 00                        LDA   #>$A000          ;       PREVDST:=$A000
24B1:85 19                        STA   PREVDST
24B3:A9 A0                        LDA   #<$A000
24B5:85 1A                        STA   PREVDST+1
24B7:20 C4 24                     JSR   NEWDST           ;       NEWDST(PG.ALIGN.IN, PREVDST.IN, CNT.IN, DST.P.OUT)
24BA:38                           SEC                    ;       A="PAGES":=PREVDST-DST.P
24BB:A5 1A                        LDA   PREVDST+1
24BD:E5 25                        SBC   DST.P+1
24BF:38                           SEC
24C0:20 FD 24                     JSR   BUILD.DSEG       ;       BUILD.DSEG(C="NEXTBANK".IN, A="PAGES".IN, DSEGLIST.IO)
24C3:                  *
24C3:60                GETM.EXIT  RTS                    ; RETURN
```

```
24C4:
****************************************************************************************************
24C4:               *
24C4:               * NEWDST ( IN:   PG.ALIGN
24C4:               *          IN:   PREVDST
24C4:               *          IN:   CNT
24C4:               *          I/O:  DST.P      )
24C4:               * (COMPUTES DESTINATION BASE ADDRESS, ALIGNING ON PAGE BOUNDARY IF REQUESTED)
24C4:
****************************************************************************************************
24C4:        24C4   NEWDST     EQU   *
24C4:38                        SEC                    ; IF (PREVDST-$2000) < CNT
24C5:A5 19                     LDA   PREVDST
24C7:E9 00                     SBC   #>$2000
24C9:AA                        TAX
24CA:A5 1A                     LDA   PREVDST+1
24CC:E9 20                     SBC   #<$2000
24CE:E4 26                     CPX   CNT
24D0:E5 27                     SBC   CNT+1
24D2:B0 08   24DC              BCS   NEWD010
24D4:A9 00                     LDA   #0               ;    THEN
24D6:85 24                     STA   DST.P            ;      DST.P:=0
24D8:85 25                     STA   DST.P+1
24DA:F0 20   24FC              BEQ   NEWD.EXIT
24DC:38             NEWD010    SEC                    ;    ELSE
24DD:A5 19                     LDA   PREVDST          ;       DST.P:=PREVDST-CNT
24DF:E5 26                     SBC   CNT
24E1:85 24                     STA   DST.P
24E3:A5 1A                     LDA   PREVDST+1
24E5:E5 27                     SBC   CNT+1
24E7:85 25                     STA   DST.P+1
24E9:A5 16                     LDA   PG.ALIGN         ;       IF PG.ALIGN <> 0
24EB:05 17                     ORA   PG.ALIGN+1       ;          THEN
24ED:F0 0D   24FC              BEQ   NEWD.EXIT
24EF:38                        SEC                    ;              DST.P:=(DST.P/256*256)-PG.ALIGN
24F0:A9 00                     LDA   #0
24F2:E5 16                     SBC   PG.ALIGN
24F4:85 24                     STA   DST.P
24F6:A5 25                     LDA   DST.P+1
24F8:E5 17                     SBC   PG.ALIGN+1
24FA:85 25                     STA   DST.P+1
24FC:60             NEWD.EXIT  RTS                    ; RETURN
```

```
24FD:
****************************************************************************************************
24FD:                  *
24FD:                  * BUILD.DSEG ( IN:   C="NEXTBANK"
24FD:                  *              IN:   A="PAGES"
24FD:                  *              I/O:  DSEGLIST     )
24FD:                  * (COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT)
24FD:
****************************************************************************************************
24FD:        24FD      BUILD.DSEG EQU   *
24FD:48                           PHA
24FE:B0 05   2505                 BCS   BLDS010          ; IF ("NEXTBANK"=TRUE OR DSEGX=$FF)
2500:AD 14 25                     LDA   DSEGX            ;    THEN
2503:10 03   2508                 BPL   BLDS020
2505:EE 14 25          BLDS010    INC   DSEGX            ;       DSEGX:=DSEGX+1
2508:AE 14 25          BLDS020    LDX   DSEGX
250B:18                           CLC                    ; DSEGLIST(DSEGX):=DSEGLIST(DSEGX)+"PAGES"
250C:68                           PLA
250D:7D 15 25                     ADC   DSEGLIST,X
2510:9D 15 25                     STA   DSEGLIST,X
2513:60                           RTS                    ; RETURN
2514:                  *
2514:                  *
2514:                  *
2514:FF                DSEGX      DFB   $FF              ; DRIVER SEGMENT LIST TABLE
2515:00                DSEGLIST   DFB   $0               ; # PAGES FOR 1ST DRIVER SEGMENT   (BANK N  )
2516:00                           DFB   $0               ;        "    2ND       "          (BANK N-1)
2517:00                           DFB   $0               ;        "    3RD       "          (BANK N-2)
2518:00                           DFB   $0               ;        "    4TH       "          (BANK N-3)
```

```
2519:              
****************************************************************************************************
2519:              *
2519:              * RELOC ( IN:   SRC.P
2519:              *         IN:   REL.P
2519:              *         IN:   DST.P
2519:              *         OUT:  RELOCATED DRIVER MODULE )
2519:              *
2519:              *         LOCAL:  REL.END, CODE.P
2519:              * (RELOCATES DRIVER MODULE'S CODE FIELD USING RELOCATION FIELD)
2519:              
****************************************************************************************************
2519:        2519  RELOC      EQU   *
2519:38                       SEC                    ; REL.END:=REL.P-(REL.P)
251A:A0 00                    LDY   #0
251C:A5 1E                    LDA   REL.P
251E:F1 1E                    SBC   (REL.P),Y
2520:85 20                    STA   REL.END
2522:C8                       INY
2523:A5 1F                    LDA   REL.P+1
2525:F1 1E                    SBC   (REL.P),Y
2527:85 21                    STA   REL.END+1
2529:38            REL.LOOP   SEC                    ; REL.P:=REL.P-2
252A:A5 1E                    LDA   REL.P
252C:E9 02                    SBC   #2
252E:85 1E                    STA   REL.P
2530:A5 1F                    LDA   REL.P+1
2532:E9 00                    SBC   #0
2534:85 1F                    STA   REL.P+1
2536:A5 1E                    LDA   REL.P            ; IF REL.P < REL.END THEN EXIT
2538:C5 20                    CMP   REL.END
253A:A5 1F                    LDA   REL.P+1
253C:E5 21                    SBC   REL.END+1
253E:90 29   2569             BCC   REL.EXIT
2540:A0 00                    LDY   #0               ; CODE.P:=X:SRC.P+(REL.P)
2542:18                       CLC
2543:A5 22                    LDA   SRC.P
2545:71 1E                    ADC   (REL.P),Y
2547:85 1C                    STA   CODE.P
2549:C8                       INY
254A:A5 23                    LDA   SRC.P+1
254C:71 1E                    ADC   (REL.P),Y
254E:85 1D                    STA   CODE.P+1
2550:AD 23 16                 LDA   CXPAGE+SRC.P+1
2553:8D 1D 16                 STA   CXPAGE+CODE.P+1
2556:A0 00                    LDY   #0               ; (CODE.P):=(CODE.P)+DST.P
2558:18                       CLC
2559:B1 1C                    LDA   (CODE.P),Y
255B:65 24                    ADC   DST.P
255D:91 1C                    STA   (CODE.P),Y
255F:C8                       INY
2560:B1 1C                    LDA   (CODE.P),Y
2562:65 25                    ADC   DST.P+1
2564:91 1C                    STA   (CODE.P),Y
2566:4C 29 25                 JMP   REL.LOOP         ; GOTO REL.LOOP
2569:              *
2569:60            REL.EXIT   RTS                    ; RETURN
```

```
256A:
****************************************************************************************************
256A:              *
256A:              * ALLOC.SEG ( IN:   K.BASE
256A:              *             IN:   I.BASE.P
256A:              *             IN:   SYSBANK )
256A:              *         I.BASE.P
256A:              *         D.BASE.PG
256A:              * (ALLOCATES SEGMENTS FOR KERNEL, INTERPRETER AND SYSTEM WORK AREA)
256A:
****************************************************************************************************
256A:        256A  ALLOC.SEG  EQU   *
256A:00                       BRK                     ; REQ.SEG(BASE=(F,0), LIMIT=(F,1D), SEGID=0, SEGNUM)
256B:40                       DFB   REQSEG
256C:85 28                    DW    SEGMENT
256E:              *
256E:A9 10                    LDA   #$10              ; SET BASE/LIMIT BANKS
2570:8D 86 28                 STA   SEGBASE
2573:8D 88 28                 STA   SEGLIM
2576:A9 00                    LDA   #0                ; AND INIT BASE PAGE
2578:8D 87 28                 STA   SEGBASE+1
257B:              *
257B:A6 01                    LDX   K.BASE+1          ; KERNEL SEGMENT, ID=1
257D:20 86 25                 JSR   RSEG
2580:              *
2580:A6 03                    LDX   I.BASE.P+1        ; INTERPRETER SEGMENT, ID=2
2582:20 86 25                 JSR   RSEG
2585:60                       RTS
```

```
2586:              199
****************************************************************************************************
2586:              200 *
2586:              201 * RSEG ( IN:  X=BASE.PAGE OF SEGMENT )
2586:              202 *
2586:              203
****************************************************************************************************
2586:        2586  204 RSEG       EQU   *
2586:EE 8A 28      205            INC   SEGID            ; SEGID:=SEGID+1
2589:AC 87 28      206            LDY   SEGBASE+1        ; LIMIT.PAGE:=BASE.PAGE-1
258C:88            207            DEY
258D:8C 89 28      208            STY   SEGLIM+1
2590:8E 87 28      209            STX   SEGBASE+1        ; BASE.PAGE:=X
2593:              210 *
2593:E0 A0         211            CPX   #$A0             ; IF BASE>=$A0 OR LIMIT<$A0 THEN
2595:B0 24   25BB  212            BCS   RSEG010          ;    THEN
2597:AD 89 28      213            LDA   SEGLIM+1         ;       REQUEST ONLY ONE SEGMENT
259A:C9 A0         214            CMP   #$A0
259C:90 1D   25BB  215            BCC   RSEG010
259E:              216 *
259E:8A            217            TXA                    ;    ELSE
259F:48            218            PHA                    ;       REQUEST TWO SEGMENTS
25A0:A2 A0         219            LDX   #$A0
25A2:8E 87 28      220            STX   SEGBASE+1
25A5:              221 *
25A5:00            222            BRK                    ;       REQ.SEG(BASE, LIMIT, SEGID, SEGNUM)
25A6:40            223            DFB   REQSEG
25A7:85 28         224            DW    SEGMENT
25A9:              225 *
25A9:68            226            PLA
25AA:8D 87 28      227            STA   SEGBASE+1
25AD:A9 9F         228            LDA   #$9F
25AF:8D 89 28      229            STA   SEGLIM+1
25B2:AD 00 00      230            LDA   SYSBANK
25B5:8D 86 28      231            STA   SEGBASE
25B8:8D 88 28      232            STA   SEGLIM
25BB:              233 *
25BB:              234 *
25BB:00            235 RSEG010    BRK                    ; REQ.SEG(BASE, LIMIT, SEGID, SEGNUM)
25BC:40            236            DFB   REQSEG
25BD:85 28         237            DW    SEGMENT
25BF:              238 *
25BF:60            239            RTS                    ; RETURN
```

```
25C0:
****************************************************************************************************
25C0:              *
25C0:              * ALLOC.DSEG ( IN:   DSEGLIST )
25C0:              *
25C0:              * (ALLOCATES SEGMENTS FOR DRIVER MODULES"
25C0:
****************************************************************************************************
25C0:        25C0  ALLOC.DSEG EQU   *
25C0:EE 14 25                 INC   DSEGX            ; DSEGX:=DSEGX+1
25C3:D0 07   25CC             BNE   ALDS010          ; IF DSEGX=0
25C5:A2 7A                    LDX   #ERR5X           ;    THEN ERROR("INVALID DRIVER FILE")
25C7:A0 13                    LDY   #ERR5L
25C9:20 E2 25                 JSR   ERROR
25CC:              *
25CC:A0 FF         ALDS010    LDY   #$FF             ; Y:=-1
25CE:C8            ALDS020    INY                    ; WHILE (Y:=Y+1) < DSEGX
25CF:CC 14 25                 CPY   DSEGX            ;    DO
25D2:B0 0D   25E1             BCS   ALDS.EXIT
25D4:B9 15 25                 LDA   DSEGLIST,Y       ;       PAGECT:=DSEGLIST(Y)
25D7:8D 7E 28                 STA   SEGPGCNT
25DA:00                       BRK                    ;       FINDSEG (SRCHMODE=0.IN, SEGID=3.IN
25DB:41                       DFB   FINDSEG          ;               PAGECT=DSEGLIST(Y)
25DC:7B 28                    DW    SEGMENT1         ;               BASE.OUT, LIMIT.OUT)
25DE:4C CE 25                 JMP   ALDS020
25E1:              *
25E1:60            ALDS.EXIT  RTS                    ; RETURN
```

```
25E2:
****************************************************************************************************
25E2:                   *
25E2:                   * ERROR (IN:  X=MESSAGE INDEX
25E2:                   *        IN:  Y=MESSAGE LENGTH
25E2:                   * (DISPLAYS ERROR MESSAGE, SOUNDS BELL AND LOOPS UNTIL CONTROL/RESET PRESSED)
25E2:
****************************************************************************************************
25E2:        25E2       ERROR      EQU   *
25E2:84 2E                         STY   ETEMP            ; CENTER MSG (Y:=LEN/2+LEN)
25E4:38                            SEC
25E5:A9 28                         LDA   #40
25E7:E5 2E                         SBC   ETEMP
25E9:4A                            LSR   A
25EA:18                            CLC
25EB:65 2E                         ADC   ETEMP
25ED:A8                            TAY
25EE:                   *
25EE:BD 05 26           PRNT010    LDA   ERR,X            ; MOVE MESSAGE TO SCREEN MEMORY
25F1:99 A7 07                      STA   EMSGADR-1,Y
25F4:CA                            DEX
25F5:88                            DEY
25F6:C6 2E                         DEC   ETEMP
25F8:D0 F4   25EE                  BNE   PRNT010
25FA:                   *
25FA:A9 73                         LDA   #$73             ; E:=( 0.1.1.1:0.0.1.1 )
25FC:8D DF FF                      STA   E.REG            ;    ( 1.I.S.R:W.P.R.S )
25FF:AD 40 C0                      LDA   $C040            ; SOUND BELL
2602:4C 02 26                      JMP   *                ; LOOP UNTIL REBOOT (CTRL/RESET)
```

```
2605:
****************************************************************************************************
2605:              *
2605:              * ERROR MESSAGES
2605:              *
2605:
****************************************************************************************************
2605:        07A8  EMSGADR    EQU   $7A8
2605:              *
2605:        2605  ERR        EQU   *
2605:49 2F 4F 20   ERR0       ASC   "I/O             ERROR"
260E:        0009  ERR0L      EQU   *-ERR0
260E:        0008  ERR0X      EQU   *-ERR-1
260E:49 4E 54 45   ERR1       ASC   "INTERPRETER     FILE NOT FOUND"
2628:        001A  ERR1L      EQU   *-ERR1
2628:        0022  ERR1X      EQU   *-ERR-1
2628:49 4E 56 41   ERR2       ASC   "INVALID         INTERPRETER FILE"
2640:        0018  ERR2L      EQU   *-ERR2
2640:        003A  ERR2X      EQU   *-ERR-1
2640:49 4E 43 4F   ERR3       ASC   "INCOMPATIBLE    INTERPRETER"
2658:        0018  ERR3L      EQU   *-ERR3
2658:        0052  ERR3X      EQU   *-ERR-1
2658:44 52 49 56   ERR4       ASC   "DRIVER          FILE NOT FOUND"
266D:        0015  ERR4L      EQU   *-ERR4
266D:        0067  ERR4X      EQU   *-ERR-1
266D:49 4E 56 41   ERR5       ASC   "INVALID         DRIVER FILE"
2680:        0013  ERR5L      EQU   *-ERR5
2680:        007A  ERR5X      EQU   *-ERR-1
2680:44 52 49 56   ERR6       ASC   "DRIVER          FILE TOO LARGE"
2695:        0015  ERR6L      EQU   *-ERR6
2695:        008F  ERR6X      EQU   *-ERR-1
2695:52 4F 4D 20   ERR7       ASC   "ROM             ERROR:  PLEASE NOTIFY YOUR DEALER"
26BA:        0025  ERR7L      EQU   *-ERR7
26BA:        00B4  ERR7X      EQU   *-ERR-1
26BA:54 4F 4F 20   ERR8       ASC   "TOO             MANY DEVICES"
26CA:        0010  ERR8L      EQU   *-ERR8
26CA:        00C4  ERR8X      EQU   *-ERR-1
26CA:54 4F 4F 20   ERR9       ASC   "TOO             MANY BLOCK DEVICES"
26E0:        0016  ERR9L      EQU   *-ERR9
26E0:        00DA  ERR9X      EQU   *-ERR-1
26E0:45 4D 50 54   ERR10      ASC   "EMPTY           DRIVER FILE"
26F1:        0011  ERR10L     EQU   *-ERR10
26F1:        00EB  ERR10X     EQU   *-ERR-1
```

```
26F1:
****************************************************************************************************
26F1:              *
26F1:              * WELCOME ()
26F1:              *
26F1:              * (PRINTS WELCOME MESSAGE - "APPLE ///", VERSION, DATE/TIME, COPYRIGHT)
26F1:
****************************************************************************************************
26F1:        26F1  WELCOME    EQU   *
26F1:              *
26F1:              *  PRINT "APPLE III" MESSAGE
26F1:              *
26F1:A0 09                    LDY   #AMSGL
26F3:B9 96 27      WAM010     LDA   AMSG-1,Y
26F6:99 B6 04                 STA   AMSGADR-1,Y
26F9:88                       DEY
26FA:D0 F7   26F3             BNE   WAM010
26FC:              *
26FC:              *  PRINT SOS VERSION MESSAGE
26FC:              *
26FC:18                       CLC
26FD:A9 28                    LDA   #40
26FF:69 00                    ADC   #>SOSVERL
2701:4A                       LSR   A
2702:AA                       TAX
2703:A0 00                    LDY   #>SOSVERL
2705:B9 FF FF      WSM010     LDA   SOSVER-1,Y
2708:09 80                    ORA   #$80
270A:9D A7 05                 STA   SMSGADR-1,X
270D:CA                       DEX
270E:88                       DEY
270F:D0 F4   2705             BNE   WSM010
2711:              *
2711:              *  PRINT DATE AND TIME MESSAGE
2711:              *
2711:00                       BRK                    ; GET.TIME(TIME.OUT)
2712:63                       DFB   GETTIME
2713:93 28                    DW    DTPARMS
2715:              *
2715:AD 9E 28                 LDA   DATETIME+8       ;SET UP WEEKDAY
2718:29 0F                    AND   #$0F
271A:F0 6F   278B             BEQ   WDM040           ;NO CLOCK
271C:85 2F                    STA   WTEMP
271E:0A                       ASL   A
271F:65 2F                    ADC   WTEMP
2721:AA                       TAX
2722:A0 03                    LDY   #3
2724:BD B4 27      WDM010     LDA   DAYNAME-1,X
2727:99 9F 27                 STA   DMSG-1,Y
272A:CA                       DEX
272B:88                       DEY
272C:D0 F6   2724             BNE   WDM010
272E:              *
272E:AD 9D 28                 LDA   DATETIME+7       ;SET UP DATE
2731:AE 9C 28                 LDX   DATETIME+6
2734:8D A6 27                 STA   DMSG+6
2737:8E A5 27                 STX   DMSG+5
273A:              *
```

```
273A:AD 9B 28                 LDA   DATETIME+5       ;SET UP MONTH
273D:29 0F                    AND   #$0F
273F:AE 9A 28                 LDX   DATETIME+4
2742:E0 31                    CPX   #$31
2744:90 02   2748             BCC   WDM020
2746:69 09                    ADC   #9
2748:85 2F         WDM020     STA   WTEMP
274A:0A                       ASL   A
274B:65 2F                    ADC   WTEMP
274D:AA                       TAX
274E:A0 03                    LDY   #3
2750:BD C9 27      WDM030     LDA   MONNAME-1,X
2753:99 A7 27                 STA   DMSG+7,Y
2756:CA                       DEX
2757:88                       DEY
2758:D0 F6   2750             BNE   WDM030
275A:              *
275A:AD 99 28                 LDA   DATETIME+3       ;SET UP YEAR
275D:AE 98 28                 LDX   DATETIME+2
2760:8D AD 27                 STA   DMSG+13
2763:8E AC 27                 STX   DMSG+12
2766:              *
2766:AD A0 28                 LDA   DATETIME+10      ;SET UP HOUR
2769:AE 9F 28                 LDX   DATETIME+09
276C:8D B1 27                 STA   DMSG+17
276F:8E B0 27                 STX   DMSG+16
2772:              *
2772:AD A2 28                 LDA   DATETIME+12      ;SET UP MINUTE
2775:AE A1 28                 LDX   DATETIME+11
2778:8D B4 27                 STA   DMSG+20
277B:8E B3 27                 STX   DMSG+19
277E:              *
277E:A0 15                    LDY   #DMSGL           ;PRINT DATE & TIME
2780:B9 9F 27      WDM050     LDA   DMSG-1,Y
2783:09 80                    ORA   #$80
2785:99 B0 06                 STA   DMSGADR-1,Y
2788:88                       DEY
2789:D0 F5   2780             BNE   WDM050
278B:              *
278B:              *  PRINT COPYRIGHT MESSAGE
278B:              *
278B:A0 28         WDM040     LDY   #CMSGL
278D:B9 ED 27      WCM010     LDA   CMSG-1,Y
2790:99 CF 07                 STA   CMSGADR-1,Y
2793:88                       DEY
2794:D0 F7   278D             BNE   WCM010
2796:60                       RTS
```

```
2797:
****************************************************************************************************
2797:                 *
2797:                 * WELCOME () - DATA DECLARATIONS
2797:                 *
2797:
****************************************************************************************************
2797:                            MSB   ON
2797:C1 D0 D0 CC          AMSG       ASC   "APPLE           ///"
27A0:        0009     AMSGL      EQU   *-AMSG
27A0:        04B7     AMSGADR    EQU   40-AMSGL/2+$4A8
27A0:                            MSB   OFF
27A0:        05A8     SMSGADR    EQU   $5A8
27A0:44 41 59 2C      DMSG       ASC   "DAY,            DD-MON-YY  HH:MM"
27B5:        0015     DMSGL      EQU   *-DMSG
27B5:        06B1     DMSGADR    EQU   40-DMSGL/2+$6A8
27B5:53 55 4E 4D      DAYNAME    ASC   "SUNMONTUEWEDTHUFRISAT"
27CA:4A 41 4E 46      MONNAME    ASC   "JANFEBMARAPRMAYJUN"
27DC:4A 55 4C 41                 ASC   "JULAUGSEPOCTNOVDEC"
27EE:                            MSB   ON
27EE:A8 C3 A9 B1      CMSG       ASC   "(C)1980,1981,1982 BY APPLE COMPUTER INC."
2816:        0028     CMSGL      EQU   *-CMSG
2816:        07D0     CMSGADR    EQU   40-CMSGL/2+$7D0
2816:                            MSB   OFF
```

```
2816:
****************************************************************************************************
2816:              *
2816:              * SOS SYSTEM CALLS (1)
2816:              *
2816:
****************************************************************************************************
2816:              * OPEN (PATHNAME.IN, REFNUM.OUT, OPENLIST.IN, OPENCNT.IN)  ** (ACCESS.IN, PAGES.IN, SYSBUF.IN)
2816:
****************************************************************************************************
2816:        00C8  OPEN       EQU   $C8
2816:              *
2816:04            OPEN.PARMS DFB   $4
2817:21 28                    DW    PATH
2819:00            OPEN.REF   DFB   $0
281A:1D 28                    DW    OPEN.LIST
281C:04                       DFB   $4
281D:00 04         OPEN.LIST  DFB   $0,$4            ; PAGES:=4
281F:06 00                    DW    SYSBUF.P
2821:        0040  PATH       DS    $40              ; PATHNAME BUFFER
2861:53 4F 53 20   I.LABEL    ASC   "SOS             NTRP" ; FILE LABELS
2869:53 4F 53 20   D.LABEL    ASC   "SOS             DRVR"
2871:
****************************************************************************************************
2871:              * READ (REFNUM.IN, BUFFER.IN, BYTES.IN, BYTESREAD.OUT)
2871:
****************************************************************************************************
2871:        00CA  READ       EQU   $CA
2871:              *
2871:04            READ.PARMS DFB   $4
2872:00            READ.REF   DFB   $0
2873:04 00         READ.BUF   DW    RDBUF.P
2875:08 F1         READ.BYT   DW    $FFFF-FILE+1
2877:00 00         READ.BYTRD DW    $0
2879:
****************************************************************************************************
2879:              * CLOSE (REFNUM.IN)
2879:
****************************************************************************************************
2879:        00CC  CLOSE      EQU   $CC
2879:              *
2879:01            CLOSE.PARMS DFB  $1
287A:00            CLOSE.REF  DFB   $0
287B:
****************************************************************************************************
287B:              * FIND.SEG (SRCHMODE.IN, PAGES.IN, SEGID.IN, BASE.OUT, LIMIT.OUT, SEGNUM.OUT)
287B:
****************************************************************************************************
287B:        0041  FINDSEG    EQU   $41
287B:              *
287B:06            SEGMENT1   DFB   $6               ; FIND.SEG(SRCHMODE, SEGID, PAGECT, BASE, LIMIT, SEGNUM)
287C:00 03         SEGSRCH    DFB   $0,$3
287E:00 00         SEGPGCNT   DW    $0000
2880:00 00                    DW    $0
2882:00 00                    DW    $0
2884:00                       DFB   $0
```

```
2885:
****************************************************************************************************
2885:              *
2885:              * SOS SYSTEM CALLS (2)
2885:              *
2885:
****************************************************************************************************
2885:
****************************************************************************************************
2885:              * REQUEST.SEG (BASE.IN, LIMIT.IN, SEGID.IN, SEGNUM.OUT)
2885:
****************************************************************************************************
2885:        0040  REQSEG     EQU   $40
2885:              *
2885:04            SEGMENT    DFB   $4               ; REQUEST SEG PARM LIST
2886:0F 00         SEGBASE    DFB   $F,$0
2888:0F 1D         SEGLIM     DFB   $F,$1D
288A:00 00         SEGID      DFB   $0,$0
288C:
****************************************************************************************************
288C:              * SET.PREFIX (PREFIXPATH.IN)
288C:
****************************************************************************************************
288C:        00C6  SETPREFIX  EQU   $C6
288C:01            PREFX.PARMS DFB  $1
288D:8F 28                    DW    PREFX.PATH
288F:03            PREFX.PATH DFB   $3
2890:2E 44 31                 ASC   '.D1'
2893:
****************************************************************************************************
2893:              * GETTIME (TIME.OUT)
2893:
****************************************************************************************************
2893:        0063  GETTIME    EQU   $63
2893:              *
2893:01            DTPARMS    DFB   1
2894:96 28                    DW    DATETIME
2896:59 59 59 59   DATETIME   ASC   "YYYYMMDDWHHMMSSMMM"
```

```
28A8:              543
****************************************************************************************************
28A8:              544 *
28A8:              545 * END OF SOSLDR CODE
28A8:              546 *
28A8:              547
****************************************************************************************************
28A8:        0050  548 SLOP       EQU   >$F8-*
28A8:        0050  549            DS    SLOP             ; +-----------------------------------+
28F8:        0200  550 INITMODULE DS    $200             ; ! KERNEL'S INIT MODULE RESIDES HERE !
2AF8:        2AF8  551 LDREND     EQU   *                ; +-----------------------------------+
2AF8:        0EF8  552 FILE       EQU   *-$2000+$400
2AF8:              553
****************************************************************************************************
2AF8:              554 * SOS INTERPRETER FILE
2AF8:              555
****************************************************************************************************
2AF8:        0EF8  556 I.FILE     EQU   FILE
2AF8:        0F00  557 I.HDR.CNT  EQU   I.FILE+$8
2AF8:              558
****************************************************************************************************
2AF8:              559 * SOS DRIVER FILE
2AF8:              560
****************************************************************************************************
2AF8:        0EF8  561 D.FILE     EQU   FILE
2AF8:        0F00  562 D.HDR.CNT  EQU   D.FILE+$8
2AF8:        0F02  563 D.DRIVES   EQU   D.HDR.CNT+$2
2AF8:        0F14  564 D.CHRSET   EQU   D.DRIVES+$2+$10
2AF8:        1324  565 D.KYBD     EQU   D.CHRSET+$10+$400
2AF8:              566
****************************************************************************************************

2AF8:              567            LST   ON
2AF8:        2AF8  568 ZZEND      EQU   *
2AF8:        0CF8  569 ZZLEN      EQU   ZZEND-ZZORG
2AF8:              570 *

***** UNDEFINED IDENTIFIER ERROR IN LINE  571
2AF8:        0CF8  571            IFNE  ZZLEN-LENLODR    ;!BITROT

>>>>>>FAILURE:"SOSORG FILE IS INCORRECT FOR SOS LOADER"
2AF8:              573            FIN
2AF8:              574 *
```

```
 1FD3 ADEV.EXIT      1F8A ADEV010        1FD0 ADEV020        22BA ADVANCE
 25E1 ALDS.EXIT      25CC ALDS010        25CE ALDS020        1F79 ALLOC.DEV
 25C0 ALLOC.DSEG     256A ALLOC.SEG      04B7 AMSGADR        0009 AMSGL
 2797 AMSG           FFEF B.REG         X0016 BFM.INIT      X0017 BFM.INIT2
 2505 BLDS010        2508 BLDS020       X0011 BLKD.SIZE     X0012 BLKDLST
X0015 BMGR.INIT      24FD BUILD.DSEG    X0013 CFMGR.INIT     2215 CLEAR0
 2217 CLEAR1        X0018 CLK.INIT       2879 CLOSE.PARMS    00CC CLOSE
 287A CLOSE.REF      07D0 CMSGADR        27EE CMSG           0028 CMSGL
   26 CNT              1C CODE.P         1B00 CSPAGE         1600 CXPAGE
 1A00 CZPAGE         0F14 D.CHRSET       0F02 D.DRIVES       0EF8 D.FILE
 0F00 D.HDR.CNT      1324 D.KYBD         2869 D.LABEL        1E3C D.PATH
 23C2 DADD           238B DADV010        237E DADVANCE       2896 DATETIME
 27B5 DAYNAME          20 DIB.DCB          17 DIB.DTYPE        02 DIB.ENTRY
   14 DIB.FLAGS        14 DIB.P            16 DIB.UNIT      X0019 DIB1
X001A DIB2          X001B DIB3          X001C DIB4          X0008 DMGR.INIT
 27A0 DMSG           06B1 DMSGADR        0015 DMSGL          2515 DSEGLIST
 2514 DSEGX            24 DST.P            2A DSTBANK        2893 DTPARMS
 FFDF E.REG          07A8 EMSGADR        2605 ERR0           0009 ERR0L
 0008 ERR0X          0011 ERR10L         00EB ERR10X         26E0 ERR10
 001A ERR1L          0022 ERR1X          260E ERR1           2628 ERR2
 003A ERR2X          2640 ERR3           0067 ERR4X          0013 ERR5L
 266D ERR5           2605 ERR            0018 ERR2L          0018 ERR3L
 0052 ERR3X          0015 ERR4L          2658 ERR4           007A ERR5X
 2680 ERR6           0015 ERR6L          008F ERR6X          2695 ERR7
 0025 ERR7L          00B4 ERR7X          00C4 ERR8X          26BA ERR8
 0010 ERR8L          26CA ERR9           0016 ERR9L          00DA ERR9X
 25E2 ERROR            2E ETEMP         X0007 EVQ.INIT      ?  00 FALSE
 0EF8 FILE           0041 FINDSEG          10 FIRST.ADIB     23DC FLAG010
 23E4 FLAG015        2403 FLAG020        241C FLAG025        2431 FLAG050
 2434 FLAG100        23DB FLAGS          24C3 GETM.EXIT      24AD GETM010
 248C GETMEM         0063 GETTIME          02 I.BASE.P       0EF8 I.FILE
 0F00 I.HDR.CNT      2861 I.LABEL        1E0C I.PATH         227B INIT.KRNL
 22B3 INITK.ERR     ?28F8 INITMODULE    X0006 INT.INIT         00 K.BASE
 1E0A K.DRIVES      ?1E00 K.FILE        ?1E0B K.FLAGS       ?1E08 K.HDR.CNT
 1E6C LDR.ADR       ?1E6E LDR.CNT        1FD6 LDR010         2056 LDR020
 2061 LDR030         2084 LDR040         20A7 LDR050         20AD LDR051
 20B9 LDR052         20C0 LDR053         20F6 LDR070         20FF LDR080
 2115 LDR090         212B LDR100         2131 LDR101         213D LDR102
 2144 LDR103         2198 LDR105         21C1 LDR110         21D6 LDR120
 21DD LDR130         21E8 LDR140         2AF8 LDREND           2C LINK.P
 1F53 LINK100        1EF9 LINK           1F60 LINK.INIT      1F2B LINK010
 1F41 LINK030       X0009 MAX.DNUM      X0002 MEMSIZE       X0014 MMGR.INIT
 27CA MONNAME        1EB3 MOVE           1EF4 MOVE.EXIT      1EDC MOVE.PAGE
 1EC7 MOVE010        1EE3 MOVE020        24FC NEWD.EXIT      24DC NEWD010
 24C4 NEWDST         2436 NEXT.DIB       21A0 NEXTDRIVER     244F NXTD010
 245D NXTD020        2468 NXTD030        247E NXTD040        248A NXTD998
 248B NXTD999        281D OPEN.LIST      2816 OPEN.PARMS     2819 OPEN.REF
 00C8 OPEN           2821 PATH             16 PG.ALIGN       288C PREFX.PARMS
 288F PREFX.PATH       12 PREV.ADIB.P      18 PREVBANK         19 PREVDST
 25EE PRNT010       *B3B0 R             04 RDBUF.P         ?2873 READ.BUF
?2877 READ.BYTRD    ?2875 READ.BYT       00CA READ           2871 READ.PARMS
 2872 READ.REF         20 REL.END        2569 REL.EXIT       2529 REL.LOOP
   1E REL.P          2519 RELOC          0040 REQSEG         237D REV.EXIT
   0C REV.SAVE       2333 REV010         2339 REV020         22FE REVERSE
 F1B9 ROM.ADR          A0 ROM.ID         2586 RSEG           25BB RSEG010
X0003 SCRNMODE      X000E SDT.ADRH      X000D SDT.ADRL      X000F SDT.BANK
```

```
X000C SDT.DIBH      X000B SDT.DIBL      X000A SDT.SIZE      X0010 SDT.UNIT
 2886 SEGBASE        288A SEGID          2888 SEGLIM         2885 SEGMENT
 287B SEGMENT1       287E SEGPGCNT      ?287C SEGSRCH        2232 SET.DRIVES
 00C6 SETPREFIX      1E73 SLDR010          50 SLOP           05A8 SMSGADR
 1E70 SOSLDR         1FD4 SOSLDR1       X0005 SOSVERL       X0004 SOSVER
   22 SRC.P          0100 SSPAGE         225F STDR010        2266 STDR020
 226D STDR030        2274 STDR040        1400 SXPAGE        X0001 SYSBANK
   06 SYSBUF.P       1800 SZPAGE           09 TEMP.ADRH        08 TEMP.BANK
?  80 TRUE           2376 V040           2225 WAIT           1F28 WALKLINKS
 26F3 WAM010         278D WCM010         2724 WDM010         2748 WDM020
 2750 WDM030         278B WDM040         2780 WDM050         26F1 WELCOME
   0A WORK.P         2705 WSM010           2F WTEMP          0000 Y
 FFD0 Z.REG            00 ZPAGE          2AF8 ZZEND          0CF8 ZZLEN
 1E00 ZZORG
```

```
ERROR SUMMARY
UNDEFINED IDENTIFIER ERROR IN LINE  571 OF FILE # 07
UNDEFINED IDENTIFIER ERROR IN LINE  571 OF FILE # 07

    3  ERRORS IN THIS ASSEMBLY
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED  2175
** FREE SPACE PAGE COUNT   73
```

```
SOURCE   FILE #01 =>INIT.SRC
 INCLUDE FILE #02 =>SOSORG
***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  186

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  229

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  429

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  431

***** EXTRN USED AS ZXTRN IN LINE  433

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  434

***** EXTRN USED AS ZXTRN IN LINE  441

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  442
```

```
0000:                             REL
0000:                             INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955      BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52      BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E      BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
28F8:        28F8                 ORG   ORGINIT
28F8:        28F8      ZZORG      EQU   *
28F8:                             MSB   OFF
28F8:
****************************************************************************************************
28F8:                  *          COPYRIGHT (C) APPLE COMPUTER INC.  1981
28F8:                  *                    ALL RIGHTS RESERVED
28F8:
****************************************************************************************************
28F8:                  *
28F8:                  * SOS INIT MODULE (VERSION = 1.1O   )
28F8:                  *                (DATE    = 8/04/81)
28F8:                  *
28F8:
****************************************************************************************************
28F8:                  *
28F8:        28F8                 ENTRY INT.INIT
28F8:        298B                 ENTRY EVQ.INIT
28F8:        29A8                 ENTRY CLK.INIT
28F8:        2A36                 ENTRY MMGR.INIT
28F8:        2A22                 ENTRY BMGR.INIT
28F8:        2A07                 ENTRY DMGR.INIT
28F8:        29FB                 ENTRY CFMGR.INIT
28F8:        2A6A                 ENTRY BFM.INIT
28F8:                  *
28F8:                  *  EXTERNAL SUBROUTINES & DATA
28F8:                  *
28F8:        0000                 EXTRN SXPAGE
28F8:        0000                 EXTRN SYSDEATH
28F8:                  *
28F8:                  *  INTERRUPT SYSTEM INITIALIZATION
28F8:                  *
28F8:        0000                 EXTRN COLDSTRT
28F8:        0000                 EXTRN IRQ.RCVR
28F8:        0000                 EXTRN NMI.RCVR
28F8:        0000                 EXTRN NMIFLAG
28F8:        0000                 EXTRN SIRTABLE
28F8:        0000                 EXTRN SIRTBLSIZ
28F8:        0000                 EXTRN ZPGSTACK
28F8:        0000                 EXTRN ZPGSTART
28F8:                  *
28F8:                  *  EVENT QUEUE INITIALIZATION
28F8:                  *
28F8:        0000                 EXTRN EV.QUEUE
28F8:        0000                 EXTRN EVQ.LEN
28F8:        0000                 EXTRN EVQ.CNT
28F8:        0000                 EXTRN EVQ.SIZ
28F8:        0000                 EXTRN EVQ.FREE
28F8:        0000                 EXTRN EVQ.LINK
28F8:                  *
28F8:                  *  CLOCK INITIALIZATION
28F8:                  *
28F8:        0000                 EXTRN PCLOCK
28F8:                  *
28F8:                  *  CHARACTER FILE MANAGER INITIALIZATION
```

```
28F8:                  *
28F8:        0000                 EXTRN CFCB.MAX
28F8:        0000                 EXTRN CFCB.DEV
28F8:                  *
28F8:                  *  DEVICE MANAGER INITIALIZATION
28F8:                  *
28F8:        0000                 EXTRN DMGR
28F8:        0000                 EXTRN MAX.DNUM
28F8:                  *
28F8:                  *  BUFFER MANAGER INITIALIZATION
28F8:                  *
28F8:        0000                 EXTRN BUF.CNT
28F8:        0000                 EXTRN PGCT.T
28F8:        0000                 EXTRN XBYTE.T
28F8:        0000                 EXTRN BUFREF
28F8:                  *
28F8:                  *  MEMORY MANAGER INITIALIZATION
28F8:                  *
28F8:        0000                 EXTRN ST.CNT
28F8:        0000                 EXTRN ST.ENTRY
28F8:        0000                 EXTRN ST.FREE
28F8:        0000                 EXTRN ST.FLINK
28F8:        0000                 EXTRN VRT.LIM
28F8:        0000                 EXTRN MEMSIZE
28F8:        0000                 EXTRN MEM2SML
28F8:                  *
28F8:                  *  BLOCK FILE MANAGER INITIALIZATION
28F8:                  *
28F8:        0000                 EXTRN FCBZPP
28F8:        0000                 EXTRN PATHBUF
28F8:        0000                 EXTRN VCB
28F8:        0000                 EXTRN WORKSPC
28F8:        0000                 EXTRN PFIXPTR
28F8:        0000                 EXTRN FCBADDRH
28F8:        0000                 EXTRN BMAPAGE
28F8:        0000                 EXTRN BMBPAGE
28F8:        0000                 EXTRN BMAMADR
28F8:        0000                 EXTRN BMBMADR
28F8:        0000                 EXTRN BFMFCB1
28F8:        0000                 EXTRN BFMFCB2
28F8:                  *
28F8:                  *  CONSTANT DECLARATIONS
28F8:                  *
28F8:        0080      TRUE       EQU   $80
28F8:        0000      FALSE      EQU   $00
28F8:        0040      BITON6     EQU   $40
28F8:        0080      BITON7     EQU   $80
28F8:                  *
28F8:                  *  SYSTEM CONTROL REGISTERS
28F8:                  *
28F8:        FFDF      E.REG      EQU   $FFDF            ;ENVIRONMENT REGISTER
28F8:        FFD0      Z.REG      EQU   $FFD0            ;ZERO PAGE REGISTER
```

```
28F8:              *
28F8:              *  6522 REGISTERS
28F8:              *
28F8:        FFD2  D.DDRB     EQU   $FFD2
28F8:        FFD3  D.DDRA     EQU   $FFD3
28F8:        FFDB  D.ACR      EQU   $FFDB
28F8:        FFDC  D.PCR      EQU   $FFDC
28F8:        FFDD  D.IFR      EQU   $FFDD
28F8:        FFDE  D.IER      EQU   $FFDE
28F8:        FFE0  E.IORB     EQU   $FFE0
28F8:        FFE2  E.DDRB     EQU   $FFE2
28F8:        FFE3  E.DDRA     EQU   $FFE3
28F8:        FFEB  E.ACR      EQU   $FFEB
28F8:        FFEC  E.PCR      EQU   $FFEC
28F8:        FFED  E.IFR      EQU   $FFED
28F8:        FFEE  E.IER      EQU   $FFEE
28F8:        C0F1  ACIASTAT   EQU   $C0F1
28F8:              *
28F8:              *
28F8:              ************************************************************
28F8:              *
28F8:              *  THIS SUBROUTINE INITIALIZES THE INTERRUPT SYSTEM.
28F8:              *  ALL HARDWARE INTERRUPTS ARE MASKED AND THE
28F8:              *  INTERRUPT ALLOCATION TABLE IS CLEARED.
28F8:              *
28F8:              ************************************************************
28F8:              *
28F8:              *
28F8:        28F8  INT.INIT   EQU   *
28F8:78                       SEI                    ;DISABLE INTERRUPTS
28F9:A9 00                    LDA   #>ZPGSTART       ;SET UP MIH
28FB:8D 00 00                 STA   ZPGSTACK         ;  ZERO PAGE STACK POINTER
28FE:              *
28FE:AD DF FF                 LDA   E.REG            ;SELECT $C000 I/O SPACE
2901:48                       PHA                    ;  AND SET 1 MHZ
2902:09 C0                    ORA   #BITON7+BITON6
2904:8D DF FF                 STA   E.REG
2907:              *
2907:8D F1 C0                 STA   ACIASTAT         ;RESET ACIA
290A:              *
290A:A9 FF                    LDA   #$FF             ;SET UP 6522 D
290C:8D D2 FF                 STA   D.DDRB
290F:8D D3 FF                 STA   D.DDRA
2912:A9 00                    LDA   #$00
2914:8D DB FF                 STA   D.ACR
2917:A9 76                    LDA   #$76
2919:8D DC FF                 STA   D.PCR
291C:A9 7F                    LDA   #$7F
291E:8D DD FF                 STA   D.IFR
2921:8D DE FF                 STA   D.IER
2924:A9 82                    LDA   #$82
2926:8D DE FF                 STA   D.IER
2929:              *
2929:A9 3F                    LDA   #$3F             ;SET UP 6522 E
292B:8D E2 FF                 STA   E.DDRB
292E:A9 0F                    LDA   #$0F
```

```
2930:8D E3 FF      165            STA   E.DDRA
2933:A9 00         166            LDA   #$00
2935:8D EB FF      167            STA   E.ACR
2938:A9 63         168            LDA   #$63
293A:8D EC FF      169            STA   E.PCR
293D:A9 7F         170            LDA   #$7F
293F:8D ED FF      171            STA   E.IFR
2942:8D EE FF      172            STA   E.IER
2945:              173 *
2945:A9 FF         174            LDA   #$FF
2947:8D E0 FF      175            STA   E.IORB           ;SOUND PORT
294A:2C D8 C0      176            BIT   $C0D8            ;DISABLE GRAPHICS SCROLL
294D:2C DA C0      177            BIT   $C0DA            ;DISABLE CHARACTER DOWNLOAD
2950:2C DC C0      178            BIT   $C0DC            ;DISABLE ENSEL
2953:2C DE C0      179            BIT   $C0DE            ;SET ENSIO FOR INPUT
2956:              180 *
2956:68            181            PLA                    ;RESTORE E REGISTER
2957:8D DF FF      182            STA   E.REG
295A:              183 *
295A:A9 00         184            LDA   #FALSE
295C:8D 00 00      185            STA   NMIFLAG          ;CLEAR NMI WAIT FLAG

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  186
295F:AC 00 00      186            LDY   #>SIRTBLSIZ-1
2962:99 00 00      187 INTI010    STA   SIRTABLE,Y       ;  ALLOCATION TABLE
2965:88            188            DEY
2966:10 FA   2962  189            BPL   INTI010
2968:A9 80         190            LDA   #TRUE
296A:8D 0A 00      191            STA   SIRTABLE+$0A     ;LOCK DOWN ANY SLOT SIR
296D:              192 *
296D:A2 05         193            LDX   #$05
296F:BD 7F 29      194 INTI020    LDA   RAMVECT,X        ;SET UP VECTORS
2972:9D FA FF      195            STA   $FFFA,X          ;  AT $FFFA - $FFFF
2975:BD 85 29      196            LDA   RAMJMPS,X        ;SET UP JMP INSTRUCTIONS
2978:9D CA FF      197            STA   $FFCA,X          ;  AT $FFCA - $FFCF
297B:CA            198            DEX
297C:10 F1   296F  199            BPL   INTI020
297E:60            200            RTS
297F:              201 *
297F:00 00         202 RAMVECT    DW    NMI.RCVR
2981:00 00         203            DW    COLDSTRT
2983:00 00         204            DW    IRQ.RCVR
2985:4C 00 00      205 RAMJMPS    JMP   NMI.RCVR
2988:4C 00 00      206            JMP   IRQ.RCVR
```

```
298B:              208 ************************************************************
298B:              209 *
298B:              210 *  THIS SUBROUTINE INITIALIZES THE EVENT QUEUE.  ALL ENTRIES
298B:              211 *  ARE CLEARED AND LINKED INTO THE FREE LIST.  THE ACTIVE
298B:              212 *  LIST IS EMPTY.
298B:              213 *
298B:              214 ************************************************************
298B:              215 *
298B:              216 *
298B:        298B  217 EVQ.INIT   EQU   *
298B:              218 *
298B:              219 *  CLEAR ALL ENTRIES
298B:              220 *
298B:A0 00         221            LDY   #>EVQ.LEN
298D:A9 00         222            LDA   #0
298F:99 FF FF      223 EVQI010    STA   EV.QUEUE-1,Y
2992:88            224            DEY
2993:D0 FA   298F  225            BNE   EVQI010
2995:              226 *
2995:              227 *  SET UP FREE LIST
2995:              228 *

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  229
2995:AE 00 00      229            LDX   #>EVQ.CNT-2
2998:A9 00         230            LDA   #>EVQ.SIZ
299A:8D 00 00      231            STA   EVQ.FREE
299D:A8            232 EVQI020    TAY
299E:18            233            CLC
299F:69 00         234            ADC   #>EVQ.SIZ
29A1:99 00 00      235            STA   EVQ.LINK,Y
29A4:CA            236            DEX
29A5:D0 F6   299D  237            BNE   EVQI020
29A7:60            238            RTS
```

```
29A8:              ************************************************************
29A8:              *
29A8:              *  THIS SUBROUTINE INITIALIZES THE PSEUDO CLOCK.  IF THE
29A8:              *  RAM BEHIND THE "D" 6522 HAS THE PROPER CHECKSUM, IT
29A8:              *  IS USED TO INITIALIZE THE PSEUDO CLOCK.  OTHERWISE,
29A8:              *  THE PSEUDO CLOCK IS SET TO ZERO.
29A8:              *
29A8:              * (ADDED 23 OCT 81)
29A8:              * BOTH THE CLOCK AND PSEUDO CLOCK ARE
29A8:              * ARE NOW INITIALIZED
29A8:              *
29A8:              ************************************************************
29A8:              *
29A8:        00F0  PCLK       EQU   $F0
29A8:        00F2  CKSUM      EQU   $F2
29A8:        0011  CLKICR     EQU   $11              ; CLOCK INTERRUPT CONTROL REG
29A8:        0016  CLKSTBY    EQU   $16              ; CLOCK STANDBY INTERRUPT
29A8:        C070  CLOCK      EQU   $C070
29A8:              *
29A8:        29A8  CLK.INIT   EQU   *
29A8:A9 D0                    LDA   #$D0
29AA:85 F0                    STA   PCLK             ;POINT (PCLK) TO 8F:FFD0
29AC:A9 FF                    LDA   #$FF
29AE:85 F1                    STA   PCLK+1
29B0:A9 8F                    LDA   #$8F
29B2:8D F1 00                 STA   SXPAGE+PCLK+1
29B5:A9 A5                    LDA   #$A5
29B7:85 F2                    STA   CKSUM            ;INITIALIZE CHECKSUM
29B9:              *
29B9:A0 00                    LDY   #$00
29BB:B1 F0         CLK010     LDA   (PCLK),Y         ;COPY SAVED CLOCK DATA
29BD:99 00 00                 STA   PCLOCK,Y         ;  TO PSEUDO CLOCK
29C0:45 F2                    EOR   CKSUM
29C2:85 F2                    STA   CKSUM            ;UPDATE CHECKSUM
29C4:C8                       INY
29C5:C0 0A                    CPY   #$0A
29C7:90 F2   29BB             BCC   CLK010
29C9:              *
29C9:D1 F0                    CMP   (PCLK),Y         ;TEST CHECKSUM
29CB:F0 08   29D5             BEQ   CLK030
29CD:              *
29CD:A9 00                    LDA   #$00
29CF:88            CLK020     DEY
29D0:99 00 00                 STA   PCLOCK,Y         ;ZERO PSEUDO CLOCK
29D3:D0 FA   29CF             BNE   CLK020
29D5:AD DF FF      CLK030     LDA   E.REG
29D8:48                       PHA
29D9:09 80                    ORA   #$80             ; SET 1 MHZ
29DB:8D DF FF                 STA   E.REG
29DE:A9 00                    LDA   #$00
29E0:AC D0 FF                 LDY   Z.REG
29E3:A2 11                    LDX   #CLKICR
29E5:8E D0 FF                 STX   Z.REG
29E8:8D 70 C0                 STA   CLOCK            ; DISABLE CLOCK INTERRUPTS
29EB:A2 16                    LDX   #CLKSTBY
29ED:8E D0 FF                 STX   Z.REG
```

```
29F0:8D 70 C0      296            STA   CLOCK            ; DISABLE STANDBY INTERRUPT
29F3:8C D0 FF      297            STY   Z.REG
29F6:68            298            PLA
29F7:8D DF FF      299            STA   E.REG
29FA:60            300            RTS

***** DIRECTIVE OPERAND ERROR IN LINE  301
29FB:              301            SBTL  "CHARACTER       FILE MANAGER INITIALIZATION"
29FB:              302 ************************************************************
29FB:              303 *
29FB:              304 * CHAR FILE MANAGER INITIALIZATION ROUTINE
29FB:              305 *
29FB:              306 * CFMGR.INIT INITIALIZES ALL ENTRIES IN THE CFCB TABLE TO
29FB:              307 * THE "FREE" STATE.
29FB:              308 *
29FB:              309 ************************************************************
29FB:              310 *
29FB:        29FB  311 CFMGR.INIT EQU   *
29FB:A9 80         312            LDA   #$80
29FD:A2 FF         313            LDX   #CFCB.MAX-1
29FF:9D 00 00      314 CFINIT010  STA   CFCB.DEV,X
2A02:CA            315            DEX
2A03:10 FA   29FF  316            BPL   CFINIT010
2A05:60            317            RTS
```

```
2A06:              ************************************************************
2A06:              *
2A06:              * DEVICE MANAGER INITIALIZATION ROUTINE
2A06:              *
2A06:              * INITIALIZES THE SYSTEM DEVICE TABLE (SDT) BY WALKING THE
2A06:              * DEVICE INFORMATION BLOCK (DIB) LINKS.  CALLED BY SYSLDR.
2A06:              *
2A06:              ************************************************************
2A06:              *
2A06:        00C0  D.TPARMX   EQU   $C0
2A06:        00C0  REQCODE    EQU   D.TPARMX+$00
2A06:        00C1  DNUM       EQU   D.TPARMX+$01
2A06:        0001  DNUM.TEMP  DS    1
2A07:              *
2A07:              *
2A07:        2A07  DMGR.INIT  EQU   *
2A07:AE 00 00                 LDX   MAX.DNUM
2A0A:EE 00 00                 INC   MAX.DNUM         ; MAX.DNUM:=MAX DEV NUMBER IN SYSTEM+1
2A0D:8E 06 2A                 STX   DNUM.TEMP
2A10:A9 08         DMI110     LDA   #8               ; INITIALIZE ALL DEVICES IN SYSTEM (D.INIT)
2A12:85 C0                    STA   REQCODE
2A14:AD 06 2A                 LDA   DNUM.TEMP
2A17:85 C1                    STA   DNUM
2A19:20 00 00                 JSR   DMGR
2A1C:CE 06 2A                 DEC   DNUM.TEMP
2A1F:D0 EF   2A10             BNE   DMI110
2A21:60                       RTS                    ; NORMAL EXIT
```

```
2A22:              ************************************************************
2A22:              *
2A22:              * BMGR.INIT
2A22:              *
2A22:              * THIS ROUTINE INITIALIZES THE BUFFER TABLE'S ENTRIES TO "FREE".
2A22:              * CALLED DURING SYSTEM BOOT.
2A22:              *
2A22:              ************************************************************
2A22:              *
2A22:        2A22  BMGR.INIT  EQU   *
2A22:A9 FF                    LDA   #$FF             ; USED WHEN FINDING LOWEST BUFFER IN TBL (BUFCOMPACT)
2A24:8D 00 00                 STA   XBYTE.T
2A27:              *
2A27:A2 FF                    LDX   #BUF.CNT-1
2A29:A9 80                    LDA   #$80
2A2B:9D 00 00      BUFI010    STA   PGCT.T,X         ;SET ALL ENTRIES "FREE"
2A2E:CA                       DEX
2A2F:D0 FA   2A2B             BNE   BUFI010
2A31:              *
2A31:8E 00 00                 STX   BUFREF           ;ZERO COUNT BYTE IN BUFFER REFERENCE TABLE
2A34:              *
2A34:18                       CLC
2A35:60                       RTS
```

```
2A36:              ************************************************************
2A36:              *
2A36:              * MMGR.INIT
2A36:              *
2A36:              * THIS ROUTINE INITIALIZES THE MEMORY MANAGER'S SEGMENT TABLE
2A36:              * TO FREE ENTRIES, AND DETERMINES THE MEMORY SIZE OF THE
2A36:              * MACHINE (96K,128K,160K,192K,224K,256K,..,512K IN 32K STEPS).
2A36:              *
2A36:              ************************************************************
2A36:              *
2A36:        2A36  MMGR.INIT  EQU   *
2A36:              *
2A36:              * INIT SEGMENT TABLE
2A36:              *
2A36:A9 00                    LDA   #0
2A38:8D 00 00                 STA   ST.ENTRY
2A3B:A9 81                    LDA   #$81
2A3D:8D 00 00                 STA   ST.FREE
2A40:              *
2A40:A0 FF                    LDY   #ST.CNT-1
2A42:A9 80                    LDA   #$80             ; SET LAST LINK TO NULL
2A44:99 00 00                 STA   ST.FLINK,Y
2A47:98            MEMI010    TYA
2A48:09 80                    ORA   #$80
2A4A:88                       DEY
2A4B:99 00 00                 STA   ST.FLINK,Y
2A4E:D0 F7   2A47             BNE   MEMI010
2A50:              *
2A50:              * COMPUTE VIRTUAL LIMIT FROM MEMORY SIZE
2A50:              * VRT.LIM := NUMBER OF PAGES IN BANK SWITCHED MEMORY - 1
2A50:              *         := (MEMSIZ-2)*64 - 1
2A50:              *         := (MEMSIZ-4)*64 + 127
2A50:              *
2A50:38                       SEC
2A51:AD 00 00                 LDA   MEMSIZE
2A54:E9 04                    SBC   #4
2A56:90 0D   2A65             BCC   MEMI.ERR
2A58:4A                       LSR   A
2A59:4A                       LSR   A
2A5A:8D 01 00                 STA   VRT.LIM+1
2A5D:A9 FE                    LDA   #$FE
2A5F:6A                       ROR   A
2A60:8D 00 00                 STA   VRT.LIM
2A63:18                       CLC
2A64:60                       RTS                    ; NORMAL EXIT
2A65:              *
2A65:A9 00         MEMI.ERR   LDA   #MEM2SML         ; FATAL ERR - MEM < 64K
2A67:20 00 00                 JSR   SYSDEATH
```

```
2A6A:              420 ************************************************************
2A6A:              421 *
2A6A:              422 *  BLOCK FILE MANAGER INITIALIZATION
2A6A:              423 *
2A6A:              424 ************************************************************
2A6A:              425 *
2A6A:        1400  426 SISTER     EQU   $1400            ;BFM XPAGE
2A6A:        2A6A  427 BFM.INIT   EQU   *
2A6A:A9 00         428            LDA   #BFMFCB1         ; ADDRESS OF PAGE 1 OF FCB

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  429
2A6C:8D 00 00      429            STA   >FCBZPP+1
2A6F:A9 00         430            LDA   #BFMFCB2         ; AND PAGE 2

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  431
2A71:8D 00 00      431            STA   >FCBZPP+3
2A74:A9 00         432            LDA   #0

***** EXTRN USED AS ZXTRN IN LINE  433
2A76:85 00         433            STA   >FCBZPP          ; FCB PAGE ALIGNED

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  434
2A78:8D 00 00      434            STA   >FCBZPP+2
2A7B:8D 01 14      435            STA   SISTER+FCBZPP+1  ; PREPARE PART OF EXTEND BYTE
2A7E:8D 03 14      436            STA   SISTER+FCBZPP+3
2A81:A8            437            TAY                    ; MAKE ZERO INTO INDEX
2A82:        2A82  438 CLRBUFFS   EQU   *
2A82:99 00 00      439            STA   PATHBUF,Y        ; PATHNAME BUFFER PAGE
2A85:99 00 00      440            STA   VCB,Y            ; VOLUME CONTROL BLOCK PAGE

***** EXTRN USED AS ZXTRN IN LINE  441
2A88:91 00         441            STA   (>FCBZPP),Y      ; BOTH FILE CONTROL BLOCK PAGES

***** RELATIVE EXPRSN OPERATOR ERROR IN LINE  442
2A8A:8D 00 00      442            STA   (>FCBZPP+2),Y
2A8D:C8            443            INY
2A8E:D0 F2   2A82  444            BNE   CLRBUFFS
2A90:A2 3F         445            LDX   #$3F             ; SIZE OF MY ZERO PAGE STUFF
2A92:95 00         446 CLRZWRK    STA   0,X              ; ZERO PAGE ZEROED
2A94:9D 00 00      447            STA   WORKSPC,X
2A97:CA            448            DEX
2A98:10 F8   2A92  449            BPL   CLRZWRK
2A9A:A9 00         450            LDA   #<PATHBUF
2A9C:8D 01 00      451            STA   PFIXPTR+1
2A9F:A9 00         452            LDA   #BFMFCB1
2AA1:8D 00 00      453            STA   FCBADDRH
2AA4:A9 00         454            LDA   #BMAPAGE         ; BIT MAP A PAGE NUMBER
2AA6:8D 00 00      455            STA   BMAMADR
2AA9:A9 00         456            LDA   #BMBPAGE         ; BIT MAP B PAGE NUMBER
2AAB:8D 00 00      457            STA   BMBMADR
2AAE:18            458            CLC
2AAF:60            459            RTS
2AB0:              460 *

2AB0:              461            LST   ON
2AB0:        2AB0  462 ZZEND      EQU   *
```

```
2AB0:        01B8  463 ZZLEN      EQU   ZZEND-ZZORG
2AB0:        0006  464            IFNE  ZZLEN-LENINIT

>>>>>>FAILURE:"SOSORG FILE IS INCORRECT FOR INIT"
2AB0:              466            FIN
```

```
 C0F1 ACIASTAT      N2A6A BFM.INIT      X0033 BFMFCB1       X0034 BFMFCB2
   40 BITON6           80 BITON7        ?2E00 BLABFMI        3200 BLABFM
 6B52 BLABUFMG       6955 BLACFM         5E99 BLADISK3       64D9 BLADMGR
 68F4 BLAFMGR       ?2CF8 BLAGLOB       ?2AF8 BLAINIT        55C0 BLAIPL
 2000 BLALODR       ?6E6E BLAMEMMG       5466 BLAOMSG        5466 BLAPATCH
 665E BLASCMGR       6404 BLASERR        5A8B BLAUMGR       X0031 BMAMADR
X002F BMAPAGE       X0032 BMBMADR       X0030 BMBPAGE       N2A22 BMGR.INIT
X001E BUF.CNT        2A2B BUFI010       X0021 BUFREF        X001B CFCB.DEV
X001A CFCB.MAX       29FF CFINIT010     N29FB CFMGR.INIT       F2 CKSUM
N29A8 CLK.INIT       29BB CLK010         29CF CLK020         29D5 CLK030
   11 CLKICR           16 CLKSTBY        C070 CLOCK          2A82 CLRBUFFS
 2A92 CLRZWRK       X000B COLDSTRT       FFDB D.ACR          FFD3 D.DDRA
 FFD2 D.DDRB         FFDE D.IER          FFDD D.IFR          FFDC D.PCR
   C0 D.TPARMX      N2A07 DMGR.INIT     X001C DMGR           2A10 DMI110
 2A06 DNUM.TEMP        C1 DNUM           FFEB E.ACR          FFE3 E.DDRA
 FFE2 E.DDRB         FFEE E.IER          FFED E.IFR          FFE0 E.IORB
 FFEC E.PCR          FFDF E.REG         X0013 EV.QUEUE      X0015 EVQ.CNT
X0017 EVQ.FREE      N298B EVQ.INIT      X0014 EVQ.LEN       X0018 EVQ.LINK
X0016 EVQ.SIZ        298F EVQI010        299D EVQI020          00 FALSE
X002E FCBADDRH      X0029 FCBZPP        N28F8 INT.INIT       2962 INTI010
 296F INTI020       X000C IRQ.RCVR      ?0400 LENBFMI        2266 LENBFM
 031C LENBUFMG       01FD LENCFM         056B LENDISK3       0185 LENDMGR
   61 LENFMGR        01B2 LENINIT        04CB LENIPL         0AF8 LENLODR
?0751 LENMEMMG       015A LENOMSG          00 LENPATCH       0296 LENSCMGR
   D5 LENSERR        040E LENUMGR       X001D MAX.DNUM      X0028 MEM2SML
 2A65 MEMI.ERR       2A47 MEMI010       X0027 MEMSIZE       N2A36 MMGR.INIT
X000D NMI.RCVR      X000E NMIFLAG        BC00 ORGBFM         B800 ORGBFMI
 F552 ORGBUFMG       F355 ORGCFM         E899 ORGDISK3       EED9 ORGDMGR
 FFBF ORGEND         F2F4 ORGFMGR       ?18FC ORGGLOB        28F8 ORGINIT
 DFC0 ORGIPL         1E00 ORGLODR        F86E ORGMEMMG       DE66 ORGOMSG
 DE66 ORGPATCH       F05E ORGSCMGR       EE04 ORGSERR        E48B ORGUMGR
X002A PATHBUF          F0 PCLK          X0019 PCLOCK        X002D PFIXPTR
X001F PGCT.T         2985 RAMJMPS        297F RAMVECT          C0 REQCODE
X000F SIRTABLE      X0010 SIRTBLSIZ      1400 SISTER        X0022 ST.CNT
X0023 ST.ENTRY      X0025 ST.FLINK      X0024 ST.FREE       X0009 SXPAGE
X000A SYSDEATH         80 TRUE          X002B VCB           X0026 VRT.LIM
X002C WORKSPC       X0020 XBYTE.T        FFD0 Z.REG         X0011 ZPGSTACK
X0012 ZPGSTART       2AB0 ZZEND          01B8 ZZLEN          28F8 ZZORG
```

```
ERROR SUMMARY
RELATIVE EXPRSN OPERATOR ERROR IN LINE  186 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  229 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  429 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  431 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  434 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  442 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  186 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  229 OF FILE # 02
DIRECTIVE OPERAND ERROR IN LINE  301 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  429 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  431 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  434 OF FILE # 02
RELATIVE EXPRSN OPERATOR ERROR IN LINE  442 OF FILE # 02

    4  WARNINGS IN THIS ASSEMBLY
   14  ERRORS IN THIS ASSEMBLY
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   525
** FREE SPACE PAGE COUNT   80
```

```
0000:                          REL
18FC:        18FC              ORG   $18FC
18FC:                          MSB   OFF
18FC:               ************************************************************
18FC:               *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
18FC:               *                    ALL RIGHTS RESERVED
18FC:               ************************************************************
18FC:               *
18FC:               *  SOS SYSTEM GLOBAL DATA & EQUATES
18FC:               *
18FC:               *  THIS MODULE CONTAINS THE SOS JUMP TABLE, AND ALL GLOBAL
18FC:               *  DATA AND EQUATES.  THE JUMP TABLE, AND ALL DATA THAT IS
18FC:               *  TO BE REFERENCED BY DEVICE HANDLERS, ARE ASSIGNED FIXED
18FC:               *  ADDRESSES AT THE BEGINNING OF MEMORY PAGE $19.  DATA
18FC:               *  THAT IS ONLY REFERENCED BY SOS BEGINS $1980, BUT MAY BE
18FC:               *  MOVED WHENEVER SOS IS RELINKED.
18FC:               *
18FC:               ************************************************************
18FC:               *
18FC:        0000              EXTRN ALLOCSIR
18FC:        0000              EXTRN DEALCSIR
18FC:        0000              EXTRN NMIDSBL
18FC:        0000              EXTRN NMIENBL
18FC:        0000              EXTRN QUEEVENT
18FC:        0000              EXTRN SELC800
18FC:        0000              EXTRN SYSDEATH
18FC:        0000              EXTRN SYSERR
18FC:        0000              EXTRN REQBUF
18FC:        0000              EXTRN GETBUFADR
18FC:        0000              EXTRN RELBUF
18FC:        0000              EXTRN NMIDBUG
18FC:        0000              EXTRN NMICONT
18FC:        0000              EXTRN COLDSTRT
18FC:               *
18FC:               *
18FC:        1900              ENTRY MEMSIZE
18FC:        1901              ENTRY SYSBANK
18FC:        1902              ENTRY SUSPFLSH
18FC:        1903              ENTRY NMIFLAG
18FC:        1906              ENTRY SCRNMODE
18FC:        1907              ENTRY GRSIZE
18FC:               *
18FC:        1980              ENTRY SERR
18FC:        1981              ENTRY DBUGBRK
18FC:        1985              ENTRY KYBDNMI
18FC:        198B              ENTRY NMISPSV
18FC:        19F6              ENTRY SDEATH.REGS
18FC:               *
18FC:        1990              ENTRY SOSVER
18FC:        0013              ENTRY SOSVERL
18FC:               *
18FC:        1800              ENTRY SZPAGE
18FC:        1400              ENTRY SXPAGE
18FC:        0100              ENTRY SSPAGE
18FC:               *
18FC:        1A00              ENTRY CZPAGE
```

```
18FC:        1600                 ENTRY CXPAGE
18FC:        1B00                 ENTRY CSPAGE
18FC:        19C8                 ENTRY CEVPRI
18FC:                   *
18FC:        19C9                 ENTRY SIRTEMP
18FC:        19CA                 ENTRY SIRARGSIZ
18FC:        19CB                 ENTRY IRQCNTR
18FC:        19CD                 ENTRY NMICNTR
18FC:        19CF                 ENTRY QEVTEMP
18FC:        19D0                 ENTRY QEV.THIS
18FC:        19D1                 ENTRY QEV.LAST
18FC:                   *
18FC:        0001                 ENTRY BADBRK
18FC:        0002                 ENTRY BADINT1
18FC:        0003                 ENTRY BADINT2
18FC:        0004                 ENTRY NMIHANG
18FC:        0005                 ENTRY EVQOVFL
18FC:        0006                 ENTRY STKOVFL
18FC:        0007                 ENTRY BADSYSCALL
18FC:        0008                 ENTRY DEV.OVFLOW
18FC:        0009                 ENTRY MEM2SML
18FC:        000A                 ENTRY VCBERR
18FC:        000B                 ENTRY FCBERR
18FC:        000C                 ENTRY ALCERR
18FC:        0051                 ENTRY DIRERR
18FC:        000E                 ENTRY TOOLONG
18FC:        000F                 ENTRY BADBUFNUM
18FC:        0010                 ENTRY BADBUFSIZ
18FC:        005A                 ENTRY BITMAPADR
18FC:                   *
18FC:        0001                 ENTRY BADSCNUM
18FC:        0002                 ENTRY BADCZPAGE
18FC:        0003                 ENTRY BADXBYTE
18FC:        0004                 ENTRY BADSCPCNT
18FC:        0005                 ENTRY BADSCBNDS
18FC:                   *
18FC:        0010                 ENTRY NODNAME
18FC:        0011                 ENTRY BADDNUM
18FC:                   *
18FC:        0040                 ENTRY BADPATH
18FC:        0041                 ENTRY CFCBFULL
18FC:        0042                 ENTRY FCBFULL
18FC:        0043                 ENTRY BADREFNUM
18FC:        0044                 ENTRY PATHNOTFND
18FC:        0045                 ENTRY VNFERR
18FC:        0046                 ENTRY FNFERR
18FC:        0047                 ENTRY DUPERR
18FC:        0048                 ENTRY OVRERR
18FC:        0049                 ENTRY DIRFULL
18FC:        004A                 ENTRY CPTERR
18FC:        004B                 ENTRY TYPERR
18FC:        004C                 ENTRY EOFERR
18FC:        004D                 ENTRY POSNERR
18FC:        004E                 ENTRY ACCSERR
18FC:        004F                 ENTRY BTSERR
18FC:        0050                 ENTRY FILBUSY
```

```
18FC:        0052                 ENTRY NOTSOS
18FC:        0053                 ENTRY BADLSTCNT
18FC:        0054                 ENTRY OUTOFMEM
18FC:        0055                 ENTRY BUFTBLFULL
18FC:        0056                 ENTRY BADSYSBUF
18FC:        0057                 ENTRY DUPVOL
18FC:        0058                 ENTRY NOTBLKDEV
18FC:        0059                 ENTRY LVLERR
18FC:                  *
18FC:        0070                 ENTRY BADJMODE
18FC:                  *
18FC:        00E0                 ENTRY BADBKPG
18FC:        00E1                 ENTRY SEGRQDN
18FC:        00E2                 ENTRY SEGTBLFULL
18FC:        00E3                 ENTRY BADSEGNUM
18FC:        00E4                 ENTRY SEGNOTFND
18FC:        00E5                 ENTRY BADSRCHMODE
18FC:        00E6                 ENTRY BADCHGMODE
18FC:        00E7                 ENTRY BADPGCNT
18FC:                  *
18FC:        0020                 ENTRY XREQCODE
18FC:        0021                 ENTRY XCTLCODE
18FC:        0022                 ENTRY XCTLPARM
18FC:        0023                 ENTRY XNOTOPEN
18FC:        0024                 ENTRY XNOTAVAIL
18FC:        0025                 ENTRY XNORESRC
18FC:        0026                 ENTRY XBADOP
18FC:        0027                 ENTRY XIOERROR
18FC:        0028                 ENTRY XNODRIVE
18FC:        002B                 ENTRY XNOWRITE
18FC:        002C                 ENTRY XBYTECNT
18FC:        002D                 ENTRY XBLKNUM
18FC:        002E                 ENTRY XDISKSW
18FC:        19D2                 ENTRY BACKMASK         ; MASK BYTE FOR BACKUP BIT.
18FC:                  *
18FC:        1908                 ENTRY E1908            ; DISK DRIVER IS READING/WRITING (SET) ELSE NOT (RESET)
18FC:                  *
```

```
18FC:00 19                    DW    SYSGLOB          ;SYSGLOB TARGET ADDRESS
18FE:00 01                    DW    $0100            ;  AND LENGTH
1900:              *
1900:              *  SYSTEM GLOBAL DATA
1900:              *    (ACCESSIBLE TO SOS AND DEVICE HANDLERS)
1900:              *
1900:        1900  SYSGLOB    EQU   *
1900:              *
1900:08            MEMSIZE    DFB   $08              ;MEMORY SIZE = 128K
1901:02            SYSBANK    DFB   $02              ;SYSTEM BANK = 2
1902:00            SUSPFLSH   DFB   $00              ;SYSOUT SUSPEND/FLUSH FLAG
1903:00            NMIFLAG    DFB   $00              ;NMI PENDING FLAG
1904:8F 19                    DW    NMIEXIT          ;DEFAULT NMI VECTOR
1906:80            SCRNMODE   DFB   $80              ;CURRENT SCREEN MODE
1907:00            GRSIZE     DFB   $00
1908:              *
1908:              *
1908:              *  SOS JUMP TABLE
1908:              *
1908:        0008             DS    SYSGLOB+$10-*,$00 ; USED BY THE MOUSE DRIVER
1910:4C 8F 19      USERNMI    JMP   NMIEXIT          ;KEYBOARD NMI VECTOR
1913:4C 00 00                 JMP   ALLOCSIR         ;ALLOCATE A SIR
1916:4C 00 00                 JMP   DEALCSIR         ;DEALLOCATE A SIR
1919:4C 00 00                 JMP   NMIDSBL          ;DISABLE NMI
191C:4C 00 00                 JMP   NMIENBL          ;ENABLE NMI
191F:4C 00 00                 JMP   QUEEVENT         ;QUEUE AN EVENT
1922:4C 00 00                 JMP   SELC800          ;SELECT I/O EXPANSION ROM
1925:4C 00 00                 JMP   SYSDEATH         ;SYSTEM DEATH
1928:4C 00 00                 JMP   SYSERR           ;SOS ERROR
192B:4C 00 00                 JMP   REQBUF           ;REQUEST BUFFER
192E:4C 00 00                 JMP   GETBUFADR        ;GET BUFFER'S ADDRESS
1931:4C 00 00                 JMP   RELBUF           ;RELEASE BUFFER
1934:4C D3 19                 JMP   CLRBMASK         ;VECTOR TO CLRBMASK
```

```
1937:              *
1937:              *  SOS DATA AND EQUATES
1937:              *    (ACCESSIBLE ONLY TO SOS)
1937:              *
1937:        0049            DS    SYSGLOB+$80-*,$00
1980:00            SERR       DFB   $00              ;SYSTEM ERROR CODE
1981:              *
1981:EA            DBUGBRK    NOP                    ;TO ENABLE DEBUG BREAK POINTS,
1982:68                       PLA                    ;  PATCH THESE BYTES TO
1983:68                       PLA                    ;  JMP TO THE DEBUGGER
1984:60                       RTS
1985:              *
1985:4C 10 19      KYBDNMI    JMP   USERNMI
1988:4C 00 00                 JMP   NMIDBUG
198B:00            NMISPSV    DFB   0
198C:4C 00 00                 JMP   NMICONT
198F:60            NMIEXIT    RTS
1990:              *
1990:              *
1990:53 4F 53 20   SOSVER     ASC   "SOS             1.3   01-DEC-82"
19A3:        0013  SOSVERL    EQU   *-SOSVER
19A3:              *
19A3:28 43 29 20              ASC   "(C)             1980, 1981 BY APPLE COMPUTER INC."
19C8:              *
19C8:        1908  E1908      EQU   $1908            ; ALLOCATED TO STEPHEN SMITH (MOUSE DRIVER)
19C8:              * ABOVE SET AND RESET IN DISK DRIVER
19C8:        1800  SZPAGE     EQU   $1800            ;SYSTEM ZERO PAGE
19C8:        1400  SXPAGE     EQU   $1400            ;SYSTEM EXTEND PAGE
19C8:        0100  SSPAGE     EQU   $0100            ;SYSTEM STACK PAGE
19C8:              *
19C8:        1A00  CZPAGE     EQU   $1A00            ;CALLER'S ZERO PAGE
19C8:        1600  CXPAGE     EQU   $1600            ;CALLER'S EXTEND PAGE
19C8:        1B00  CSPAGE     EQU   $1B00            ;CALLER'S STACK PAGE
19C8:00            CEVPRI     DFB   $00              ;CALLER'S EVENT PRIORITY
19C9:              *
19C9:00            SIRTEMP    DFB   $00              ;TEMP FOR ALLOCSIR & DEALCSIR
19CA:00            SIRARGSIZ  DFB   $00              ;ARGUMENT COUNT FOR ALLOCSIR & DEALCSIR
19CB:00 00         IRQCNTR    DW    $0000            ;FALSE IRQ COUNTER
19CD:00 00         NMICNTR    DW    $0000            ;COUNTER FOR NMILOCK
19CF:00            QEVTEMP    DFB   $00              ;TEMP FOR QUEEVENT
19D0:00            QEV.THIS   DFB   $00              ;POINTER FOR QUEEVENT
19D1:00            QEV.LAST   DFB   $00              ;POINTER FOR QUEEVENT
19D2:              *
19D2:        0000  SOSQUIT    DS    COLDSTRT
19D2:20            BACKMASK   DFB   BACKBIT          ; MASK USED BY BFM TO UPDATE BACKUP BIT
19D3:              *
19D3:              * TO CLEAR THE BACKUP BIT, A PROGRAM MUST JSR TO CLRBMASK THRU 1934 THEN DO A
19D3:              * SET-FILE-INFO WITH NO INTERVENING SOS CALLS.  ANY SOS CALL WILL
19D3:              * SET BACKMASK AGAIN.  THIS FEATURE IS INTENTIONALLY LEFT UNDOCUMENTED.
19D3:              *
19D3:29 20         CLRBMASK   AND   #BACKBIT         ; PURIFY
19D5:8D D2 19                 STA   BACKMASK         ; SET THE MASK
19D8:60                       RTS                     ; AND BACK TO THE CALLER
```

```
19D9:              *
19D9:              *  SYSTEM DEATH REGISTER SAVE AREA
19D9:              * (SYSTEM STACK MOVED TO $1700-$17FF)
19D9:              *
19D9:        001D             DS    SYSGLOB+$F6-*,$00
19F6:        19F6  SDEATH.REGS EQU  *
19F6:00                       DFB   $00              ;BANK
19F7:00                       DFB   $00              ;ZERO PAGE
19F8:00                       DFB   $00              ;ENVIRONMENT
19F9:00                       DFB   $00              ;Y
19FA:00                       DFB   $00              ;X
19FB:00                       DFB   $00              ;A
19FC:00                       DFB   $00              ;STATUS
19FD:00 00                    DW    $00              ;PROGRAM COUNTER
19FF:00                       DFB   $00              ;STACK POINTER
1A00:              *
1A00:              *  SYSTEM DEATH ERROR NUMBERS
1A00:              *
1A00:        0001  BADBRK     EQU   $01              ;BRK FROM SOS
1A00:        0002  BADINT1    EQU   $02              ;INTERRUPT NOT FOUND
1A00:        0003  BADINT2    EQU   $03              ;BAD ZERO PAGE ALLOCATION
1A00:        0004  NMIHANG    EQU   $04              ;UNABLE TO LOCK NMI
1A00:        0005  EVQOVFL    EQU   $05              ;EVENT QUEUE OVERFLOW
1A00:        0006  STKOVFL    EQU   $06              ;STACK OVERFLOW
1A00:              *
1A00:        0007  BADSYSCALL EQU   $07              ;DMGR DETECTED INVALID REQUEST CODE
1A00:        0008  DEV.OVFLOW EQU   $08              ;DMGR - TOO MANY DEVICE HANDLERS
1A00:        0009  MEM2SML    EQU   $09              ;MEMORY SIZE < 64K
1A00:        000A  VCBERR     EQU   $0A              ;VOLUME CONTROL BLOCK NOT USABLE (BFMGR)
1A00:        000B  FCBERR     EQU   $0B              ;FILE CONTROL BLOCK CRASHED
1A00:        000C  ALCERR     EQU   $0C              ;ALLOCATION BLOCKS INVALID
1A00:        000E  TOOLONG    EQU   $0E              ;PATHNAME BUFFER OVERFLOW
1A00:        000F  BADBUFNUM  EQU   $0F              ;INVALID BUFFER NUMBER
1A00:        0010  BADBUFSIZ  EQU   $10              ;INVALID BUFFER SIZE (=0 OR >16K)
```

```
1A00:              *
1A00:              *  SYSTEM ERROR NUMBERS
1A00:              *
1A00:              * - SYSTEM CALL MANAGER
1A00:              *
1A00:        0001  BADSCNUM   EQU   $01              ;BAD SYSTEM CALL NUMBER
1A00:        0002  BADCZPAGE  EQU   $02              ;BAD CALLER'S ZPAGE (MUST=$1A)
1A00:        0003  BADXBYTE   EQU   $03              ;BITS  6..4 <> 0
1A00:        0004  BADSCPCNT  EQU   $04              ;BAD SYSTEM CALL PARM COUNT
1A00:        0005  BADSCBNDS  EQU   $05              ;SYS CALL PARM ADR
1A00:              *
1A00:              * - DEVICE MANAGER
1A00:              *
1A00:        0010  NODNAME    EQU   $10              ;DEVICE NAME NOT FOUND
1A00:        0011  BADDNUM    EQU   $11              ;INVALID DEV.NUM PARM
1A00:              *
1A00:              * - DEVICE HANDLERS (STANDARD ERRORS)
1A00:              *
1A00:        0020  XREQCODE   EQU   $20              ;INVALID REQUEST CODE
1A00:        0021  XCTLCODE   EQU   $21              ;INVALID CONTROL/STATUS CODE
1A00:        0022  XCTLPARM   EQU   $22              ;INVALID CONTROL/STATUS PARM
1A00:        0023  XNOTOPEN   EQU   $23              ;DEVICE NOT OPEN
1A00:        0024  XNOTAVAIL  EQU   $24              ;DEVICE NOT AVAILABLE
1A00:        0025  XNORESRC   EQU   $25              ;UNABLE TO OBTAIN RESOURCE
1A00:        0026  XBADOP     EQU   $26              ;INVALID OPERATION
1A00:        0027  XIOERROR   EQU   $27              ;I/O ERROR
1A00:              *
1A00:        0028  XNODRIVE   EQU   $28              ;NO DRIVE CONNECTED
1A00:        002B  XNOWRITE   EQU   $2B              ;DEVICE WRITE PROTECTED
1A00:        002C  XBYTECNT   EQU   $2C              ;BYTE COUNT <> A MULTIPLE OF 512
1A00:        002D  XBLKNUM    EQU   $2D              ;BLOCK NUMBER TOO LARGE
1A00:        002E  XDISKSW    EQU   $2E              ;DISK MEDIA HAS BEEN SWITCHED
1A00:              *
1A00:              * - NOTE: ERROR CODES $30-$3F HAVE BEEN RESERVED FOR DEVICE
1A00:              *   HANDLER SPECIFIC ERRORS
1A00:              *
1A00:              *
1A00:              * - FILE MANAGER
1A00:              *
1A00:        0040  BADPATH    EQU   $40              ;PATHNAME, INVALID SYNTAX
1A00:        0041  CFCBFULL   EQU   $41              ;CHAR FILE CTRL BLOCK TABLE FULL
1A00:        0042  FCBFULL    EQU   $42              ;BLOCK FILE CTRL BLOCK TABLE FULL
1A00:        0043  BADREFNUM  EQU   $43              ;INVALID REF.NUM PARM
1A00:        0044  PATHNOTFND EQU   $44              ;PATHNAME NOT FOUND
1A00:        0045  VNFERR     EQU   $45              ;VOLUME NOT FOUND
1A00:        0046  FNFERR     EQU   $46              ;FILE NOT FOUND
1A00:        0047  DUPERR     EQU   $47              ;DUPLICATE FILE NAME ERROR
1A00:        0048  OVRERR     EQU   $48              ;NOT ENOUGH DISK SPACE FOR PREALLOCATION
1A00:        0049  DIRFULL    EQU   $49              ;DIRECTORY FULL ERROR
1A00:        004A  CPTERR     EQU   $4A              ;FILE INCOMPATIBLE SOS VERSION
1A00:        004B  TYPERR     EQU   $4B              ;NOT CURRENTLY SUPPORTED FILE TYPE
1A00:        004C  EOFERR     EQU   $4C              ;POSITION ATTEMPTED BEYOND END OF FILE
1A00:        004D  POSNERR    EQU   $4D              ;ILLEGAL POSITION (L.T. 0 OR G.T. $FFFFFF)
1A00:        004E  ACCSERR    EQU   $4E              ;FILE ACCESS R/W REQUEST CONFLICTS WITH ATTRIBUTES
1A00:        004F  BTSERR     EQU   $4F              ;USER SUPPLIED BUFFER TOO SMALL
1A00:        0050  FILBUSY    EQU   $50              ;EITHER WRITE WAS REQUESTED OR WRITE ACCESS ALREADY ALLOCATED
```

```
1A00:        0051  DIRERR     EQU   $51              ;DIRECTORY ERROR
1A00:        0052  NOTSOS     EQU   $52              ;NOT A SOS DISKETTE
1A00:        0053  BADLSTCNT  EQU   $53              ;INVALID VALUE IN LIST PARAMETER
1A00:        0054  OUTOFMEM   EQU   $54              ;OUT OF FREE MEMORY FOR BUFFER
1A00:        0055  BUFTBLFULL EQU   $55              ;BUFFER TABLE FULL
1A00:        0056  BADSYSBUF  EQU   $56              ;INVALID SYSBUF PARAMETER
1A00:        0057  DUPVOL     EQU   $57              ;SON A BITCH GOT TWO VOLUMES OF SAME ROOT NAME!!!
1A00:        0058  NOTBLKDEV  EQU   $58
1A00:        0059  LVLERR     EQU   $59              ;INVALID FILE LEVEL
1A00:        005A  BITMAPADR  EQU   $5A
1A00:        0020  BACKBIT    EQU   $20              ; MASK FOR BACKUP BIT
1A00:              *
1A00:              * - UTILITY MANAGER
1A00:              *
1A00:        0070  BADJMODE   EQU   $70              ;INVALID JOYSTICK REQUEST
1A00:              *
1A00:              * - MEMORY MANAGER
1A00:              *
1A00:        00E0  BADBKPG    EQU   $E0              ;INVALID BANK/PAGE PAIR
1A00:        00E1  SEGRQDN    EQU   $E1              ;SEGMENT REQUEST DENIED
1A00:        00E2  SEGTBLFULL EQU   $E2              ;SEGMENT TABLE FULL
1A00:        00E3  BADSEGNUM  EQU   $E3              ;INVALID SEGMENT NUMBER
1A00:        00E4  SEGNOTFND  EQU   $E4              ;SEGMENT NOT FOUND
1A00:        00E5  BADSRCHMODE EQU  $E5              ;INVALID SEARCH MODE PARM
1A00:        00E6  BADCHGMODE EQU   $E6              ;INVALID CHANGE MODE PARM
1A00:        00E7  BADPGCNT   EQU   $E7              ;INVALID PAGE COUNT PARM
1A00:        1A00             ORG   SYSGLOB+$100
1A00:00 B8                    DW    $B800            ;KERNEL TARGET ADDRESS
1A02:C0 47                    DW    $47C0            ;  AND LENGTH
```

```
N004E ACCSERR       N000C ALCERR        X0001 ALLOCSIR       0020 BACKBIT
N19D2 BACKMASK      N00E0 BADBKPG       N0001 BADBRK        N000F BADBUFNUM
N0010 BADBUFSIZ     N00E6 BADCHGMODE    N0002 BADCZPAGE     N0011 BADDNUM
N0002 BADINT1       N0003 BADINT2       N0070 BADJMODE      N0053 BADLSTCNT
N0040 BADPATH       N00E7 BADPGCNT      N0043 BADREFNUM     N0005 BADSCBNDS
N0001 BADSCNUM      N0004 BADSCPCNT     N00E3 BADSEGNUM     N00E5 BADSRCHMODE
N0056 BADSYSBUF     N0007 BADSYSCALL    N0003 BADXBYTE      N005A BITMAPADR
N004F BTSERR        N0055 BUFTBLFULL    N19C8 CEVPRI        N0041 CFCBFULL
 19D3 CLRBMASK      X000E COLDSTRT      N004A CPTERR        N1B00 CSPAGE
N1600 CXPAGE        N1A00 CZPAGE        N1981 DBUGBRK       X0002 DEALCSIR
N0008 DEV.OVFLOW    N0051 DIRERR        N0049 DIRFULL       N0047 DUPERR
N0057 DUPVOL        N1908 E1908         N004C EOFERR        N0005 EVQOVFL
N000B FCBERR        N0042 FCBFULL       N0050 FILBUSY       N0046 FNFERR
X000A GETBUFADR     N1907 GRSIZE        N19CB IRQCNTR       N1985 KYBDNMI
N0059 LVLERR        N0009 MEM2SML       N1900 MEMSIZE       N19CD NMICNTR
X000D NMICONT       X000C NMIDBUG       X0003 NMIDSBL       X0004 NMIENBL
 198F NMIEXIT       N1903 NMIFLAG       N0004 NMIHANG       N198B NMISPSV
N0010 NODNAME       N0058 NOTBLKDEV     N0052 NOTSOS        N0054 OUTOFMEM
N0048 OVRERR        N0044 PATHNOTFND    N004D POSNERR       N19D1 QEV.LAST
N19D0 QEV.THIS      N19CF QEVTEMP       X0005 QUEEVENT      X000B RELBUF
X0009 REQBUF        N1906 SCRNMODE      N19F6 SDEATH.REGS   N00E4 SEGNOTFND
N00E1 SEGRQDN       N00E2 SEGTBLFULL    X0006 SELC800       N1980 SERR
N19CA SIRARGSIZ     N19C9 SIRTEMP       ?19D2 SOSQUIT       N0013 SOSVERL
N1990 SOSVER        N0100 SSPAGE        N0006 STKOVFL       N1902 SUSPFLSH
N1400 SXPAGE        N1901 SYSBANK       X0007 SYSDEATH      X0008 SYSERR
 1900 SYSGLOB       N1800 SZPAGE        N000E TOOLONG       N004B TYPERR
 1910 USERNMI       N000A VCBERR        N0045 VNFERR        N0026 XBADOP
N002D XBLKNUM       N002C XBYTECNT      N0021 XCTLCODE      N0022 XCTLPARM
N002E XDISKSW       N0027 XIOERROR      N0028 XNODRIVE      N0025 XNORESRC
N0024 XNOTAVAIL     N0023 XNOTOPEN      N002B XNOWRITE      N0020 XREQCODE
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   359
** FREE SPACE PAGE COUNT   83
```

```
SOURCE   FILE #01 =>BFM.INIT2.SRC
 INCLUDE FILE #02 =>SOSORG
```

```
0000:                            REL
0000:                            INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955   BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52   BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E   BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
B800:        B800              ORG   ORGBFMI
B800:                          MSB   OFF
B800:               ************************************************************
B800:               *          COPYRIGHT (C) APPLE COMPUTER INC.  1980
B800:               *                    ALL RIGHTS RESERVED
B800:               ************************************************************
B800:               *
B800:               * BLOCK FILE MANAGER INIT2
B800:               *
B800:               *  SECONDARY INITIALIZATION ROUTINE FOR BLOCK FILE MANAGER
B800:               *
B800:               * MODIFIED: 03/25/81 TO UTILIZE NEW
B800:               *   DISK DRIVER'S SEEKDSK3 ROUTINE.
B800:               *  CHANGES MARKED BY 'D3RRA81084'
B800:               *
B800:               * MODIFIED: 08/19/81 TO WORK WITH NEW
B800:               *   SOSLDR MODULE.
B800:               ************************************************************
B800:               *
B800:        B801              ENTRY BFM.INIT2
B800:               *
B800:               *EXTRN I.BASE.P ; ENTRY IN SOSLDR
B800:        0000              EXTRN SYSBANK
B800:        0000              EXTRN SXPAGE
B800:        0000              EXTRN CZPAGE
B800:        0000              EXTRN SEEKDSK3         ;IN DISKDH/D3RRA81084
B800:        0000              EXTRN NMIDSBL          ;/D3RRA81084
B800:        0002   I.BASE.P   EQU   $2
```

```
B800:               *
B800:               * CONSTANTS
B800:               *
B800:        B800   KERNEL.BASE EQU  $B800            ; BASE ADDRESS OF SOS KERNEL
B800:        00A0   ROMID      EQU   $A0              ;$F1B9 OF NEW ROM/D3RRA81084
B800:        0060   SLOT       EQU   $60
B800:        0009   BEGTRK     EQU   $9
B800:        0002   BEGSECT    EQU   $2
B800:        0006   ENDSECT    EQU   $6
B800:               *
B800:               * ZERO PAGE
B800:               *
B800:        0099   TRACK      EQU   $99
B800:        0098   SECTOR     EQU   $98
B800:        009A   VOLUME     EQU   $9A
B800:        00E0   KEY        EQU   $E0              ; THRU $E7
B800:        00E8   PREV.K     EQU   KEY+$8
B800:        00E9   XIDX       EQU   KEY+$9
B800:        00EA   I          EQU   KEY+$A           ; & $B
B800:               *
B800:               * ROM ROUTINES
B800:               *
B800:        F1B9   RDADR      EQU   $F1B9            ;REV1
B800:        F1BD   RDADRX     EQU   $F1BD            ;REV0
B800:               *
B800:               * HARDWARE LOCATIONS
B800:               *
B800:        FFDF   E.REG      EQU   $FFDF
B800:        FFEF   B.REG      EQU   $FFEF
B800:        C089   MOTORON    EQU   $C089
B800:        C088   MOTOROFF   EQU   $C088
```

```
B800:                  ************************************************************
B800:                  *
B800:                  * BFM.INIT2 ENTRY POINT
B800:                  *
B800:                  ************************************************************
B800:                  *
B800:FE                STATE      DFB   $FE              ; FF=1ST ENTRY, 0=2ND ENTRY, 1=PROT
B801:                  *
B801:         B801     BFM.INIT2  EQU   *
B801:EE 00 B8                     INC   STATE
B804:30 13    B819                BMI   BFMI050
B806:20 9D B8                     JSR   GETK
B809:AD 9B B8                     LDA   RETRY
B80C:F0 0D    B81B                BEQ   BADNEWS
B80E:90 09    B819                BCC   BFMI050
B810:20 00 00                     JSR   NMIDSBL
B813:20 1D B8                     JSR   DC
B816:EE 00 B8                     INC   STATE
B819:18                BFMI050    CLC
B81A:60                           RTS
B81B:38                BADNEWS    SEC                    ; I/O ERROR
B81C:60                           RTS
```

```
B81D:               ************************************************************
B81D:               *
B81D:               * DECODE SUBROUTINE
B81D:               *
B81D:               * TO ENCODE:
B81D:               *    E0.E8:        - INIT KEY  & PREV.K
B81D:               *    B84E:4C 64 B8 - JUMPS AROUND INTERP'S 3 BYTE OVERWRITE
B81D:               *    1A02.1A03:    - NEW INTERP'S LOAD ADR (LO,HII)
B81D:               *    B81DG:        - JSR FROM MONITOR
B81D:               *
B81D:               ************************************************************
B81D:        B81D   DC         EQU   *
B81D:AD EF FF                  LDA   B.REG            ; SAVE BANK REGISTER
B820:48                        PHA
B821:AD 00 00                  LDA   SYSBANK          ;    AND SWITCH TO SYSTEM BANK
B824:8D EF FF                  STA   B.REG
B827:18                        CLC                    ; FETCH LOADER'S INTERPRETER POINTER
B828:AD 02 00                  LDA   CZPAGE+I.BASE.P
B82B:69 03                     ADC   #3
B82D:85 EA                     STA   I
B82F:48                        PHA
B830:AD 03 00                  LDA   CZPAGE+I.BASE.P+1
B833:69 00                     ADC   #0
B835:85 EB                     STA   I+1
B837:48                        PHA
B838:A9 00                     LDA   #0
B83A:8D EB 00                  STA   SXPAGE+I+1
B83D:               *
B83D:A4 EA                     LDY   I                ; ALIGN I PTR TO PAGE BOUNDARY
B83F:A9 00                     LDA   #0
B841:85 EA                     STA   I
B843:85 E8                     STA   PREV.K
B845:               *
B845:20 69 B8                  JSR   DCLOOP           ; DECODE
B848:               *
B848:68                        PLA                    ; RETRIEVE LOADER'S INTERPRETER POINTER
B849:85 EB                     STA   I+1
B84B:68                        PLA
B84C:85 EA                     STA   I
B84E:               *
B84E:A0 01                     LDY   #1               ; REPOSITION LOADER'S INTERPRETER POINTER (PUT ENCODE JMP HERE)
B850:B1 EA                     LDA   (I),Y
B852:8D 02 00                  STA   CZPAGE+I.BASE.P
B855:C8                        INY
B856:B1 EA                     LDA   (I),Y
B858:8D 03 00                  STA   CZPAGE+I.BASE.P+1
B85B:               *
B85B:A0 02                     LDY   #2               ; WALK ON INTERPRETER'S FIRST INSTRUCTION (3 BYTES)
B85D:A9 00                     LDA   #0
B85F:91 EA          DCA        STA   (I),Y
B861:88                        DEY
B862:10 FB   B85F              BPL   DCA
B864:68                        PLA                    ; RESTORE BANK REGISTER                 (ENCODE JMP JUMPS TO
HERE)
B865:8D EF FF                  STA   B.REG
B868:60                        RTS
```

```
B869:              ************************************************************
B869:              *
B869:              * DECODE LOOP SUBROUTINE
B869:              *
B869:              ************************************************************
B869:        B869  DCLOOP     EQU   *
B869:A2 07                    LDX   #7               ; SHIFT LEFT ONE BIT
B86B:18                       CLC
B86C:A5 E0                    LDA   KEY
B86E:10 01   B871             BPL   DC1
B870:38                       SEC
B871:36 E0         DC1        ROL   KEY,X
B873:CA                       DEX
B874:10 FB   B871             BPL   DC1
B876:              *
B876:98            DC2        TYA
B877:29 07                    AND   #7
B879:49 02                    EOR   #2
B87B:AA                       TAX
B87C:B5 E0                    LDA   KEY,X
B87E:48                       PHA
B87F:29 07                    AND   #7
B881:AA                       TAX
B882:68                       PLA
B883:18                       CLC
B884:65 E8                    ADC   PREV.K
B886:18                       CLC
B887:75 E0                    ADC   KEY,X
B889:85 E8                    STA   PREV.K
B88B:51 EA                    EOR   (I),Y            ; DECODE BYTE
B88D:91 EA                    STA   (I),Y            ; AND PUT IT BACK
B88F:C8                       INY
B890:D0 E4   B876             BNE   DC2
B892:E6 EB                    INC   I+1
B894:A5 EB                    LDA   I+1
B896:C9 B8                    CMP   #<KERNEL.BASE
B898:90 CF   B869             BCC   DCLOOP
B89A:60                       RTS
```

```
B89B:              ************************************************************
B89B:              *
B89B:              * GETKEY SUBROUTINE
B89B:              *
B89B:              ************************************************************
B89B:              *
B89B:0B            RETRY      DFB   10+1             ;TEN RETRIES
B89C:        0001  OURTRACK   DS    1                ;CURRENT TRACK/D3RRA81084
B89D:              *
B89D:        B89D  GETK       EQU   *
B89D:A2 07                    LDX   #7
B89F:86 E9                    STX   XIDX
B8A1:A2 60                    LDX   #SLOT
B8A3:BD 89 C0                 LDA   MOTORON,X        ;ENSURE MOTOR STAYS ON
B8A6:AD DF FF                 LDA   E.REG            ; SELECT 1MHZ, ROM
B8A9:09 83                    ORA   #$83
B8AB:8D DF FF                 STA   E.REG
B8AE:              *
B8AE:              * NOTE: THE SEEKDSK3 ROUTINE HAS THESE /D3RRA81084
B8AE:              *  CAVEATS: 1MHZ MODE, MOTOR IS ON, /D3RRA81084
B8AE:              *  DRIVE CURRENTLY SELECTED, ROM+I/O ENABLED! /D3RRA81084
B8AE:              *
B8AE:A9 09         GETK010    LDA   #BEGTRK
B8B0:8D 9C B8                 STA   OURTRACK         ;WHERE WE SEEK TO /D3RRA81084
B8B3:20 00 00                 JSR   SEEKDSK3         ;HAVE DISKDH SEEK FOR US /D3RRA81084
B8B6:A2 60         GETK020    LDX   #SLOT
B8B8:20 05 B9                 JSR   DOREAD           ;FIND A SECTOR HEADER
B8BB:B0 5D   B91A             BCS   IOERROR          ;=>RETRY IF BAD
B8BD:A5 98                    LDA   SECTOR           ;WHERE ARE WE?
B8BF:C9 02                    CMP   #BEGSECT         ;AT THE RIGHT PLACE?
B8C1:D0 F3   B8B6             BNE   GETK020          ;=>NO, GET THERE
B8C3:              *
B8C3:A2 01         GETK100    LDX   #1
B8C5:20 25 B9                 JSR   WAIT             ; (X * 1284) + 15 MILISECONDS
B8C8:A6 E9                    LDX   XIDX
B8CA:A5 9A                    LDA   VOLUME
B8CC:95 E0                    STA   KEY,X
B8CE:C6 E9                    DEC   XIDX
B8D0:30 14   B8E6             BMI   ENUFF
B8D2:EE 9C B8                 INC   OURTRACK         ;BUMP FOR NEXT TRACK /D3RRA81084
B8D5:AD 9C B8                 LDA   OURTRACK         ;WHERE TO GO /D3RRA81084
B8D8:A2 60                    LDX   #SLOT
B8DA:20 00 00                 JSR   SEEKDSK3         ;DISKDH, PLEASE SEEK ME /D3RRA81084
B8DD:A2 60                    LDX   #SLOT
B8DF:20 05 B9                 JSR   DOREAD
B8E2:90 DF   B8C3             BCC   GETK100
B8E4:B0 34   B91A             BCS   IOERROR
B8E6:              *
B8E6:A2 60         ENUFF      LDX   #SLOT
B8E8:BD 88 C0                 LDA   MOTOROFF,X
B8EB:AD DF FF                 LDA   E.REG            ; SELECT 2MHZ, RAM
B8EE:29 7C                    AND   #$7C
B8F0:8D DF FF                 STA   E.REG
```

```
B8F3:A5 98                    LDA   SECTOR
B8F5:C9 06                    CMP   #ENDSECT         ;TRACKS SYNC'ED?
B8F7:D0 08   B901             BNE   NOTPROT
B8F9:A5 E0                    LDA   KEY
B8FB:45 E1                    EOR   KEY+1
B8FD:F0 02   B901             BEQ   NOTPROT          ;IF FIRST 2 VOLS ARE EQUAL
B8FF:38                       SEC
B900:60                       RTS
B901:              *
B901:A9 00         NOTPROT    LDA   #0
B903:18                       CLC
B904:60                       RTS
B905:              *
B905:              *
B905:20 10 B9      DOREAD     JSR   WHICHROM
B908:B0 03   B90D             BCS   OLDREAD
B90A:4C B9 F1                 JMP   RDADR
B90D:4C BD F1      OLDREAD    JMP   RDADRX
B910:              *
B910:              *
B910:AD B9 F1      WHICHROM   LDA   RDADR
B913:C9 A0                    CMP   #ROMID
B915:18                       CLC
B916:F0 01   B919             BEQ   NEWROM
B918:38                       SEC
B919:60            NEWROM     RTS
B91A:              *
B91A:              *
B91A:CE 9B B8      IOERROR    DEC   RETRY
B91D:F0 03   B922             BEQ   ERR1
B91F:4C 9D B8                 JMP   GETK             ; TRY, TRY AGAIN
B922:4C E6 B8      ERR1       JMP   ENUFF            ; I/O ERROR, CLEANUP AND EXIT
B925:              *
B925:              *
B925:A0 00         WAIT       LDY   #0
B927:88            W1         DEY
B928:D0 FD   B927             BNE   W1
B92A:CA                       DEX
B92B:D0 FA   B927             BNE   W1
B92D:60                       RTS
B92E:        0400  ZZLEN      EQU   $400
B92E:        0000             IFNE  ZZLEN-LENBFMI
 S                            FAIL  2,"SOSORG        FILE IS INCORRECT FOR BFM.INIT2"
B92E:                         FIN
```

```
 FFEF B.REG          B81B BADNEWS          02 BEGSECT          09 BEGTRK
NB801 BFM.INIT2      B819 BFMI050       ?2E00 BLABFMI        3200 BLABFM
 6B52 BLABUFMG       6955 BLACFM         5E99 BLADISK3       64D9 BLADMGR
 68F4 BLAFMGR       ?2CF8 BLAGLOB       ?2AF8 BLAINIT        55C0 BLAIPL
 2000 BLALODR       ?6E6E BLAMEMMG       5466 BLAOMSG        5466 BLAPATCH
 665E BLASCMGR       6404 BLASERR        5A8B BLAUMGR       X0004 CZPAGE
 B876 DC2            B85F DCA            B869 DCLOOP         B81D DC
 B871 DC1            B905 DOREAD         FFDF E.REG            06 ENDSECT
 B8E6 ENUFF          B922 ERR1           B89D GETK          ?B8AE GETK010
 B8B6 GETK020        B8C3 GETK100          02 I.BASE.P       B91A IOERROR
   EA I              B800 KERNEL.BASE      E0 KEY            2266 LENBFM
 0400 LENBFMI        031C LENBUFMG       01FD LENCFM         056B LENDISK3
 0185 LENDMGR          61 LENFMGR       ?01B2 LENINIT        04CB LENIPL
 0AF8 LENLODR       ?0751 LENMEMMG       015A LENOMSG          00 LENPATCH
 0296 LENSCMGR         D5 LENSERR        040E LENUMGR        C088 MOTOROFF
 C089 MOTORON        B919 NEWROM        X0006 NMIDSBL        B901 NOTPROT
 B90D OLDREAD        B800 ORGBFMI        BC00 ORGBFM         F552 ORGBUFMG
 F355 ORGCFM         E899 ORGDISK3       EED9 ORGDMGR        FFBF ORGEND
 F2F4 ORGFMGR       ?18FC ORGGLOB        28F8 ORGINIT        DFC0 ORGIPL
 1E00 ORGLODR        F86E ORGMEMMG       DE66 ORGOMSG        DE66 ORGPATCH
 F05E ORGSCMGR       EE04 ORGSERR        E48B ORGUMGR        B89C OURTRACK
   E8 PREV.K         F1BD RDADRX         F1B9 RDADR          B89B RETRY
   A0 ROMID            98 SECTOR        X0005 SEEKDSK3         60 SLOT
 B800 STATE         X0003 SXPAGE        X0002 SYSBANK       ?  99 TRACK
   9A VOLUME         B927 W1             B925 WAIT           B910 WHICHROM
   E9 XIDX           0400 ZZLEN
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   339
** FREE SPACE PAGE COUNT   84
```

```
SOURCE   FILE #01 =>OPRMSG.SRC
 INCLUDE FILE #02 =>SOSORG
```

```
0000:                        REL
0000:                        INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955   BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52   BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E   BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
DE66:        DE66                          ORG   ORGOMSG
DE66:        DE66   ZZORG      EQU   *
DE66:                          MSB   OFF
DE66:               ************************************************************
DE66:               *
DE66:               *          COPYRIGHT (C) APPLE COMPUTER INC. 1981
DE66:               *                   ALL RIGHTS RESERVED
DE66:               *
DE66:               ************************************************************
DE66:               *
DE66:               *  THIS MODULE CONTAINS THE BLOCK FILE MANAGERS'S OPERATOR
DE66:               *  INTERFACE.  IT DISPLAYS A MESSAGE IN A FOUR LINE BY
DE66:               *  FOURTY COLUMN WINDOW, THEN WAITS FOR THE USER TO TOGGLE
DE66:               *  THE ALPHA-LOCK KEY BEFORE RETURNING.
DE66:               *
DE66:               *  THE VERTICAL BLANKING FLAGS AND COMPOSITE BLANKING
DE66:               *  TIMER ARE USED TO MAINTAIN THE DISPLAY.  MEMORY PAGE
DE66:               *  $02 IS USED FOR TEMPORARY STORAGE.  ON EXIT, ALL
DE66:               *  RESOURCES ARE RESTORED TO THEIR PREVIOUS STATES.
DE66:               *
DE66:               *  ENTRY POINT:  OPMSGRPLY
DE66:               *
DE66:               *  PARAMETERS:  X -- MESSAGE ADDRESS (LOW BYTE)
DE66:               *               Y -- MESSAGE ADDRESS (HIGH BYTE)
DE66:               *     (THE MESSAGE MUST RESIDE IN THE CURRENT BANK)
DE66:               *
DE66:               *  RESULT:  A -- RESPONSE KEYSTROKE
DE66:               *           X, Y -- UNDEFINED
DE66:               *
DE66:               ************************************************************
DE66:               *
DE66:               *
DE66:        DE66              ENTRY OPMSGRPLY
DE66:               *
DE66:        0000              EXTRN SCRNMODE
```

```
DE66:               *
DE66:               *  HARDWARE EQUATES
DE66:               *
DE66:        FFD0   Z.REG      EQU   $FFD0
DE66:        FFDF   E.REG      EQU   $FFDF
DE66:               *
DE66:        C008   KBPORT     EQU   $C008
DE66:               *
DE66:        C040   BELL       EQU   $C040
DE66:               *
DE66:        C050   VM0        EQU   $C050
DE66:        C052   VM1        EQU   $C052
DE66:        C054   VM2        EQU   $C054
DE66:        C056   VM3        EQU   $C056
DE66:               *
DE66:        FFE8   E.T2       EQU   $FFE8
DE66:        FFEB   E.ACR      EQU   $FFEB
DE66:        FFEC   E.PCR      EQU   $FFEC
DE66:        FFED   E.IFR      EQU   $FFED
DE66:        FFEE   E.IER      EQU   $FFEE
DE66:               *
DE66:               *  ZERO PAGE DECLARATIONS
DE66:               *
0000:                          DSECT
0000:        0200   ZPBASE     EQU   $200
0000:        0000              ORG   $0000             ;ZERO PAGE DECLARATIONS
0000:        0002   MSGPTR     DS    2                 ;MESSAGE POINTER
0002:        0001   MSGIDX     DS    1
0003:               *
0003:        0001   SCRNIDX    DS    1
0004:        0002   SCRNPTR    DS    2
0006:        0002   DATAPTR    DS    2
0008:        00A0   DATABUF    DS    160
00A8:               *
00A8:        0001   SV.ZREG    DS    1
00A9:        0001   SV.EREG    DS    1
00AA:        0001   SV.SMODE   DS    1
00AB:        0001   SV.EACR    DS    1
00AC:        0001   SV.EPCR    DS    1
00AD:        0001   SV.EIER    DS    1
00AE:               *
00AE:        0001   FLAG       DS    1
DE66:                          DEND
```

```
DE66:        DE66  OPMSGRPLY  EQU   *
DE66:              *
DE66:              *
DE66:              *  SAVE CURRENT VALUES AND SET UP ZERO PAGE,
DE66:              *  ENVIRONMENT, SCREEN MODE, AND E.6522 REGISTERS.
DE66:              *
DE66:08                       PHP
DE67:78                       SEI
DE68:AD D0 FF                 LDA   Z.REG
DE6B:8D A8 02                 STA   ZPBASE+SV.ZREG   ;SAVE ZERO PAGE
DE6E:A9 02                    LDA   #<ZPBASE
DE70:8D D0 FF                 STA   Z.REG
DE73:86 00                    STX   MSGPTR           ;SAVE MESSAGE ADDRESS
DE75:84 01                    STY   MSGPTR+1
DE77:AD DF FF                 LDA   E.REG
DE7A:85 A9                    STA   SV.EREG          ;SAVE ENVIRONMENT
DE7C:29 5F                    AND   #$5F
DE7E:09 40                    ORA   #$40
DE80:8D DF FF                 STA   E.REG            ;SCREEN OFF, I/O SPACE ON
DE83:AD 00 00                 LDA   SCRNMODE
DE86:85 AA                    STA   SV.SMODE         ;SAVE SCREEN MODE
DE88:A9 00                    LDA   #$00
DE8A:8D 00 00                 STA   SCRNMODE
DE8D:2C 50 C0                 BIT   VM0              ;SET 40 COLUMN
DE90:2C 52 C0                 BIT   VM1              ;  BLACK & WHITE TEXT
DE93:2C 54 C0                 BIT   VM2
DE96:2C 56 C0                 BIT   VM3
DE99:AE EB FF                 LDX   E.ACR
DE9C:8A                       TXA
DE9D:29 20                    AND   #$20
DE9F:85 AB                    STA   SV.EACR          ;SAVE AUXILIARY CONTROL REG
DEA1:8A                       TXA
DEA2:09 20                    ORA   #$20
DEA4:8D EB FF                 STA   E.ACR            ;SET UP BL TIMER
DEA7:AE EC FF                 LDX   E.PCR
DEAA:8A                       TXA
DEAB:29 F0                    AND   #$F0
DEAD:85 AC                    STA   SV.EPCR          ;SAVE PERIPHERAL CONTROL REG
DEAF:8A                       TXA
DEB0:29 0F                    AND   #$0F
DEB2:09 60                    ORA   #$60
DEB4:8D EC FF                 STA   E.PCR            ;SET UP VBL FLAGS
DEB7:AD EE FF                 LDA   E.IER
DEBA:29 38                    AND   #$38
DEBC:8D EE FF                 STA   E.IER            ;MASK VBL & BL INTERRUPTS
DEBF:85 AD                    STA   SV.EIER          ;SAVE INTERRUPT MASKS
DEC1:28                       PLP
DEC2:              *
DEC2:              *
DEC2:              *  SAVE SCREEN DATA AND CLEAR MESSAGE WINDOW
DEC2:              *
DEC2:A2 03                    LDX   #3
DEC4:20 A7 DF      OPR010     JSR   SETPTRS
DEC7:A0 27                    LDY   #39
DEC9:B1 04         OPR020     LDA   (SCRNPTR),Y      ;SAVE SCREEN DATA
DECB:91 06                    STA   (DATAPTR),Y
```

```
DECD:A9 A0                    LDA   #$A0
DECF:91 04                    STA   (SCRNPTR),Y      ;BLANK SCREEN
DED1:88                       DEY
DED2:10 F5   DEC9             BPL   OPR020
DED4:CA                       DEX
DED5:10 ED   DEC4             BPL   OPR010
DED7:              *
DED7:              *
DED7:              *  MOVE MESSAGE TO WINDOW
DED7:              *
DED7:2C 40 C0                 BIT   BELL
DEDA:A2 00                    LDX   #$00
DEDC:86 02                    STX   MSGIDX
DEDE:20 A7 DF      OPR100     JSR   SETPTRS
DEE1:A0 00                    LDY   #$00
DEE3:84 03                    STY   SCRNIDX
DEE5:A4 02         OPR110     LDY   MSGIDX
DEE7:E6 02                    INC   MSGIDX
DEE9:B1 00                    LDA   (MSGPTR),Y       ;SET UP MESSAGE
DEEB:F0 F8   DEE5             BEQ   OPR110
DEED:30 15   DF04             BMI   OPR200
DEEF:C9 0D                    CMP   #$0D
DEF1:F0 0C   DEFF             BEQ   OPR120
DEF3:A4 03                    LDY   SCRNIDX
DEF5:E6 03                    INC   SCRNIDX
DEF7:09 80                    ORA   #$80
DEF9:91 04                    STA   (SCRNPTR),Y
DEFB:C0 27                    CPY   #39
DEFD:90 E6   DEE5             BCC   OPR110
DEFF:E8            OPR120     INX
DF00:E0 04                    CPX   #4
DF02:90 DA   DEDE             BCC   OPR100
DF04:              *
DF04:              *
DF04:              *  DISPLAY MESSAGE UNTIL ALPHA-LOCK KEY TOGGLES
DF04:              *
DF04:A0 02         OPR200     LDY   #2
DF06:AD 08 C0                 LDA   KBPORT
DF09:29 08                    AND   #$08
DF0B:85 AE                    STA   FLAG
DF0D:20 77 DF      OPR210     JSR   VIDEO
DF10:AD 08 C0                 LDA   KBPORT
DF13:29 08                    AND   #$08
DF15:C5 AE                    CMP   FLAG
DF17:F0 F4   DF0D             BEQ   OPR210
DF19:85 AE                    STA   FLAG
DF1B:88                       DEY
DF1C:D0 EF   DF0D             BNE   OPR210
DF1E:              *
DF1E:              *
DF1E:              *  RESTORE PREVIOUS CONTENTS OF WINDOW
DF1E:              *
DF1E:A2 03                    LDX   #3
DF20:20 A7 DF      OPR400     JSR   SETPTRS
DF23:A0 27                    LDY   #39
DF25:B1 06         OPR410     LDA   (DATAPTR),Y
```

```
DF27:91 04                    STA   (SCRNPTR),Y
DF29:88                       DEY
DF2A:10 F9   DF25             BPL   OPR410
DF2C:CA                       DEX
DF2D:10 F1   DF20             BPL   OPR400
DF2F:              *
DF2F:              *
DF2F:              *  RESTORE E.6522, SCREEN MODE, ENVIRONMENT, & ZERO PAGE
DF2F:              *  THEN RETURN TO CALLER
DF2F:              *
DF2F:08                       PHP
DF30:78                       SEI
DF31:AD EB FF                 LDA   E.ACR
DF34:29 DF                    AND   #$DF
DF36:05 AB                    ORA   SV.EACR          ;RESTORE AUXILIARY CONTROL REG
DF38:8D EB FF                 STA   E.ACR
DF3B:AD EC FF                 LDA   E.PCR
DF3E:29 0F                    AND   #$0F
DF40:05 AC                    ORA   SV.EPCR          ;RESTORE PERIPHERAL CONTROL REG
DF42:8D EC FF                 STA   E.PCR
DF45:A5 AD                    LDA   SV.EIER          ;RESTORE INTERRUPT ENABLE REG
DF47:09 80                    ORA   #$80
DF49:8D EE FF                 STA   E.IER
DF4C:A5 AA                    LDA   SV.SMODE         ;RESTORE SCREEN MODE
DF4E:8D 00 00                 STA   SCRNMODE
DF51:4A                       LSR   A
DF52:90 03   DF57             BCC   OPR500
DF54:2C 51 C0                 BIT   VM0+1            ;RESTORE VIDEO MODE
DF57:4A            OPR500     LSR   A
DF58:90 03   DF5D             BCC   OPR510
DF5A:2C 53 C0                 BIT   VM1+1
DF5D:4A            OPR510     LSR   A
DF5E:90 03   DF63             BCC   OPR520
DF60:2C 55 C0                 BIT   VM2+1
DF63:2C 00 00      OPR520     BIT   SCRNMODE
DF66:50 03   DF6B             BVC   OPR530
DF68:2C 57 C0                 BIT   VM3+1
DF6B:A5 A9         OPR530     LDA   SV.EREG          ;RESTORE ENVIRONMENT
DF6D:8D DF FF                 STA   E.REG
DF70:A5 A8                    LDA   SV.ZREG          ;RESTORE ZERO PAGE
DF72:8D D0 FF                 STA   Z.REG
DF75:28                       PLP
DF76:60                       RTS
```

```
DF77:              ************************************************************
DF77:              *
DF77:              *  SUBROUTINE VIDEO
DF77:              *
DF77:              *  THIS SUBROUTINE POLLS THE VERTICAL-BLANKING AND
DF77:              *  COMPOSITE-BLANKING-TIMER FLAGS AND TURNS THE SCREEN
DF77:              *  OFF AND ON SO THAT ONLY THE MESSAGE WINDOW WILL BE
DF77:              *  DISPLAYED.
DF77:              *
DF77:              *  THE E.6522 MUST BE INITIALIZED SO THAT E.CB2 FLAGS THE
DF77:              *  POSITIVE EDGE OF VBL AND E.T2 COUNTS BL PULSES.  THE
DF77:              *  INTERRUPTS MUST BE MASKED AND THE PROPER COUNT MUST
DF77:              *  ALREADY BE STORED IN THE LOW ORDER BYTE OF E.T2.
DF77:              *
DF77:              *  ENTRY:  VIDEO
DF77:              *
DF77:              *  PARAMETERS:  INTERRUPT SYSTEM DISABLED
DF77:              *
DF77:              *  EXIT:  A -- UNDEFINED
DF77:              *         X, Y -- PRESERVED
DF77:              *
DF77:              ************************************************************
DF77:              *
DF77:        DF77  VIDEO      EQU   *
DF77:AD ED FF                 LDA   E.IFR
DF7A:29 28                    AND   #$28             ;GET VBL & BL FLAGS
DF7C:F0 28   DFA6             BEQ   VID030
DF7E:8D ED FF                 STA   E.IFR            ;CLEAR FLAGS
DF81:29 20                    AND   #$20             ;WHICH FLAG?
DF83:D0 12   DF97             BNE   VID010           ;  BL
DF85:              *
DF85:A9 1F                    LDA   #$1F
DF87:8D E8 FF                 STA   E.T2             ;SET UP BL TIMER
DF8A:A9 00                    LDA   #$00
DF8C:8D E9 FF                 STA   E.T2+1
DF8F:AD DF FF                 LDA   E.REG
DF92:09 20                    ORA   #$20             ;SET UP FOR SCREEN ON
DF94:38                       SEC
DF95:B0 06   DF9D             BCS   VID020
DF97:              *
DF97:AD DF FF      VID010     LDA   E.REG
DF9A:29 DF                    AND   #$DF             ;SET UP FOR SCREEN OFF
DF9C:18                       CLC
DF9D:              *
DF9D:8D DF FF      VID020     STA   E.REG
DFA0:A9 00                    LDA   #$00
DFA2:6A                       ROR   A
DFA3:8D 00 00                 STA   SCRNMODE
DFA6:60            VID030     RTS
```

```
DFA7:              ************************************************************
DFA7:              *
DFA7:              *  SUBROUTINE SETPTRS
DFA7:              *
DFA7:              *  THIS SUBROUTINE SETS UP THE POINTERS TO THE MESSAGE
DFA7:              *  WINDOW AND DATA SAVE AREA.
DFA7:              *
DFA7:              *  ENTRY:  SETPTRS
DFA7:              *
DFA7:              *  PARAMETERS:  X -- LINE NUMBER [0..3]
DFA7:              *
DFA7:              *  EXIT:  A -- UNDEFINED
DFA7:              *         X, Y -- PRESERVED
DFA7:              *
DFA7:              ************************************************************
DFA7:              *
DFA7:        DFA7  SETPTRS    EQU   *
DFA7:8A                       TXA
DFA8:4A                       LSR   A
DFA9:09 04                    ORA   #$04
DFAB:85 05                    STA   SCRNPTR+1
DFAD:A9 00                    LDA   #$00
DFAF:6A                       ROR   A
DFB0:85 04                    STA   SCRNPTR
DFB2:A9 00                    LDA   #<DATABUF
DFB4:85 07                    STA   DATAPTR+1
DFB6:BD BC DF                 LDA   DBUFADR,X
DFB9:85 06                    STA   DATAPTR
DFBB:60                       RTS
DFBC:              *
DFBC:        DFBC  DBUFADR    EQU   *
DFBC:08                       DFB   >0*40+DATABUF
DFBD:30                       DFB   >1*40+DATABUF
DFBE:58                       DFB   >2*40+DATABUF
DFBF:80                       DFB   >3*40+DATABUF

DFC0:                         LST   ON
DFC0:        DFC0  ZZEND      EQU   *
DFC0:        015A  ZZLEN      EQU   ZZEND-ZZORG
DFC0:        0000             IFNE  ZZLEN-LENOMSG
 S                            FAIL  2,"SOSORG        FILE IS INCORRECT FOR OPRMSG"
DFC0:                         FIN
```

```
 C040 BELL           3200 BLABFM        ?2E00 BLABFMI        6B52 BLABUFMG
 6955 BLACFM         5E99 BLADISK3       64D9 BLADMGR        68F4 BLAFMGR
?2CF8 BLAGLOB       ?2AF8 BLAINIT        55C0 BLAIPL         2000 BLALODR
?6E6E BLAMEMMG       5466 BLAOMSG        5466 BLAPATCH       665E BLASCMGR
 6404 BLASERR        5A8B BLAUMGR          08 DATABUF          06 DATAPTR
 DFBC DBUFADR        FFEB E.ACR          FFEE E.IER          FFED E.IFR
 FFEC E.PCR          FFDF E.REG          FFE8 E.T2             AE FLAG
 C008 KBPORT        ?0400 LENBFMI        2266 LENBFM         031C LENBUFMG
 01FD LENCFM         056B LENDISK3       0185 LENDMGR          61 LENFMGR
?01B2 LENINIT        04CB LENIPL         0AF8 LENLODR       ?0751 LENMEMMG
 015A LENOMSG          00 LENPATCH       0296 LENSCMGR         D5 LENSERR
 040E LENUMGR          02 MSGIDX           00 MSGPTR        NDE66 OPMSGRPLY
 DEC4 OPR010         DEC9 OPR020         DEDE OPR100         DEE5 OPR110
 DEFF OPR120         DF04 OPR200         DF0D OPR210         DF20 OPR400
 DF25 OPR410         DF57 OPR500         DF5D OPR510         DF63 OPR520
 DF6B OPR530         BC00 ORGBFM         B800 ORGBFMI        F552 ORGBUFMG
 F355 ORGCFM         E899 ORGDISK3       EED9 ORGDMGR        FFBF ORGEND
 F2F4 ORGFMGR       ?18FC ORGGLOB        28F8 ORGINIT        DFC0 ORGIPL
 1E00 ORGLODR        F86E ORGMEMMG       DE66 ORGOMSG        DE66 ORGPATCH
 F05E ORGSCMGR       EE04 ORGSERR        E48B ORGUMGR          03 SCRNIDX
X0002 SCRNMODE         04 SCRNPTR        DFA7 SETPTRS          AB SV.EACR
   AD SV.EIER          AC SV.EPCR          A9 SV.EREG          AA SV.SMODE
   A8 SV.ZREG        DF97 VID010         DF9D VID020         DFA6 VID030
 DF77 VIDEO          C050 VM0            C052 VM1            C054 VM2
 C056 VM3            FFD0 Z.REG          0200 ZPBASE         DFC0 ZZEND
 015A ZZLEN          DE66 ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   389
** FREE SPACE PAGE COUNT   84
```

```
SOURCE   FILE #01 =>IPL.SRC1
 INCLUDE FILE #02 =>SOSORG
SOURCE   FILE #03 =>IPL.SRC2
```

```
0000:                2            REL
0000:                3            INCLUDE SOSORG
0000:                1
****************************************************************************************************
0000:                2 *  SOS KERNEL MODULE ORIGINS
0000:        1E00    3 ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8    4 ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC    5 ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800    6 ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00    7 ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66    8 ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66    9 ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0   10 ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B   11 ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899   12 ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04   13 ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9   14 ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E   15 ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4   16 ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355   17 ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552   18 ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E   19 ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF   20 ORGEND     EQU   $FFBF            ; END MARKER
0000:               21
****************************************************************************************************
0000:               22 *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8   23 LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2   24 LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400   25 LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266   26 LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000   27 LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A   28 LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB   29 LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E   30 LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B   31 LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5   32 LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185   33 LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296   34 LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061   35 LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD   36 LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C   37 LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751   38 LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:               39
****************************************************************************************************
0000:               40 *      SOS BLOAD ADDRESSES
0000:        2000   41 BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8   42 BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8   43 BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00   44 BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200   45 BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466   46 BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466   47 BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0   48 BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B   49 BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99   50 BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404   51 BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9   52 BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E   53 BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4   54 BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955      BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52      BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E      BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:                 
****************************************************************************************************
DFC0:        DFC0                 ORG   ORGIPL
DFC0:        DFC0      ZZORG      EQU   *
DFC0:                             MSB   OFF
DFC0:                  ************************************************************
DFC0:                  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
DFC0:                  *                    ALL RIGHTS RESERVED
DFC0:                  ************************************************************
DFC0:                  *
DFC0:                  *  THIS MODULE IS RESPONSIBLE FOR FIELDING ALL INTERRUPTS
DFC0:                  *  AND RELAUNCHING THE INTERRUPTED CODE AFTER THE INTERRUPTS
DFC0:                  *  HAVE BEEN PROCESSED.  THE MAJOR FUNCTIONAL AREAS ARE:
DFC0:                  *
DFC0:                  *       GENERAL INTERRUPT RECEIVER
DFC0:                  *       NMI INTERRUPT RECEIVER
DFC0:                  *       DISPATCHER
DFC0:                  *       INTERRUPT ALLOCATION & DEALLOCATION
DFC0:                  *       EVENT QUEUE MANAGER
DFC0:                  *       TABLE INITIALIZATION
DFC0:                  *
DFC0:                  ************************************************************
DFC0:                  *
DFC0:                  *  SUBROUTINE ENTRY POINTS
DFC0:                  *
DFC0:        E050                 ENTRY IRQ.RCVR         ;GENERAL INTERRUPT RECEIVER
DFC0:        E1A4                 ENTRY NMI.RCVR         ;NON-MASKABLE INTRPT RCVR
DFC0:        E21D                 ENTRY DISPATCH         ;DISPATCHER
DFC0:        E2CA                 ENTRY ALLOCSIR         ;SIR ALLOCATION
DFC0:        E352                 ENTRY DEALCSIR         ;SIR DEALLOCATION
DFC0:        E3A9                 ENTRY SELC800          ;SELECT I/O EXPANSION ROM
DFC0:        E3C2                 ENTRY NMIDSBL          ;DISABLE NMI
DFC0:        E3F3                 ENTRY NMIENBL          ;ENABLE NMI
DFC0:        E3FC                 ENTRY NMIDBUG          ;NMI DEBUG ENTRY
DFC0:        E410                 ENTRY NMICONT          ;NMI DEBUG CONTINUATION
DFC0:        E41D                 ENTRY QUEEVENT         ;QUEUE AN EVENT
DFC0:                  *
DFC0:                  *  EXTERNAL SUBROUTINES & DATA
DFC0:                  *
DFC0:        0000                 EXTRN SCMGR
DFC0:        0000                 EXTRN CHKBUF
DFC0:                  *
DFC0:                  *  SYSTEM DEATH ERRORS
DFC0:                  *
DFC0:        0000                 EXTRN SYSDEATH
DFC0:        0000                 EXTRN BADBRK
DFC0:        0000                 EXTRN BADINT1
DFC0:        0000                 EXTRN BADINT2
DFC0:        0000                 EXTRN NMIHANG
DFC0:        0000                 EXTRN EVQOVFL
DFC0:        0000                 EXTRN STKOVFL
DFC0:                  *
DFC0:                  *  LINKAGE DATA FOR INITIALIZATION ROUTINES
DFC0:                  *
```

```
DFC0:        E026             ENTRY EV.QUEUE
DFC0:        0007             ENTRY EVQ.CNT
DFC0:        0006             ENTRY EVQ.SIZ
DFC0:        002A             ENTRY EVQ.LEN
DFC0:        E028             ENTRY EVQ.FREE
DFC0:        E026             ENTRY EVQ.LINK
DFC0:        DFC5             ENTRY SIRTABLE
DFC0:        0018             ENTRY SIRTBLSIZ
DFC0:        DFC4             ENTRY ZPGSTACK
DFC0:        00F8             ENTRY ZPGSTART
DFC0:              *
DFC0:              *  SYSGLOB DATA
DFC0:              *
DFC0:        0000             EXTRN SERR
DFC0:        0000             EXTRN CEVPRI           ;CALLER'S EVENT PRIORITY
DFC0:        0000             EXTRN SYSBANK          ;SYSTEM BANK
DFC0:        0000             EXTRN KYBDNMI
DFC0:        0000             EXTRN NMISPSV
DFC0:        0000             EXTRN NMIFLAG          ;NMI PENDING FLAG
DFC0:        0000             EXTRN SCRNMODE         ;CURRENT SCREEN MODE
DFC0:        0000             EXTRN SIRTEMP          ;FOR ALLOCSIR & DEALCSIR
DFC0:        0000             EXTRN SIRARGSIZ
DFC0:        0000             EXTRN IRQCNTR          ;FLASE IRQ COUNTER
DFC0:        0000             EXTRN NMICNTR          ;TWO BYTE COUNTER
DFC0:        0000             EXTRN QEVTEMP
DFC0:        0000             EXTRN QEV.THIS
DFC0:        0000             EXTRN QEV.LAST
DFC0:        0000                   EXTRN            BACKMASK
DFC0:              *
DFC0:              *  CONSTANT DECLARATIONS
DFC0:              *
DFC0:        0000  FALSE      EQU   $00
DFC0:        0001  BITON0     EQU   $01
DFC0:        0002  BITON1     EQU   $02
DFC0:        0004  BITON2     EQU   $04
DFC0:        0010  BITON4     EQU   $10
DFC0:        0020  BITON5     EQU   $20
DFC0:        0040  BITON6     EQU   $40
DFC0:        0080  BITON7     EQU   $80
DFC0:        00F7  BITOFF3    EQU   $F7
DFC0:        00EF  BITOFF4    EQU   $EF
DFC0:        00DF  BITOFF5    EQU   $DF
DFC0:        00BF  BITOFF6    EQU   $BF
DFC0:        007F  BITOFF7    EQU   $7F
DFC0:        0020  BACKBIT    EQU   $20              ; BACKUP BIT MASK
DFC0:              *
DFC0:              *  SYSTEM CONTROL REGISTERS
DFC0:              *
DFC0:        FFEF  B.REG      EQU   $FFEF            ;BANK REGISTER
DFC0:        FFDF  E.REG      EQU   $FFDF            ;ENVIRONMENT REGISTER
DFC0:        FFD0  Z.REG      EQU   $FFD0            ;ZERO PAGE REGISTER
DFC0:              *
DFC0:              *  6522 REGISTERS
DFC0:              *
DFC0:        FFDD  D.IFR      EQU   $FFDD
DFC0:        FFDE  D.IER      EQU   $FFDE
```

```
DFC0:        FFE0  112 E.IORB     EQU   $FFE0
DFC0:        FFED  113 E.IFR      EQU   $FFED
DFC0:        FFEE  114 E.IER      EQU   $FFEE
DFC0:        FFEF  115 E.IORA     EQU   $FFEF
```

```
DFC0:              *
DFC0:              *  REGISTER PRESERVATION EQUATES
DFC0:              *  FOR USE DURING INTERRUPT PROCESSING
DFC0:              *
DFC0:        0103  A.SAVE     EQU   $103
DFC0:        0104  S.SAVE     EQU   $104
DFC0:        01FF  SP.SAVE    EQU   $1FF
DFC0:        01FE  E.SAVE     EQU   $1FE
DFC0:        01FD  Z.SAVE     EQU   $1FD
DFC0:        01FC  B.SAVE     EQU   $1FC
DFC0:00            EXPNSLOT   DFB   $00              ;CURRENT I/O EXPANSION SLOT
DFC1:              *
DFC1:              *  STATUS LOCATIONS FOR INTERRUPT POLLING
DFC1:              *
DFC1:        C0F1  ACIASTAT   EQU   $C0F1
DFC1:02            ANYSLOT    DFB   BITON1
DFC2:        C065  SLOT1      EQU   $C065
DFC2:        C064  SLOT2      EQU   $C064
DFC2:20            SLOT3      DFB   BITON5
DFC3:10            SLOT4      DFB   BITON4
DFC4:              *
DFC4:              *  INTERRUPT ZERO PAGE STORAGE & EQUATES
DFC4:              *
DFC4:        00F9  SIRARGS    EQU   $F9              ;AND $FA
DFC4:        00FB  QEVARGS    EQU   $FB              ;AND $FC
DFC4:        00FD  IRQADDR    EQU   $FD              ;AND $FE
DFC4:        00FF  ZPGSP      EQU   $FF
DFC4:        00F8  ZPGSTART   EQU   $F8
DFC4:        0028  ZPGSTOP    EQU   $28
DFC4:        0020  ZPGSPACE   EQU   $20
DFC4:F8            ZPGSTACK   DFB   ZPGSTART
DFC5:              *
DFC5:              *  SYSTEM INTERNAL RESOURCE
DFC5:              *  TABLE STORAGE AND EQUATES
DFC5:              *
DFC5:        0018  SIRTBLSIZ  EQU   $18
DFC5:        0018  SIRTABLE   DS    SIRTBLSIZ
DFDD:        0018  SIRADR.L   DS    SIRTBLSIZ
DFF5:        0001  NMIADR.L   DS    1                ;MUST PRECEED SIRADR.H
DFF6:        0018  SIRADR.H   DS    SIRTBLSIZ
E00E:        0018  SIRADR.B   DS    SIRTBLSIZ
E026:              *
E026:              *  EVENT QUEUE STORAGE AND EQUATES
E026:              *
E026:        0006  EVQ.SIZ    EQU   6                ;ENTRY SIZE
E026:        0007  EVQ.CNT    EQU   $07              ;ENTRY COUNT
E026:        002A  EVQ.LEN    EQU   $2A              ;(EVQ.SIZ*EVQ.CNT)
E026:        002A  EV.QUEUE   DS    EVQ.LEN
E050:        E028  EVQ.FREE   EQU   EV.QUEUE+2       ;FIRST FREE ENTRY INDEX
E050:        E026  EVQ.LINK   EQU   EV.QUEUE+0       ;NEXT ACTIVE ENTRY INDEX
E050:        E027  EVQ.PRI    EQU   EV.QUEUE+1       ;EVENT PRIORITY
E050:        E028  EVQ.ID     EQU   EV.QUEUE+2       ;EVENT IDENTIFICATION
E050:        E029  EVQ.ADRL   EQU   EV.QUEUE+3       ;EVENT ADDRESS:  LOW BYTE
E050:        E02A  EVQ.ADRH   EQU   EV.QUEUE+4       ;EVENT ADDRESS:  HIGH BYTE
E050:        E02B  EVQ.BANK   EQU   EV.QUEUE+5       ;EVENT ADDRESS:  BANK
```

```
E050:              ************************************************************
E050:              *
E050:              *  THIS IS THE GENERAL INTERRUPT RECEIVER.  WHEN AN
E050:              *  INTERRUPT OCCURS, THE CPU PASSES CONTROL TO THE GIR
E050:              *  THROUGH THE IRQ VECTOR.  THE GIR IS RESPONSIBLE FOR
E050:              *  SAVING THE CURRENT ENVIRONMENT, SETTING UP THE SOS
E050:              *  ENVIRONMENT, AND CALLING THE APPROPRIATE CODE MODULE.
E050:              *  IF THE INTERRUPT WAS CAUSED BY A BRK, THE GIR CALLS
E050:              *  THE SYSTEM CALL MANAGER.  OTHERWISE, THE GIR POLLS THE
E050:              *  I/O DEVICES AND CALLS THE APPROPRIATE MASTER INTERRUPT
E050:              *  HANDLER.  WHEN THE SCM OR MIH RETURNS, THE GIR PASSES
E050:              *  CONTROL TO THE DISPATCHER.
E050:              *
E050:              ************************************************************
E050:              *
E050:        E050  IRQ.RCVR   EQU   *
E050:              *
E050:              *  SAVE CPU REGISTERS A, X, & Y ON CURRENT STACK
E050:              *
E050:48                       PHA
E051:8A                       TXA
E052:48                       PHA
E053:98                       TYA
E054:48                       PHA
E055:              *
E055:              *  CHECK FOR STACK OVERFLOW AND
E055:              *  SAVE INTERRUPTED STATUS IN Y REGISTER.
E055:              *
E055:BA                       TSX
E056:E0 FA                    CPX   #$FA
E058:90 05   E05F             BCC   GIR005
E05A:A9 00                    LDA   #>STKOVFL
E05C:20 00 00                 JSR   SYSDEATH
E05F:BC 04 01      GIR005     LDY   S.SAVE,X
E062:              *
E062:              *  SET UP INTERRUPT ENVIRONMENT:
E062:              *    BINARY ARITHMETIC, 2 MHZ, I/O ENABLED,
E062:              *    RAM WRITE ENABLED, PRIMARY STACK,
E062:              *    AND $F000 RAM SELECTED.  PRESERVE
E062:              *    USER STATE OF SCREEN AND RESET LOCK.
E062:              *
E062:D8                       CLD
E063:AD DF FF                 LDA   E.REG
E066:AA                       TAX
E067:29 30                    AND   #BITON5+BITON4
E069:09 44                    ORA   #BITON6+BITON2
E06B:8D DF FF                 STA   E.REG
E06E:              *
E06E:              *  IF NOT ALREADY ON PRIMARY STACK, SAVE USER'S STACK
E06E:              *  POINTER AND SET UP SOS STACK POINTER.
E06E:              *
E06E:8A                       TXA
E06F:29 04                    AND   #BITON2
E071:D0 09   E07C             BNE   GIR010
E073:8A                       TXA
E074:BA                       TSX
```

```
E075:8E FF 01                 STX   SP.SAVE
E078:A2 FE                    LDX   #>E.SAVE
E07A:9A                       TXS
E07B:AA                       TAX
E07C:              *
E07C:              *  SAVE E, Z, B, & I/O EXPANSION SLOT ON SOS STACK
E07C:              *  IF BRK THEN CALL SCMGR ELSE POLL I/O DEVICES
E07C:              *
E07C:8A            GIR010     TXA
E07D:48                       PHA
E07E:AD D0 FF                 LDA   Z.REG
E081:48                       PHA
E082:AD EF FF                 LDA   B.REG
E085:48                       PHA
E086:AD C0 DF                 LDA   EXPNSLOT
E089:48                       PHA
E08A:2C FF CF                 BIT   $CFFF
E08D:2C 20 C0                 BIT   $C020            ;RESET I/O SPACE
E090:A9 00                    LDA   #$00
E092:8D C0 DF                 STA   EXPNSLOT
E095:98                       TYA
E096:29 10                    AND   #BITON4
E098:F0 40   E0DA             BEQ   POLL.IO
E09A:              *
E09A:              *  CALL SYSTEM CALL MANAGER; ON RETURN, PUT ERROR CODE IN
E09A:              *  USER'S A REGISTER AND SET RETURN STATUS, THEN DISPATCH.
E09A:              *
E09A:BA                       TSX                    ;CHECK FOR
E09B:E0 FA                    CPX   #>B.SAVE-2       ;  REENTRANT
E09D:F0 05   E0A4             BEQ   GIR020           ;  SYSTEM CALL
E09F:A9 00                    LDA   #>BADBRK
E0A1:20 00 00                 JSR   SYSDEATH
E0A4:AD DF FF      GIR020     LDA   E.REG            ;SELECT $C000 RAM
E0A7:29 BF                    AND   #BITOFF6
E0A9:8D DF FF                 STA   E.REG
E0AC:58                       CLI                    ;ENABLE INTERRUPTS
E0AD:20 00 00                 JSR   SCMGR            ;CALL THE SYSTEM CALL MGR
E0B0:A9 20                    LDA   #BACKBIT         ; GET THE MASK
E0B2:8D 00 00                 STA   BACKMASK         ; SET IT IN SYSGLOB
E0B5:20 00 00                 JSR   CHKBUF
E0B8:78                       SEI
E0B9:AE FF 01                 LDX   SP.SAVE
E0BC:AD FD 01                 LDA   Z.SAVE
E0BF:49 01                    EOR   #BITON0          ;SET ZERO PAGE TO
E0C1:8D D0 FF                 STA   Z.REG            ;  CALLER'S STACK
E0C4:AD 00 00                 LDA   SERR
E0C7:95 03                    STA   >A.SAVE,X
E0C9:08                       PHP
E0CA:B5 04                    LDA   >S.SAVE,X
E0CC:29 7D                    AND   #$7D
E0CE:95 04                    STA   >S.SAVE,X
E0D0:68                       PLA
E0D1:29 82                    AND   #$82
E0D3:15 04                    ORA   >S.SAVE,X
E0D5:95 04                    STA   >S.SAVE,X
E0D7:4C 1D E2                 JMP   DISPATCH
```

```
E0DA:              *
E0DA:              *  SET INTERRUPT ZERO PAGE AND SOS BANK
E0DA:              *    THEN POLL I/O DEVICES
E0DA:              *
E0DA:2C EF FF      POLL.IO    BIT   E.IORA           ;VERIFY THAT 'IRQ IS LOW
E0DD:10 0B   E0EA             BPL   PIO006
E0DF:EE 00 00                 INC   IRQCNTR          ;BUMP FALSE IRQ COUNTER
E0E2:D0 03   E0E7             BNE   PIO004
E0E4:EE 01 00                 INC   IRQCNTR+1
E0E7:4C 1D E2      PIO004     JMP   DISPATCH
E0EA:A9 00         PIO006     LDA   #0               ;SET INTERRUPT ZERO PAGE
E0EC:8D D0 FF                 STA   Z.REG
E0EF:AD DF FF                 LDA   E.REG
E0F2:09 80                    ORA   #BITON7          ;FORCE 1 MHZ FOR
E0F4:8D DF FF                 STA   E.REG            ;  READING ACIA STATUS
E0F7:29 7F                    AND   #BITOFF7
E0F9:A2 01                    LDX   #$01
E0FB:AC F1 C0                 LDY   ACIASTAT         ;ANY INTERRUPT ON ACIA?
E0FE:8D DF FF                 STA   E.REG
E101:30 5C   E15F             BMI   PIO070
E103:AD ED FF                 LDA   E.IFR            ;ANY INTERRUPT ON E-6522?
E106:10 10   E118             BPL   PIO020           ;  NO
E108:2D EE FF                 AND   E.IER
E10B:A0 07                    LDY   #7
E10D:A2 02                    LDX   #$02
E10F:4A            PIO010     LSR   A                ;CHECK FLAG BITS
E110:B0 4D   E15F             BCS   PIO070
E112:E8                       INX
E113:88                       DEY
E114:D0 F9   E10F             BNE   PIO010
E116:F0 18   E130             BEQ   PIO035
E118:AD DD FF      PIO020     LDA   D.IFR            ;ANY INTERRUPT ON D-6522?
E11B:10 13   E130             BPL   PIO035
E11D:2D DE FF                 AND   D.IER
E120:2C C1 DF                 BIT   ANYSLOT          ;ANY SLOT INTERRUPT?
E123:D0 0F   E134             BNE   PIO040           ;  YES
E125:A0 07                    LDY   #7
E127:A2 09                    LDX   #$09
E129:4A            PIO030     LSR   A                ;CHECK FLAG BITS
E12A:B0 33   E15F             BCS   PIO070
E12C:E8                       INX
E12D:88                       DEY
E12E:D0 F9   E129             BNE   PIO030
E130:A2 10         PIO035     LDX   #$10             ;INTERRUPT NOT FOUND
E132:D0 1E   E152             BNE   PIO050
E134:A2 11         PIO040     LDX   #$11
E136:2C 65 C0                 BIT   SLOT1            ;SLOT 1?
E139:10 24   E15F             BPL   PIO070
E13B:E8                       INX
E13C:2C 64 C0                 BIT   SLOT2            ;SLOT 2?
E13F:10 1E   E15F             BPL   PIO070
E141:AD EF FF                 LDA   E.IORA
E144:E8                       INX
E145:2C C2 DF                 BIT   SLOT3            ;SLOT 3?
E148:F0 15   E15F             BEQ   PIO070
E14A:E8                       INX
```

```
E14B:2C C3 DF                 BIT   SLOT4            ;SLOT 4?
E14E:F0 0F   E15F             BEQ   PIO070
E150:A2 0A                    LDX   #$0A
E152:              *
E152:              *  BAD INTERRUPT -- SYSTEM DEATH
E152:              *
E152:A9 00         PIO050     LDA   #>BADINT1        ;INTERRUPT NOT FOUND
E154:20 00 00                 JSR   SYSDEATH
E157:A9 00         PIO060     LDA   #>BADINT2        ;BAD ZERO PAGE ALLOCATION
E159:20 00 00                 JSR   SYSDEATH
E15C:              *
E15C:              *  INTERRUPTING DEVICE FOUND
E15C:              *    ALLOCATE ZERO PAGE AND CALL MASTER INTERRUPT HANDLER
E15C:              *
E15C:              *  NOTE:
E15C:              *    SINCE READING THE ACIA'S STATUS REGISTER RESETS THE
E15C:              *    DSR AND DCD BITS, THE STATUS READ BY THE POLLING
E15C:              *    ROUTINE MUST BE PASSED TO THE INTERRUPT HANDLER;
E15C:              *    THE Y REGISTER HAS BEEN SELECTED FOR THIS PURPOSE.
E15C:              *    THE CURRENT IMPLEMENTATION DOES NOT USE Y IN CALLING
E15C:              *    THE INTERRUPT HANDLER.  IF SUBSEQUENT REVISIONS
E15C:              *    NEED TO USE Y, THE STATUS MUST BE PRESERVED AND
E15C:              *    RESTORED BEFORE CALLING THE INTERRUPT HANDLER.
E15C:              *
E15C:6C FD 00      CALLMIH    JMP   (IRQADDR)
E15F:              *
E15F:BD C5 DF      PIO070     LDA   SIRTABLE,X       ;INTERRUPT ALLOCATED?
E162:10 EE   E152             BPL   PIO050           ;  NO
E164:BD DD DF                 LDA   SIRADR.L,X       ;GET INTERRUPT ADDRESS
E167:85 FD                    STA   IRQADDR
E169:1D F6 DF                 ORA   SIRADR.H,X       ;CHECK FOR ADDRESS = $00
E16C:F0 E4   E152             BEQ   PIO050           ;  BAD ADDRESS
E16E:BD F6 DF                 LDA   SIRADR.H,X
E171:85 FE                    STA   IRQADDR+1
E173:BD 0E E0                 LDA   SIRADR.B,X
E176:8D EF FF                 STA   B.REG
E179:AD C4 DF                 LDA   ZPGSTACK         ;ALLOCATE MIH ZERO PAGE
E17C:C9 48                    CMP   #ZPGSTOP+ZPGSPACE
E17E:90 D7   E157             BCC   PIO060           ;TOO MANY NESTED INTERRUPTS
E180:E9 20                    SBC   #ZPGSPACE
E182:8D C4 DF                 STA   ZPGSTACK
E185:85 FF                    STA   ZPGSP
E187:AA                       TAX
E188:20 5C E1                 JSR   CALLMIH          ;CALL INTERRUPT HANDLER
E18B:78                       SEI
E18C:A9 00                    LDA   #$00
E18E:8D D0 FF                 STA   Z.REG
E191:18                       CLC
E192:AD C4 DF                 LDA   ZPGSTACK         ;DEALLOCATE MIH ZERO PAGE
E195:69 20                    ADC   #ZPGSPACE
E197:8D C4 DF                 STA   ZPGSTACK
E19A:85 FF                    STA   ZPGSP
E19C:A9 02                    LDA   #BITON1
E19E:8D DD FF                 STA   D.IFR            ;CLEAR ANY SLOT INTERRUPT
E1A1:4C 1D E2                 JMP   DISPATCH
```

```
E1A4:              ************************************************************
E1A4:              *
E1A4:              *  THIS IS THE NON-MASKABLE INTERRUPT RECEIVER.  WHEN AN
E1A4:              *  NMI OCCURS, THE CPU PASSES CONTROL TO THE NMI RECEIVER
E1A4:              *  THROUGH THE NMI VECTOR.  THE OPERATION OF THE NMI
E1A4:              *  RECEIVER IS ESSENTIALLY THE SAME AS THE GIR EXCEPT
E1A4:              *  THAT IT IS NOT CONCERNED WITH BRK, AND THE ONLY VALID
E1A4:              *  SOURCE OF AN NMI IS THE KEYBOARD OR THE I/O DEVICE THAT
E1A4:              *  HAS ALLOCATED THE NMI RESOURCE.
E1A4:              *
E1A4:              ************************************************************
E1A4:              *
E1A4:              *
E1A4:         E1A4 NMI.RCVR   EQU   *
E1A4:              *
E1A4:              *  SAVE CPU REGISTERS A, X, & Y ON CURRENT STACK
E1A4:              *
E1A4:48                       PHA
E1A5:8A                       TXA
E1A6:48                       PHA
E1A7:98                       TYA
E1A8:48                       PHA
E1A9:              *
E1A9:              *  CHECK FOR STACK OVERFLOW
E1A9:              *
E1A9:BA                       TSX
E1AA:E0 FA                    CPX   #$FA
E1AC:90 05   E1B3             BCC   NMI005
E1AE:A9 00                    LDA   #>STKOVFL
E1B0:20 00 00                 JSR   SYSDEATH
E1B3:              *
E1B3:              *  SET UP INTERRUPT ENVIRONMENT:
E1B3:              *    BINARY ARITHMETIC, 2 MHZ, I/O ENABLED,
E1B3:              *    RAM WRITE ENABLED, PRIMARY STACK,
E1B3:              *    AND $F000 RAM SELECTED.  PRESERVE
E1B3:              *    USER STATE OF SCREEN AND RESET LOCK.
E1B3:              *
E1B3:D8            NMI005     CLD
E1B4:AD DF FF                 LDA   E.REG
E1B7:AA                       TAX
E1B8:29 30                    AND   #BITON5+BITON4
E1BA:09 44                    ORA   #BITON6+BITON2
E1BC:8D DF FF                 STA   E.REG
E1BF:              *
E1BF:              *  IF NOT ALREADY ON PRIMARY STACK, SAVE USER'S
E1BF:              *  STACK POINTER AND SET UP SOS STACK POINTER.
E1BF:              *
E1BF:8A                       TXA
E1C0:29 04                    AND   #BITON2
E1C2:D0 09   E1CD             BNE   NMI010
E1C4:8A                       TXA
E1C5:BA                       TSX
E1C6:8E FF 01                 STX   SP.SAVE
E1C9:A2 FE                    LDX   #>E.SAVE
E1CB:9A                       TXS
E1CC:AA                       TAX
```

```
E1CD:              *
E1CD:              *  SAVE SYSTEM CONTROL REGISTERS E, Z, & B ON SOS STACK
E1CD:              *
E1CD:8A            NMI010     TXA
E1CE:48                       PHA
E1CF:AD D0 FF                 LDA   Z.REG
E1D2:48                       PHA
E1D3:AD EF FF                 LDA   B.REG
E1D6:48                       PHA
E1D7:AD C0 DF                 LDA   EXPNSLOT
E1DA:48                       PHA
E1DB:2C FF CF                 BIT   $CFFF
E1DE:2C 20 C0                 BIT   $C020            ;RESET I/O SPACE
E1E1:A9 00                    LDA   #$00
E1E3:8D C0 DF                 STA   EXPNSLOT
E1E6:              *
E1E6:              *  SET INTERRUPT ZERO PAGE
E1E6:              *
E1E6:A9 00                    LDA   #0
E1E8:8D D0 FF                 STA   Z.REG
E1EB:              *
E1EB:              *  SEE IF NMI IS FROM KEYBOARD OR I/O DEVICE
E1EB:              *
E1EB:AD E0 FF                 LDA   E.IORB
E1EE:30 20   E210             BMI   NMI030
E1F0:              *
E1F0:              *  NMI IS FROM I/O DEVICE
E1F0:              *
E1F0:AD C5 DF                 LDA   SIRTABLE         ;NMI ALLOCATED?
E1F3:10 16   E20B             BPL   NMI020
E1F5:20 FC E1                 JSR   CALLNMI
E1F8:78                       SEI
E1F9:4C 1D E2                 JMP   DISPATCH
E1FC:AD DD DF      CALLNMI    LDA   SIRADR.L
E1FF:8D F5 DF                 STA   NMIADR.L
E202:AD 0E E0                 LDA   SIRADR.B
E205:8D EF FF                 STA   B.REG
E208:6C F5 DF                 JMP   (NMIADR.L)
E20B:              *
E20B:              *  BAD INTERRUPT -- SYSTEM DEATH
E20B:              *
E20B:A9 00         NMI020     LDA   #>BADINT1        ;NMI NOT ALLOCATED
E20D:20 00 00                 JSR   SYSDEATH
E210:              *
E210:              *  NMI IS FROM THE KEYBOARD
E210:              *
E210:AD 00 00      NMI030     LDA   SYSBANK
E213:8D EF FF                 STA   B.REG
E216:20 00 00                 JSR   KYBDNMI
E219:78                       SEI
E21A:4C 1D E2                 JMP   DISPATCH
```

```
E21D:              ************************************************************
E21D:              *
E21D:              *  THIS IS THE DISPATCHER.  UPON COMPLETION, ALL SOS CALLS
E21D:              *  AND INTERRUPT HANDLERS RETURN CONTROL TO THE DISPATCHER.
E21D:              *  ITS PURPOSE IS TO SET UP THE APPROPRIATE ENVIRONMENT AND
E21D:              *  PASS CONTROL TO WHATEVER CODE SHOULD RUN NEXT.
E21D:              *
E21D:              *  WHEN SOS IS INTERRUPTED, CONTROL ALWAYS RETURNS TO THE
E21D:              *  INTERRUPTED CODE.  HOWEVER, WHEN THE USER IS INTERRUPTED,
E21D:              *  BY EITHER A SOS CALL OR AN INTERRUPT, THE DISPATCHER
E21D:              *  MUST CHECK THE EVENT QUEUE.  IF THERE IS AN ACTIVE EVENT
E21D:              *  WITH A PRIORITY HIGHER THAN THE CURRENT EVENT FENCE,
E21D:              *  CONTROL IS PASSED TO THE EVENT CODE.  OTHERWISE, CONTROL
E21D:              *  RETURNS TO THE INTERRUPTED CODE.
E21D:              *
E21D:              ************************************************************
E21D:              *
E21D:        E21D  DISPATCH   EQU   *
E21D:              *
E21D:              *  DISABLE INTERRUPTS AND RESTORE
E21D:              *  SYSTEM CONTROL REGISTERS B & Z
E21D:              *
E21D:78                       SEI
E21E:AD DF FF                 LDA   E.REG
E221:09 40                    ORA   #BITON6          ;ENABLE I/O
E223:8D DF FF                 STA   E.REG
E226:68                       PLA
E227:20 A9 E3                 JSR   SELC800          ;RESTORE I/O SPACE
E22A:68                       PLA
E22B:8D EF FF                 STA   B.REG
E22E:68                       PLA
E22F:8D D0 FF                 STA   Z.REG
E232:              *
E232:              *  CHECK SAVED ENVIRONMENT REGISTER
E232:              *  IF RETURNING TO PRIMARY STACK
E232:              *    THEN RESTORE E REG AND RELAUNCH SOS
E232:              *    ELSE RESET STACK POINTER & RESTORE E REG
E232:              *
E232:68                       PLA
E233:09 20                    ORA   #BITON5          ;SET SCREEN STATE TO
E235:2C 00 00                 BIT   SCRNMODE         ;  CURRENT SCREEN MODE
E238:30 02   E23C             BMI   DSP005
E23A:29 DF                    AND   #BITOFF5
E23C:A8            DSP005     TAY
E23D:29 04                    AND   #BITON2
E23F:F0 05   E246             BEQ   DSP010
E241:8C DF FF                 STY   E.REG
E244:D0 41   E287             BNE   DSP030
E246:68            DSP010     PLA
E247:AA                       TAX
E248:9A                       TXS
E249:8C DF FF                 STY   E.REG
E24C:              *
E24C:              *  CHECK FOR ACTIVE EVENT WITH PRIORITY > FENCE
E24C:              *
E24C:AD 00 00      DSP020     LDA   CEVPRI
```

```
E24F:AE 26 E0                 LDX   EVQ.LINK
E252:DD 27 E0                 CMP   EVQ.PRI,X
E255:B0 30   E287             BCS   DSP030
E257:              *
E257:              *  PROCESS ACTIVE EVENT TRAP
E257:              *  SAVE E, Z, B, & CALLER'S PRIORITY ON STACK THEN CALL
E257:              *  EVENT.  UPON RETURN, RESTORE PRIORITY, B, Z, & E THEN
E257:              *  CHECK FOR MORE EVENTS.
E257:              *
E257:AD DF FF                 LDA   E.REG
E25A:48                       PHA
E25B:AD D0 FF                 LDA   Z.REG
E25E:48                       PHA
E25F:AD EF FF                 LDA   B.REG
E262:48                       PHA
E263:AD 00 00                 LDA   CEVPRI
E266:48                       PHA
E267:20 8D E2                 JSR   DO.EVENT
E26A:78                       SEI
E26B:68                       PLA
E26C:8D 00 00                 STA   CEVPRI
E26F:68                       PLA
E270:8D EF FF                 STA   B.REG
E273:68                       PLA
E274:8D D0 FF                 STA   Z.REG
E277:68                       PLA
E278:09 20                    ORA   #BITON5          ;SET SCREEN STATE TO
E27A:2C 00 00                 BIT   SCRNMODE         ;  CURRENT SCREEN MODE
E27D:30 02   E281             BMI   DSP025
E27F:29 DF                    AND   #BITOFF5
E281:8D DF FF      DSP025     STA   E.REG
E284:4C 4C E2                 JMP   DSP020
E287:              *
E287:              *  RESTORE CPU REGISTERS Y, X, & A AND LAUNCH
E287:              *
E287:68            DSP030     PLA
E288:A8                       TAY
E289:68                       PLA
E28A:AA                       TAX
E28B:68                       PLA
E28C:40                       RTI
```

```
E28D:               ************************************************************
E28D:               *
E28D:               *  THIS SUBROUTINE CALLS THE HIGHEST PRIORITY ACTIVE EVENT.
E28D:               *  FIRST, IT DELINKS THE FIRST ENTRY ON THE ACTIVE LIST AND
E28D:               *  LINKS IT TO THE FREE LIST.  THEN, IT SETS UP THE BANK,
E28D:               *  ADDRESS, ID, & STATUS AND CALLS THE EVENT VIA AN RTI.
E28D:               *
E28D:               ************************************************************
E28D:               *
E28D:         E28D  DO.EVENT   EQU   *
E28D:               *
E28D:               *  WRITE ENABLE RAM
E28D:               *
E28D:AC DF FF                  LDY   E.REG
E290:98                        TYA
E291:29 F7                     AND   #BITOFF3
E293:8D DF FF                  STA   E.REG
E296:               *
E296:               *  DELINK ENTRY FROM ACTIVE LIST AND RELINK IT TO FREE LIST
E296:               *
E296:AE 26 E0                  LDX   EVQ.LINK
E299:BD 26 E0                  LDA   EVQ.LINK,X
E29C:8D 26 E0                  STA   EVQ.LINK
E29F:AD 28 E0                  LDA   EVQ.FREE
E2A2:9D 26 E0                  STA   EVQ.LINK,X
E2A5:8E 28 E0                  STX   EVQ.FREE
E2A8:               *
E2A8:               *  SET FENCE TO EVENT PRIORITY THEN RESTORE E REG
E2A8:               *
E2A8:BD 27 E0                  LDA   EVQ.PRI,X
E2AB:8D 00 00                  STA   CEVPRI
E2AE:8C DF FF                  STY   E.REG
E2B1:               *
E2B1:               *  SET UP B, Z, E, ADDRESS, ID, & STATUS
E2B1:               *
E2B1:BD 2B E0                  LDA   EVQ.BANK,X
E2B4:8D EF FF                  STA   B.REG
E2B7:BD 2A E0                  LDA   EVQ.ADRH,X
E2BA:48                        PHA
E2BB:BD 29 E0                  LDA   EVQ.ADRL,X
E2BE:48                        PHA
E2BF:BC 28 E0                  LDY   EVQ.ID,X
E2C2:08                        PHP
E2C3:68                        PLA
E2C4:29 82                     AND   #$82
E2C6:48                        PHA
E2C7:98                        TYA
E2C8:40                        RTI
E2C9:               *
E2C9:                          CHN   IPL.SRC2
```

```
E2C9:                  ************************************************************
E2C9:                  *
E2C9:                  *  SYSTEM INTERNAL RESOURCE NUMBERS
E2C9:                  *
E2C9:                  *
E2C9:                  *  SIR  RESOURCE
E2C9:                  *
E2C9:                  *   0   SOUND PORT / I/O NMI
E2C9:                  *   1   ACIA
E2C9:                  *   2   E.CA2 -- KEYBOARD
E2C9:                  *   3   E.CA1 -- CLOCK
E2C9:                  *   4   E.SR
E2C9:                  *   5   E.CB2 -- VBL +
E2C9:                  *   6   E.CB1 -- VBL -
E2C9:                  *   7   E.T2
E2C9:                  *   8   E.T1
E2C9:                  *   9   D.CA2 -- CSP INPUT FLAG / INPUT SWITCH 1
E2C9:                  *   A   D.CA1 -- ANY SLOT (RESERVED FOR SOS)
E2C9:                  *   B   D.SR  -- CSP DATA REGISTER
E2C9:                  *   C   D.CB2 -- CSP DATA I/O / ENSIO
E2C9:                  *   D   D.CB1 -- CSP CLOCK / ENSEL / A/D SELECT / INPUT SW3
E2C9:                  *   E   D.T2
E2C9:                  *   F   D.T1
E2C9:                  *  10   DISK STEPPER / GRAPHICS SCROLL / CHARACTER DOWNLOAD
E2C9:                  *  11   SLOT 1
E2C9:                  *  12   SLOT 2
E2C9:                  *  13   SLOT 3
E2C9:                  *  14   SLOT 4
E2C9:                  *  15   (UNASSIGNED)
E2C9:                  *  16   (UNASSIGNED)
E2C9:                  *  17   (UNASSIGNED)
E2C9:                  *
E2C9:                  ************************************************************
```

```
E2C9:                 ************************************************************
E2C9:                 *
E2C9:                 *  RESOURCE ALLOCATION AND DEALLOCATION
E2C9:                 *
E2C9:                 *  SIRS ARE ALLOCATED AND DEALLOCATED BY THE SUBROUTINES
E2C9:                 *  'ALLOCSIR' AND 'DEALCSIR'.  THE RESOURCE PARAMETERS ARE
E2C9:                 *  PASSED IN A TABLE THAT CONTAINS ONE FIVE-BYTE ENTRY FOR
E2C9:                 *  EACH SIR THAT IS TO BE ALLOCATED OR DEALLOCATED.  THE
E2C9:                 *  TABLE ENTRY FORMAT IS SHOWN BELOW:
E2C9:                 *
E2C9:                 *           0       1       2       3       4
E2C9:                 *       +-------+-------+-------+-------+-------+
E2C9:                 *       | SIR # |  ID   | ADR.L | ADR.H | ADR.B |
E2C9:                 *       +-------+-------+-------+-------+-------+
E2C9:                 *
E2C9:                 *  SIR # -- SYSTEM INTERNAL RESOURCE NUMBER
E2C9:                 *  ID    -- IDENTIFICATION BYTE
E2C9:                 *           SUPPLIED BY ALLOCSIR, CHECKED BY DEALCSIR
E2C9:                 *  ADR   -- INTERRUPT ADDRESS (LOW, HIGH, BANK)
E2C9:                 *           ZERO IF NO INTERRUPT HANDLER
E2C9:                 *
E2C9:                 *
E2C9:                 *  ALLOCSIR -- ALLOCATE SYSTEM INTERNAL RESOURCES
E2C9:                 *
E2C9:                 *    PARAMETERS:
E2C9:                 *      A:  NUMBER OF BYTES IN TABLE
E2C9:                 *      X:  TABLE ADDRESS (LOW BYTE)
E2C9:                 *      Y:  TABLE ADDRESS (HIGH BYTE)
E2C9:                 *
E2C9:                 *    NORMAL EXIT -- SIRS ALLOCATED
E2C9:                 *      CARRY:  CLEAR
E2C9:                 *      A, X, Y:  UNDEFINED
E2C9:                 *
E2C9:                 *    ERROR EXIT -- SIRS NOT ALLOCATED
E2C9:                 *      CARRY:  SET
E2C9:                 *      X:  SIR NUMBER
E2C9:                 *      A, Y:  UNDEFINED
E2C9:                 *
E2C9:                 *
E2C9:                 *  DEALCSIR -- DEALLOCATE SYSTEM INTERNAL RESOURCES
E2C9:                 *
E2C9:                 *    PARAMETERS:
E2C9:                 *      A:  NUMBER OF BYTES IN TABLE
E2C9:                 *      X:  TABLE ADDRESS (LOW BYTE)
E2C9:                 *      Y:  TABLE ADDRESS (HIGH BYTE)
E2C9:                 *
E2C9:                 *    NORMAL EXIT -- SIRS DEALLOCATED
E2C9:                 *      CARRY:  CLEAR
E2C9:                 *      A, X, Y:  UNDEFINED
E2C9:                 *
E2C9:                 *    ERROR EXIT -- SIRS NOT DEALLOCATED
E2C9:                 *      CARRY:  SET
E2C9:                 *      X:  SIR NUMBER
E2C9:                 *      A, Y:  UNDEFINED
E2C9:                 *
E2C9:                 ************************************************************
```

```
E2C9:              *
E2C9:81            IDBYTE     DFB   $81
E2CA:              *
E2CA:        E2CA  ALLOCSIR   EQU   *
E2CA:18                       CLC
E2CB:08                       PHP
E2CC:78                       SEI
E2CD:8D 00 00                 STA   SIRARGSIZ        ;SAVE TABLE SIZE
E2D0:AD DF FF                 LDA   E.REG
E2D3:8D 00 00                 STA   SIRTEMP
E2D6:09 04                    ORA   #BITON2          ;FORCE PRIMARY STACK
E2D8:29 F7                    AND   #BITOFF3         ;  AND WRITE ENABLE
E2DA:8D DF FF                 STA   E.REG
E2DD:AD 00 00                 LDA   SIRTEMP
E2E0:48                       PHA
E2E1:AD D0 FF                 LDA   Z.REG
E2E4:48                       PHA
E2E5:A9 00                    LDA   #$00
E2E7:8D D0 FF                 STA   Z.REG            ;SET ZERO PAGE := $00
E2EA:86 F9                    STX   SIRARGS
E2EC:84 FA                    STY   SIRARGS+1        ;SET POINTER TO TABLE
E2EE:              *
E2EE:A0 00                    LDY   #$00
E2F0:B1 F9         ASIR010    LDA   (SIRARGS),Y      ;GET SIR NUMBER
E2F2:C9 18                    CMP   #SIRTBLSIZ
E2F4:AA                       TAX
E2F5:B0 33   E32A             BCS   ASIR020
E2F7:BD C5 DF                 LDA   SIRTABLE,X       ;CHECK ALLOCATION
E2FA:30 2E   E32A             BMI   ASIR020
E2FC:AD C9 E2                 LDA   IDBYTE
E2FF:9D C5 DF                 STA   SIRTABLE,X       ;ALLOCATE SIR
E302:C8                       INY
E303:91 F9                    STA   (SIRARGS),Y      ;RETURN ID BYTE
E305:C8                       INY
E306:B1 F9                    LDA   (SIRARGS),Y
E308:9D DD DF                 STA   SIRADR.L,X       ;SAVE INTERRUPT ADDRESS
E30B:C8                       INY
E30C:B1 F9                    LDA   (SIRARGS),Y
E30E:9D F6 DF                 STA   SIRADR.H,X
E311:C8                       INY
E312:B1 F9                    LDA   (SIRARGS),Y
E314:9D 0E E0                 STA   SIRADR.B,X
E317:C8                       INY
E318:CC 00 00                 CPY   SIRARGSIZ
E31B:90 D3   E2F0             BCC   ASIR010
E31D:              *
E31D:18                       CLC
E31E:EE C9 E2                 INC   IDBYTE           ;BUMP ID BYTE
E321:30 1F   E342             BMI   SIREXIT
E323:A9 81                    LDA   #$81
E325:8D C9 E2                 STA   IDBYTE
E328:30 18   E342             BMI   SIREXIT
E32A:              *
E32A:8E 00 00      ASIR020    STX   SIRTEMP          ;SAVE BAD SIR NUMBER
E32D:38            ASIR030    SEC
E32E:98                       TYA
```

```
E32F:E9 05                    SBC   #5
E331:A8                       TAY
E332:90 0A   E33E             BCC   ASIR040
E334:B1 F9                    LDA   (SIRARGS),Y      ;GET SIR NUMBER
E336:AA                       TAX
E337:A9 00                    LDA   #FALSE
E339:9D C5 DF                 STA   SIRTABLE,X       ;RELEASE ALLOCATED SIRS
E33C:F0 EF   E32D             BEQ   ASIR030
E33E:              *
E33E:AE 00 00      ASIR040    LDX   SIRTEMP          ;RETURN BAD SIR
E341:38                       SEC
E342:              *
E342:              *
E342:              *
E342:68            SIREXIT    PLA
E343:8D D0 FF                 STA   Z.REG            ;RESTORE Z REGISTER
E346:68                       PLA
E347:8D DF FF                 STA   E.REG            ;RESTORE E REGISTER
E34A:90 04   E350             BCC   SIREXIT1
E34C:68                       PLA
E34D:09 01                    ORA   #BITON0
E34F:48                       PHA
E350:28            SIREXIT1   PLP
E351:60                       RTS
E352:              *
E352:              *
E352:              *
E352:        E352  DEALCSIR   EQU   *
E352:18                       CLC
E353:08                       PHP
E354:78                       SEI
E355:8D 00 00                 STA   SIRARGSIZ        ;SAVE TABLE SIZE
E358:AD DF FF                 LDA   E.REG
E35B:8D 00 00                 STA   SIRTEMP
E35E:09 04                    ORA   #BITON2          ;FORCE PRIMARY STACK
E360:29 F7                    AND   #BITOFF3         ;  AND WRITE ENABLE
E362:8D DF FF                 STA   E.REG
E365:AD 00 00                 LDA   SIRTEMP
E368:48                       PHA
E369:AD D0 FF                 LDA   Z.REG
E36C:48                       PHA
E36D:A9 00                    LDA   #$00
E36F:8D D0 FF                 STA   Z.REG            ;SET ZERO PAGE := $00
E372:86 F9                    STX   SIRARGS
E374:84 FA                    STY   SIRARGS+1        ;SET POINTER TO TABLE
E376:              *
E376:A0 00                    LDY   #$00
E378:B1 F9         DSIR010    LDA   (SIRARGS),Y      ;GET SIR NUMBER
E37A:AA                       TAX
E37B:E0 18                    CPX   #SIRTBLSIZ
E37D:B0 27   E3A6             BCS   DSIR030
E37F:C8                       INY
E380:BD C5 DF                 LDA   SIRTABLE,X
E383:10 21   E3A6             BPL   DSIR030          ;VERIFY ALLOCATION
E385:D1 F9                    CMP   (SIRARGS),Y
E387:D0 1D   E3A6             BNE   DSIR030
```

```
E389:C8                       INY
E38A:C8                       INY
E38B:C8                       INY
E38C:C8                       INY
E38D:CC 00 00                 CPY   SIRARGSIZ
E390:90 E6   E378             BCC   DSIR010
E392:               *
E392:AC 00 00                 LDY   SIRARGSIZ
E395:38             DSIR020   SEC
E396:98                       TYA
E397:E9 05                    SBC   #5
E399:A8                       TAY
E39A:90 A6   E342             BCC   SIREXIT
E39C:B1 F9                    LDA   (SIRARGS),Y      ;GET SIR NUMBER
E39E:AA                       TAX
E39F:A9 00                    LDA   #FALSE
E3A1:9D C5 DF                 STA   SIRTABLE,X
E3A4:F0 EF   E395             BEQ   DSIR020
E3A6:               *
E3A6:38             DSIR030   SEC
E3A7:B0 99   E342             BCS   SIREXIT
```

```
E3A9:              ************************************************************
E3A9:              *
E3A9:              *  SUBROUTINE 'SELC800' IS CALLED TO SELECT THE C800 I/O EX-
E3A9:              *  PANSION ADDRESS SPACE FOR A PERIPHERAL SLOT.  ON ENTRY,
E3A9:              *  THE SLOT NUMBER IS PASSED IN THE ACCUMULATOR.  IF NO
E3A9:              *  ERROR OCCURS, CARRY IS CLEARED; OTHERWISE, CARRY IS SET
E3A9:              *  AND THE PREVIOUS SLOT REMAINS SELECTED.
E3A9:              *
E3A9:              *  PARAMETERS:
E3A9:              *    A:  SLOT NUMBER
E3A9:              *
E3A9:              *  NORMAL EXIT -- NEW SLOT SELECTED
E3A9:              *    CARRY:  CLEAR
E3A9:              *    A:  UNDEFINED
E3A9:              *    X, Y:  UNCHANGED
E3A9:              *
E3A9:              *  ERROR EXIT -- SLOT NOT CHANGED
E3A9:              *    CARRY:  SET
E3A9:              *    A, X, Y:  UNCHANGED
E3A9:              *
E3A9:              *  WARNING !!!
E3A9:              *    'SELC800' USES SELF-MODIFYING CODE!
E3A9:              *
E3A9:              ************************************************************
E3A9:              *
E3A9:        E3A9  SELC800    EQU   *
E3A9:C9 05                    CMP   #$05             ;CHECK SLOT NUMBER
E3AB:B0 14   E3C1             BCS   SC8EXIT          ;  INVALID
E3AD:08                       PHP
E3AE:78                       SEI
E3AF:8D C0 DF                 STA   EXPNSLOT
E3B2:09 C0                    ORA   #$C0             ;MAKE SLOT INTO $CN00
E3B4:8D BF E3                 STA   CNADDR+2         ;  AND MODIFY BIT ADDRESS
E3B7:2C 20 C0                 BIT   $C020
E3BA:2C FF CF                 BIT   $CFFF            ;DESELECT PREVIOUS SLOT
E3BD:2C FF C0      CNADDR     BIT   $C0FF            ;  AND SELECT CURRENT SLOT
E3C0:28                       PLP
E3C1:60            SC8EXIT    RTS
```

```
E3C2:              266 ************************************************************
E3C2:              267 *
E3C2:              268 *  THE SUBROUTINES NMIDSBL AND NMIENBL ARE CALLED TO
E3C2:              269 *  DISABLE AND ENABLE NMI, RESPECTIVELY.  THERE ARE NO
E3C2:              270 *  INPUT PARAMETERS.  ON EXIT, THE REGISTERS ARE UN-
E3C2:              271 *  DEFINED.  NMIDSBL CLEARS THE CARRY FLAG IF NMI WAS
E3C2:              272 *  SUCCESSFULLY DISABLED; OTHERWISE, CARRY IS SET.
E3C2:              273 *
E3C2:              274 ************************************************************
E3C2:              275 *
E3C2:        E3C2  276 NMIDSBL    EQU   *
E3C2:AE DF FF      277            LDX   E.REG
E3C5:2C 00 00      278            BIT   NMIFLAG
E3C8:10 22   E3EC  279            BPL   NDS020
E3CA:8A            280            TXA
E3CB:09 80         281            ORA   #BITON7
E3CD:8D DF FF      282            STA   E.REG            ;SET 1MHZ
E3D0:A9 00         283            LDA   #$00
E3D2:8D 00 00      284            STA   NMICNTR
E3D5:8D 01 00      285            STA   NMICNTR+1
E3D8:2C 00 00      286 NDS010     BIT   NMIFLAG          ;NMI PENDING?
E3DB:10 0F   E3EC  287            BPL   NDS020           ;  NO
E3DD:EE 00 00      288            INC   NMICNTR          ;BUMP NMI COUNTER
E3E0:D0 F6   E3D8  289            BNE   NDS010           ;  AND RECHECK NMI FLAG
E3E2:EE 01 00      290            INC   NMICNTR+1
E3E5:D0 F1   E3D8  291            BNE   NDS010
E3E7:A9 00         292            LDA   #>NMIHANG        ;CAN'T LOCK NMI
E3E9:20 00 00      293            JSR   SYSDEATH
E3EC:8A            294 NDS020     TXA                    ;GET E.REG
E3ED:29 EF         295            AND   #BITOFF4         ;DISABLE NMI
E3EF:8D DF FF      296            STA   E.REG
E3F2:60            297            RTS
E3F3:              298 *
E3F3:              299 *
E3F3:              300 *
E3F3:        E3F3  301 NMIENBL    EQU   *
E3F3:AD DF FF      302            LDA   E.REG
E3F6:09 10         303            ORA   #BITON4          ;ENABLE NMI
E3F8:8D DF FF      304            STA   E.REG
E3FB:60            305            RTS
```

```
E3FC:                307 ***********************************************************
E3FC:                308 *
E3FC:                309 *  BY DEFAULT, KEYBOARD NMI IS IGNORED.  THE USER MAY
E3FC:                310 *  PROCESS NMI BY CHANGING THE ADDRESS IN SYSTEM GLOBAL.
E3FC:                311 *
E3FC:                312 ***********************************************************
E3FC:                313 *
E3FC:         E3FC   314 NMIDBUG    EQU   *
E3FC:BA              315            TSX                    ;SAVE THE STACK POINTER
E3FD:8E 00 00        316            STX   NMISPSV
E400:A9 03           317            LDA   #$03             ;SELECT MONITOR'S ZERO PAGE
E402:8D D0 FF        318            STA   Z.REG
E405:AD DF FF        319            LDA   E.REG
E408:09 03           320            ORA   #$03             ;SELECT MONITOR ROM
E40A:8D DF FF        321            STA   E.REG
E40D:20 01 F9        322            JSR   $F901            ;CALL THE MONITOR
E410:                323 *
E410:         E410   324 NMICONT    EQU   *
E410:AD DF FF        325            LDA   E.REG
E413:09 04           326            ORA   #BITON2          ;FORCE PRIMARY STACK
E415:8D DF FF        327            STA   E.REG
E418:AE 00 00        328            LDX   NMISPSV
E41B:9A              329            TXS                    ;RESTORE STACK POINTER
E41C:60              330            RTS
```

```
E41D:               ************************************************************
E41D:               *
E41D:               *  THE EVENT QUEUE IS USED TO HOLD THE PARAMETERS OF EVENTS
E41D:               *  THAT HAVE BEEN DETECTED BUT NOT YET RECOGNIZED.  EVENT
E41D:               *  QUEUE ENTRIES ARE ORGANIZED INTO TWO LINKED LISTS; A FREE
E41D:               *  LIST AND AN ACTIVE LIST.  EACH ENTRY IS SIX BYTES LONG,
E41D:               *  WITH THE FIRST BYTE (BYTE 0) USED AS A LINK.  THE LINK
E41D:               *  BYTE CONTAINS THE TABLE INDEX OF THE NEXT ENTRY IN THE
E41D:               *  LIST.  BECAUSE OF THE INDEXING METHOD, THE EVENT QUEUE
E41D:               *  MUST NOT EXCEED 256 BYTES.
E41D:               *
E41D:               *  ENTRY ZERO IS A SPECIAL ENTRY.  BYTE 0 IS THE INDEX OF
E41D:               *  THE FIRST ACTIVE ENTRY; BYTE 1 CONTAINS A ZERO, ALLOWING
E41D:               *  ENTRY 0 TO BE USED AS THE ACTIVE EVENT LIST TERMINATER;
E41D:               *  BYTE 2 CONTAINS THE INDEX OF THE FIRST FREE ENTRY; AND
E41D:               *  BYTES 4 THROUGH 6 ARE UNUSED.
E41D:               *
E41D:               *  THE FREE LIST IS LINKED LIFO.  THE ONLY VALID BYTE IN A
E41D:               *  FREE ENTRY IS THE LINK BYTE; THE REMAINING BYTES ARE
E41D:               *  UNDEFINED.  THE FREE LIST IS TERMINATED BY A LINK BYTE
E41D:               *  CONTAINING A ZERO.
E41D:               *
E41D:               *  THE ACTIVE LIST IS LINKED IN DECREASING PRIORITY ORDER
E41D:               *  WITH ENTRIES OF EQUAL PRIORITY LINKED FIFO.  BYTES 1
E41D:               *  THROUGH 5 CONTAIN THE EVENT PRIORITY, EVENT ID, LOW BYTE
E41D:               *  OF THE EVENT ADDRESS, HIGH BYTE OF THE EVENT ADDRESS, AND
E41D:               *  THE ADDRESS BANK.  THE ACTIVE LIST IS TERMINATED BY AN
E41D:               *  ENTRY WITH AN EVENT PRIORITY OF ZERO.
E41D:               *
E41D:               ************************************************************
```

```
E41D:              ***********************************************************
E41D:              *
E41D:              *  SUBROUTINE 'QUEEVENT' IS USED TO ENTER AN EVENT INTO THE
E41D:              *  EVENT QUEUE.  ACTIVE EVENTS ARE LINKED IN DECREASING
E41D:              *  PRIORITY ORDER WITH EVENTS OF EQUAL PRIORITY LINKED FIFO.
E41D:              *  EVENTS ARE REMOVED FROM THE QUEUE AS THEY ARE RECOGNIZED
E41D:              *  BY THE DISPATCHER.
E41D:              *
E41D:              *  PARAMETERS:
E41D:              *    X:  EVENT PARAMETER ADDRESS (LOW BYTE)
E41D:              *    Y:  EVENT PARAMETER ADDRESS (HIGH BYTE)
E41D:              *
E41D:              *    EVENT       0       1       2       3       4
E41D:              *    PARMS:  +-------+-------+-------+-------+-------+
E41D:              *            |  PRI  |   ID  | ADR.L | ADR.H | ADR.B |
E41D:              *            +-------+-------+-------+-------+-------+
E41D:              *            PRI:  EVENT PRIORITY
E41D:              *            ID:   EVENT ID BYTE
E41D:              *            ADR:  EVENT ADDRESS (LOW, HIGH, BANK)
E41D:              *
E41D:              *  EXIT CONDITIONS:
E41D:              *    CARRY:  CLEAR
E41D:              *    A, X, Y:  UNDEFINED
E41D:              *
E41D:              ***********************************************************
E41D:              *
E41D:        E41D  QUEEVENT   EQU   *
E41D:18                       CLC
E41E:08                       PHP
E41F:78                       SEI
E420:AD DF FF                 LDA   E.REG
E423:8D 00 00                 STA   QEVTEMP
E426:09 04                    ORA   #BITON2          ;FORCE PRIMARY STACK
E428:29 F7                    AND   #BITOFF3         ;  AND WRITE ENABLE
E42A:8D DF FF                 STA   E.REG
E42D:AD 00 00                 LDA   QEVTEMP
E430:48                       PHA
E431:AD D0 FF                 LDA   Z.REG
E434:48                       PHA
E435:A9 00                    LDA   #0
E437:8D D0 FF                 STA   Z.REG            ;SET ZERO PAGE := 0
E43A:              *
E43A:86 FB                    STX   QEVARGS
E43C:84 FC                    STY   QEVARGS+1        ;SET ARGUMENT POINTER
E43E:A0 00                    LDY   #0
E440:B1 FB                    LDA   (QEVARGS),Y      ;GET PRIORITY
E442:F0 38   E47C             BEQ   Q.EXIT           ;  IGNORE IF ZERO
E444:              *
E444:AE 28 E0                 LDX   EVQ.FREE
E447:F0 3D   E486             BEQ   Q.FULL
E449:8E 00 00                 STX   QEV.THIS         ;GET FIRST FREE ENTRY
E44C:BD 26 E0                 LDA   EVQ.LINK,X       ;  AND DELINK IT
E44F:8D 28 E0                 STA   EVQ.FREE
E452:              *
E452:A0 04                    LDY   #EVQ.SIZ-2
E454:B1 FB         QEV010     LDA   (QEVARGS),Y      ;COPY ARGUMENTS
```

```
E456:9D 2B E0                 STA   EVQ.BANK,X       ;  INTO NEW ENTRY
E459:CA                       DEX
E45A:88                       DEY
E45B:10 F7   E454             BPL   QEV010
E45D:              *
E45D:AE 00 00                 LDX   QEV.THIS
E460:A0 00                    LDY   #0
E462:8C 00 00      QEV020     STY   QEV.LAST
E465:B9 26 E0                 LDA   EVQ.LINK,Y
E468:A8                       TAY
E469:B9 27 E0                 LDA   EVQ.PRI,Y        ;SCAN EVENT QUEUE
E46C:DD 27 E0                 CMP   EVQ.PRI,X        ;  FOR PROPER POSITION
E46F:B0 F1   E462             BCS   QEV020
E471:              *
E471:98                       TYA
E472:9D 26 E0                 STA   EVQ.LINK,X       ;RELINK EVENT INTO QUEUE
E475:8A                       TXA
E476:AC 00 00                 LDY   QEV.LAST
E479:99 26 E0                 STA   EVQ.LINK,Y
E47C:              *
E47C:68            Q.EXIT     PLA
E47D:8D D0 FF                 STA   Z.REG            ;RESTORE Z REGISTER
E480:68                       PLA
E481:8D DF FF                 STA   E.REG            ;RESTORE E REGISTER
E484:28                       PLP
E485:60                       RTS
E486:              *
E486:A9 00         Q.FULL     LDA   #>EVQOVFL        ;EVENT QUEUE OVERFLOW
E488:20 00 00                 JSR   SYSDEATH

E48B:                         LST   ON
E48B:        E48B  ZZEND      EQU   *
E48B:        04CB  ZZLEN      EQU   ZZEND-ZZORG
E48B:        0000             IFNE  ZZLEN-LENIPL
 S                            FAIL  2,"SOSORG        FILE IS INCORRECT FOR IPL"
E48B:                         FIN
```

```
 0103 A.SAVE         C0F1 ACIASTAT      NE2CA ALLOCSIR       DFC1 ANYSLOT
 E2F0 ASIR010        E32A ASIR020        E32D ASIR030        E33E ASIR040
 FFEF B.REG          01FC B.SAVE           20 BACKBIT       X002D BACKMASK
X000F BADBRK        X0010 BADINT1       X0011 BADINT2          F7 BITOFF3
   EF BITOFF4          DF BITOFF5          BF BITOFF6          7F BITOFF7
   01 BITON0           02 BITON1           04 BITON2           10 BITON4
   20 BITON5           40 BITON6           80 BITON7         3200 BLABFM
?2E00 BLABFMI        6B52 BLABUFMG       6955 BLACFM         5E99 BLADISK3
 64D9 BLADMGR        68F4 BLAFMGR       ?2CF8 BLAGLOB       ?2AF8 BLAINIT
 55C0 BLAIPL         2000 BLALODR       ?6E6E BLAMEMMG       5466 BLAOMSG
 5466 BLAPATCH       665E BLASCMGR       6404 BLASERR        5A8B BLAUMGR
 E15C CALLMIH        E1FC CALLNMI       X0020 CEVPRI        X000D CHKBUF
 E3BD CNADDR         FFDE D.IER          FFDD D.IFR         NE352 DEALCSIR
NE21D DISPATCH       E28D DO.EVENT       E378 DSIR010        E395 DSIR020
 E3A6 DSIR030        E23C DSP005         E246 DSP010         E24C DSP020
 E281 DSP025         E287 DSP030         FFEE E.IER          FFED E.IFR
 FFEF E.IORA         FFE0 E.IORB         FFDF E.REG          01FE E.SAVE
NE026 EV.QUEUE       E02A EVQ.ADRH       E029 EVQ.ADRL       E02B EVQ.BANK
N0007 EVQ.CNT       NE028 EVQ.FREE       E028 EVQ.ID        N002A EVQ.LEN
NE026 EVQ.LINK       E027 EVQ.PRI       N0006 EVQ.SIZ       X0013 EVQOVFL
 DFC0 EXPNSLOT         00 FALSE          E05F GIR005         E07C GIR010
 E0A4 GIR020         E2C9 IDBYTE        NE050 IRQ.RCVR         FD IRQADDR
X0028 IRQCNTR       X0022 KYBDNMI       ?0400 LENBFMI        2266 LENBFM
 031C LENBUFMG       01FD LENCFM         056B LENDISK3       0185 LENDMGR
   61 LENFMGR       ?01B2 LENINIT        04CB LENIPL         0AF8 LENLODR
?0751 LENMEMMG       015A LENOMSG          00 LENPATCH       0296 LENSCMGR
   D5 LENSERR        040E LENUMGR        E3D8 NDS010         E3EC NDS020
NE1A4 NMI.RCVR       E1B3 NMI005         E1CD NMI010         E20B NMI020
 E210 NMI030         DFF5 NMIADR.L      X0029 NMICNTR       NE410 NMICONT
NE3FC NMIDBUG       NE3C2 NMIDSBL       NE3F3 NMIENBL       X0024 NMIFLAG
X0012 NMIHANG       X0023 NMISPSV        B800 ORGBFMI        BC00 ORGBFM
 F552 ORGBUFMG       F355 ORGCFM         E899 ORGDISK3       EED9 ORGDMGR
 FFBF ORGEND         F2F4 ORGFMGR       ?18FC ORGGLOB        28F8 ORGINIT
 DFC0 ORGIPL         1E00 ORGLODR        F86E ORGMEMMG       DE66 ORGOMSG
 DE66 ORGPATCH       F05E ORGSCMGR       EE04 ORGSERR        E48B ORGUMGR
 E0E7 PIO004         E0EA PIO006         E10F PIO010         E118 PIO020
 E129 PIO030         E130 PIO035         E134 PIO040         E152 PIO050
 E157 PIO060         E15F PIO070         E0DA POLL.IO        E47C Q.EXIT
 E486 Q.FULL        X002C QEV.LAST      X002B QEV.THIS       E454 QEV010
 E462 QEV020           FB QEVARGS       X002A QEVTEMP       NE41D QUEEVENT
 0104 S.SAVE         E3C1 SC8EXIT       X000C SCMGR         X0025 SCRNMODE
NE3A9 SELC800       X001F SERR           E00E SIRADR.B       DFF6 SIRADR.H
 DFDD SIRADR.L         F9 SIRARGS       X0027 SIRARGSIZ      E350 SIREXIT1
 E342 SIREXIT       NDFC5 SIRTABLE      N0018 SIRTBLSIZ     X0026 SIRTEMP
 C065 SLOT1          C064 SLOT2          DFC2 SLOT3          DFC3 SLOT4
 01FF SP.SAVE       X0014 STKOVFL       X0021 SYSBANK       X000E SYSDEATH
 FFD0 Z.REG          01FD Z.SAVE           FF ZPGSP            20 ZPGSPACE
NDFC4 ZPGSTACK      N00F8 ZPGSTART         28 ZPGSTOP        E48B ZZEND
 04CB ZZLEN          DFC0 ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED  1165
** FREE SPACE PAGE COUNT   79
```

```
SOURCE   FILE #01 =>UMGR.SRC
 INCLUDE FILE #02 =>SOSORG
```

```
0000:                             REL
0000:                             INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955   BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52   BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E   BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
E48B:        E48B              ORG   ORGUMGR
E48B:        E48B   ZZORG      EQU   *
E48B:                          MSB   OFF
E48B:               ************************************************************
E48B:               *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
E48B:               *                    ALL RIGHTS RESERVED
E48B:               ************************************************************
E48B:               *  UTILITY MANAGER
E48B:               *
E48B:               *  THIS MODULE HANDLES THE FOLLOWING SOS CALLS:
E48B:               *    SET.FENCE,   GET.FENCE
E48B:               *    SET.TIME,    GET.TIME
E48B:               *    JOYSTICK,    COLDSTRT
E48B:               *
E48B:               *  IN ADDITION, IT CONTAINS THE ROUITNE DATETIME WHICH
E48B:               *  PROVIDES THE DATE AND TIME FOR THE BLOCK FILE MANAGER.
E48B:               *
E48B:               ************************************************************
E48B:               *
E48B:        E48B              ENTRY UMGR
E48B:        E656              ENTRY DATETIME
E48B:        E706              ENTRY BCDBIN
E48B:        E833              ENTRY COLDSTRT
E48B:               *
E48B:        E4C3              ENTRY PCLOCK
E48B:               *
E48B:        0000              EXTRN SYSBANK
E48B:        0000              EXTRN CEVPRI
E48B:        0000              EXTRN SYSERR
E48B:        0000              EXTRN BADSCNUM
E48B:        0000              EXTRN BADJMODE
E48B:        0000              EXTRN XNORESRC
E48B:        0000              EXTRN ALLOCSIR
E48B:        0000              EXTRN DEALCSIR
E48B:               *
E48B:        00C0   U.TPARMX   EQU   $C0
E48B:        00C0   U.REQCODE  EQU   U.TPARMX
E48B:        00C1   PRIORITY   EQU   U.TPARMX+1
E48B:        00C1   J.MODE     EQU   U.TPARMX+1
E48B:        00C2   J.VALUE    EQU   U.TPARMX+2
E48B:        00C1   TIME       EQU   U.TPARMX+1
E48B:        00C1   MEMORY     EQU   U.TPARMX+1
E48B:               *
E48B:        0004   BITON2     EQU   $04
E48B:        0020   BITON5     EQU   $20
E48B:        0040   BITON6     EQU   $40
E48B:        0080   BITON7     EQU   $80
E48B:        00DF   BITOFF5    EQU   $DF
E48B:               *
E48B:        FFD0   Z.REG      EQU   $FFD0
E48B:        FFDF   E.REG      EQU   $FFDF
E48B:        FFEF   B.REG      EQU   $FFEF
```

```
E48B:                   ***********************************
E48B:                   *
E48B:                   * UTILITY SWITCH
E48B:                   *
E48B:                   ***********************************
E48B:                   *
E48B:                   *
E48B:         E48B      UMGR       EQU   *
E48B:AD DF FF                      LDA   E.REG            ;SELECT $C000 I/O SPACE
E48E:09 40                         ORA   #BITON6
E490:8D DF FF                      STA   E.REG
E493:                   *
E493:A5 C0                         LDA   U.REQCODE
E495:C9 06                         CMP   #USWCNT
E497:B0 0B    E4A4                 BCS   UMGRERR
E499:0A                            ASL   A
E49A:AA                            TAX
E49B:BD AA E4                      LDA   USWTBL+1,X
E49E:48                            PHA
E49F:BD A9 E4                      LDA   USWTBL,X
E4A2:48                            PHA
E4A3:60                            RTS
E4A4:                   *
E4A4:A9 00              UMGRERR    LDA   #>BADSCNUM
E4A6:20 00 00                      JSR   SYSERR
E4A9:                   *
E4A9:                   *  UTILITY SWITCH TABLE
E4A9:                   *
E4A9:         E4A9      USWTBL     EQU   *
E4A9:B4 E4                         DW    SET.FENCE-1
E4AB:BA E4                         DW    GET.FENCE-1
E4AD:D9 E4                         DW    SET.TIME-1
E4AF:C0 E5                         DW    GET.TIME-1
E4B1:1D E7                         DW    JOYSTICK-1
E4B3:32 E8                         DW    COLDSTRT-1
E4B5:         0006      USWCNT     EQU   *-USWTBL/2
```

```
E4B5:              ************************************************************
E4B5:              *
E4B5:              * SET.FENCE(IN.PRIORITY) SYSTEM CALL
E4B5:              *
E4B5:              * GET.FENCE(OUT.PRIORITY) SYSTEM CALL
E4B5:              *
E4B5:              * THESE TWO CALLS ALLOW THE CALLER TO EITHER RETRIEVE OR SET
E4B5:              * THE CURRENT SYSTEM EVENT PRIORITY THRESHOLD.  BY RAISING
E4B5:              * THE FENCE, A USER MAY INHIBIT THE EXECUTION OF EVENTS WHOSE
E4B5:              * PRIORITY IS EQUAL TO OR LESS THAN THE VALUE OF THE SYSTEM
E4B5:              * FENCE.
E4B5:              *
E4B5:              ************************************************************
E4B5:              *
E4B5:              *
E4B5:        E4B5  SET.FENCE  EQU   *
E4B5:A5 C1                    LDA   PRIORITY
E4B7:8D 00 00                 STA   CEVPRI
E4BA:60                       RTS                    ; NORMAL EXIT
E4BB:              *
E4BB:              *
E4BB:        E4BB  GET.FENCE  EQU   *
E4BB:AD 00 00                 LDA   CEVPRI
E4BE:A0 00                    LDY   #0
E4C0:91 C1                    STA   (PRIORITY),Y
E4C2:60                       RTS                    ; NORMAL EXIT
```

```
E4C3:              ************************************************************
E4C3:              *
E4C3:              *  SET.TIME(IN.TIME)
E4C3:              *  GET.TIME(OUT.TIME)
E4C3:              *
E4C3:              *  THESE SYSTEM CALLS ALLOW THE USER TO SET AND READ THE
E4C3:              *  SYSTEM'S CLOCK.  THE TIME IS EXPRESSED AS AN EIGHTEEN
E4C3:              *  DIGIT ASCII STRING IN THE FORM "YYYYMMDDWHHMMSSMMM".
E4C3:              *
E4C3:              *    YYYY  YEAR       [1900-1999]
E4C3:              *      MM  MONTH        [01-12]
E4C3:              *      DD  DAY          [01-31]
E4C3:              *       W  WEEKDAY       [1-7]  1 => SUNDAY
E4C3:              *      HH  HOUR         [00-23]
E4C3:              *      MM  MINUTE       [00-59]
E4C3:              *      SS  SECOND       [00-59]
E4C3:              *     MMM  MILLISECOND [000-999]
E4C3:              *
E4C3:              *  THE CLOCK CHIP AUTOMATICALLY MAINTAINS THE TIME AND
E4C3:              *  DATE FROM MILLISECONDS TO MONTHS.  IT DOES NOT MAINTAIN
E4C3:              *  THE YEAR, HOWEVER, NOR DOES IT RECOGNIZE 29 FEBRUARY
E4C3:              *  IN LEAP YEARS.  THE SOFTWARE SETS THE DAY AND MONTH
E4C3:              *  LATCHES TO THE DON'T CARE STATE AND USES THE REMAINING
E4C3:              *  EIGHT BITS TO HOLD A TWO DIGIT BCD YEAR.  THE CLOCK
E4C3:              *  MUST BE RESET AT THE BEGINNING OF EACH YEAR AND ON
E4C3:              *  29 FEBRUARY IN LEAP YEARS.
E4C3:              *
E4C3:              *  SET.TIME ASSUMES THAT THE DATE IS VALID AND CORRECT.
E4C3:              *  THE CENTURY IS IGNORED AND MILLISECONDS ARE ALWAYS SET
E4C3:              *  TO ZERO.  GET.TIME ALWAYS SETS THE CENTURY TO 19.
E4C3:              *
E4C3:              ************************************************************
E4C3:              *
E4C3:              *
E4C3:              *  TEMPORARY ZERO PAGE
E4C3:              *
E4C3:        00D0  PCLK       EQU   $D0              ;POINTER TO SAVED PCLOCK
E4C3:        00D2  WKDAY      EQU   $D2
E4C3:        00D3  CKSUM      EQU   $D3
E4C3:        18D4  CLKTEMP    EQU   $18D4            ;THROUGH $18DD - ABSOLUTE
E4C3:              *
E4C3:              *  CLOCK LOCAL DATA
E4C3:              *
E4C3:        000A  PCLOCK     DS    $0A              ;PSEUDO CLOCK REGISTERS
E4CD:        0001  RETRY      DS    $01
E4CE:              *
E4CE:              *  CLOCK HARDWARE ADDRESSES
E4CE:              *
E4CE:        C070  CLOCK      EQU   $C070
E4CE:        0002  CSEC       EQU   $02
E4CE:        0003  CMIN       EQU   $03
E4CE:        0007  CMON       EQU   $07
E4CE:        000E  LDAY       EQU   $0E
E4CE:        0012  CRESET     EQU   $12
E4CE:        0014  STATUS     EQU   $14
E4CE:              *
```

```
E4CE:08 0B 0B 07   WKMON      DFB   8,11,11,7,9,12
E4D4:07 0A 0D 08              DFB   7,10,13,8,11,13
E4DA:              *
E4DA:              *
E4DA:        E4DA  SET.TIME   EQU   *
E4DA:A2 00                    LDX   #$00
E4DC:A0 12                    LDY   #$12
E4DE:A9 30                    LDA   #'0'
E4E0:D0 03   E4E5             BNE   STIM011
E4E2:              *
E4E2:E8            STIM010    INX
E4E3:B1 C1                    LDA   (TIME),Y         ;CONVERT TIME FROM
E4E5:29 0F         STIM011    AND   #$0F             ;  ASCII TO BCD AND
E4E7:9D C3 E4                 STA   PCLOCK,X         ;  TRANSFER TO PCLOCK
E4EA:88                       DEY
E4EB:C0 07                    CPY   #$07
E4ED:F0 F3   E4E2             BEQ   STIM010
E4EF:B1 C1                    LDA   (TIME),Y
E4F1:0A                       ASL   A
E4F2:0A                       ASL   A
E4F3:0A                       ASL   A
E4F4:0A                       ASL   A
E4F5:1D C3 E4                 ORA   PCLOCK,X
E4F8:9D C3 E4                 STA   PCLOCK,X
E4FB:88                       DEY
E4FC:10 E4   E4E2             BPL   STIM010
E4FE:              *
E4FE:AD CA E4                 LDA   PCLOCK+7         ;CALCULATE WEEKDAY
E501:20 06 E7                 JSR   BCDBIN
E504:AA                       TAX
E505:AD CB E4                 LDA   PCLOCK+8
E508:20 06 E7                 JSR   BCDBIN
E50B:A8                       TAY
E50C:4A                       LSR   A
E50D:4A                       LSR   A
E50E:85 D2                    STA   WKDAY
E510:98                       TYA
E511:29 03                    AND   #$03
E513:D0 05   E51A             BNE   STIM015
E515:E0 03                    CPX   #3
E517:B0 01   E51A             BCS   STIM015          ; <SRS 82.162>
E519:88                       DEY
E51A:18            STIM015    CLC
E51B:98                       TYA
E51C:65 D2                    ADC   WKDAY
E51E:7D CD E4                 ADC   WKMON-1,X
E521:85 D2                    STA   WKDAY
E523:AD C9 E4                 LDA   PCLOCK+6
E526:20 06 E7                 JSR   BCDBIN
E529:18                       CLC
E52A:65 D2                    ADC   WKDAY
E52C:38                       SEC
E52D:E9 07         STIM016    SBC   #7
E52F:C9 08                    CMP   #8
E531:B0 FA   E52D             BCS   STIM016
E533:8D C8 E4                 STA   PCLOCK+5
```

```
E536:              *
E536:A9 D0                   LDA   #$D0             ;POINT (PCLK) TO 8F:FFD0
E538:85 D0                   STA   PCLK
E53A:A9 FF                   LDA   #$FF
E53C:85 D1                   STA   PCLK+1
E53E:A9 8F                   LDA   #$8F
E540:8D D1 14                STA   $1401+PCLK
E543:A9 A5                   LDA   #$A5
E545:85 D3                   STA   CKSUM            ;INITIALIZE CHECKSUM
E547:A0 00                   LDY   #$00
E549:              *
E549:B9 C3 E4      STIM020   LDA   PCLOCK,Y         ;SAVE PCLOCK
E54C:91 D0                   STA   (PCLK),Y         ;  BEHIND 6522
E54E:45 D3                   EOR   CKSUM
E550:85 D3                   STA   CKSUM
E552:C8                      INY
E553:C0 0A                   CPY   #$0A
E555:90 F2   E549            BCC   STIM020
E557:91 D0                   STA   (PCLK),Y         ;SAVE CHECKSUM
E559:              *
E559:AD D0 FF                LDA   Z.REG
E55C:48                      PHA                    ;SAVE ZERO PAGE
E55D:AD DF FF                LDA   E.REG
E560:48                      PHA                    ;SAVE ENVIRONMENT
E561:09 80                   ORA   #BITON7          ;  AND SET 1 MHZ
E563:8D DF FF                STA   E.REG
E566:              *
E566:A0 14                   LDY   #STATUS
E568:8C D0 FF                STY   Z.REG
E56B:AD 70 C0                LDA   CLOCK            ;DOES CLOCK EXIST?
E56E:30 48   E5B8            BMI   STIM050          ;  NO
E570:              *
E570:A2 12                   LDX   #CRESET
E572:8E D0 FF                STX   Z.REG
E575:A9 FF                   LDA   #$FF             ;RESET ALL COUNTERS
E577:8D 70 C0                STA   CLOCK
E57A:8D 70 C0                STA   CLOCK
E57D:              *
E57D:A2 01                   LDX   #CSEC-1
E57F:E8            STIM030   INX
E580:08                      PHP
E581:78                      SEI                    ;DISABLE INTERRUPTS
E582:8E D0 FF      STIM040   STX   Z.REG
E585:AD 70 C0                LDA   CLOCK            ;(DUMMY READ FOR STATUS)
E588:BD C3 E4                LDA   PCLOCK,X
E58B:8D 70 C0                STA   CLOCK            ;SET CLOCK COUNTER
E58E:AD 70 C0                LDA   CLOCK            ;(DUMMY READ FOR STATUS)
E591:8C D0 FF                STY   Z.REG
E594:AD 70 C0                LDA   CLOCK            ;CHECK STATUS BIT
E597:D0 E9   E582            BNE   STIM040
E599:28                      PLP                    ;RESTORE INTERRUPTS
E59A:E0 07                   CPX   #CMON
E59C:90 E1   E57F            BCC   STIM030
E59E:              *
E59E:A2 0E                   LDX   #LDAY
E5A0:8E D0 FF                STX   Z.REG
```

```
E5A3:AD CB E4                  LDA   PCLOCK+8
E5A6:09 CC                     ORA   #$CC             ;STUFF YEAR INTO DAY
E5A8:8D 70 C0                  STA   CLOCK            ;  AND MONTH LATCHES
E5AB:EE D0 FF                  INC   Z.REG
E5AE:AD CB E4                  LDA   PCLOCK+8
E5B1:4A                        LSR   A
E5B2:4A                        LSR   A
E5B3:09 CC                     ORA   #$CC
E5B5:8D 70 C0                  STA   CLOCK
E5B8:                  *
E5B8:68                STIM050 PLA
E5B9:8D DF FF                  STA   E.REG            ;RESTORE ENVIRONMENT
E5BC:68                        PLA
E5BD:8D D0 FF                  STA   Z.REG            ;  AND ZERO PAGE
E5C0:60                        RTS
```

```
E5C1:        E5C1  GET.TIME   EQU   *
E5C1:AD D0 FF                 LDA   Z.REG            ;SAVE ZERO PAGE
E5C4:48                       PHA
E5C5:AD DF FF                 LDA   E.REG            ;SAVE ENVIRONMENT
E5C8:48                       PHA
E5C9:09 80                    ORA   #BITON7
E5CB:8D DF FF                 STA   E.REG            ;SET 1 MHZ
E5CE:              *
E5CE:A0 14                    LDY   #STATUS
E5D0:8C D0 FF                 STY   Z.REG
E5D3:AD 70 C0                 LDA   CLOCK            ;DOES CLOCK EXIST?
E5D6:30 45   E61D             BMI   GTIM050          ;  NO
E5D8:              *
E5D8:A9 10                    LDA   #$10             ;ALLOW $10 RETRYS
E5DA:8D CD E4                 STA   RETRY
E5DD:A2 08         GTIM010    LDX   #CMON+1
E5DF:08                       PHP
E5E0:78                       SEI                    ;DISABLE INTERRUPTS
E5E1:              *
E5E1:CA            GTIM020    DEX
E5E2:30 19   E5FD             BMI   GTIM030          ;ALL DONE
E5E4:8E D0 FF                 STX   Z.REG
E5E7:AD 70 C0                 LDA   CLOCK            ;COPY CLOCK COUNTERS
E5EA:9D D4 18                 STA   CLKTEMP,X        ;  TO TEMP REGISTERS
E5ED:8C D0 FF                 STY   Z.REG
E5F0:AD 70 C0                 LDA   CLOCK            ;CHECK STATUS BIT
E5F3:F0 EC   E5E1             BEQ   GTIM020
E5F5:              *
E5F5:28                       PLP                    ;CLOCK READ ERROR
E5F6:CE CD E4                 DEC   RETRY
E5F9:10 E2   E5DD             BPL   GTIM010          ;TRY AGAIN
E5FB:30 20   E61D             BMI   GTIM050
E5FD:              *
E5FD:28            GTIM030    PLP                    ;RESTORE INTERRUPTS
E5FE:A2 0F                    LDX   #LDAY+1
E600:8E D0 FF                 STX   Z.REG
E603:AD 70 C0                 LDA   CLOCK            ;READ YEAR FROM DAY
E606:38                       SEC                    ;  AND MONTH LATCHES
E607:2A                       ROL   A
E608:2A                       ROL   A
E609:CE D0 FF                 DEC   Z.REG
E60C:2D 70 C0                 AND   CLOCK
E60F:8D DC 18                 STA   CLKTEMP+8
E612:              *
E612:A2 09                    LDX   #$09
E614:BD D4 18      GTIM040    LDA   CLKTEMP,X        ;COPY CLOCK DATA
E617:9D C3 E4                 STA   PCLOCK,X         ;  TO PSEUDO CLOCK
E61A:CA                       DEX
E61B:10 F7   E614             BPL   GTIM040
E61D:              *
E61D:A9 19         GTIM050    LDA   #$19
E61F:8D CC E4                 STA   PCLOCK+9
E622:              *
E622:68                       PLA
E623:8D DF FF                 STA   E.REG            ;RESTORE ENVIRONMENT
E626:68                       PLA
```

```
E627:8D D0 FF                STA   Z.REG            ;  AND ZERO PAGE
E62A:              *
E62A:A0 11                   LDY   #$11
E62C:A2 00                   LDX   #$00
E62E:BD C3 E4      GTIM060   LDA   PCLOCK,X         ;GET MOST SIGNIFICANT
E631:4A                      LSR   A                ;  BCD DIGIT
E632:4A                      LSR   A
E633:4A                      LSR   A
E634:4A                      LSR   A
E635:09 30                   ORA   #$30             ;CONVERT TO ASCII
E637:91 C1                   STA   (TIME),Y
E639:E8                      INX
E63A:88                      DEY
E63B:30 11   E64E            BMI   GTIM080
E63D:BD C3 E4      GTIM070   LDA   PCLOCK,X         ;GET LEAST SIGNIFICANT
E640:29 0F                   AND   #$0F             ;  BCD DIGIT
E642:09 30                   ORA   #$30             ;CONVERT TO ASCII
E644:91 C1                   STA   (TIME),Y
E646:88                      DEY
E647:C0 07                   CPY   #$07
E649:D0 E3   E62E            BNE   GTIM060
E64B:E8                      INX
E64C:D0 EF   E63D            BNE   GTIM070
E64E:60            GTIM080   RTS
```

```
E64F:              ************************************************************
E64F:              *
E64F:              *  SUBROUTINE DATETIME
E64F:              *
E64F:              *  THIS SUBROUTINE READS THE CLOCK AND WRITES A DATE/TIME
E64F:              *  STAMP TO A FOUR BYTE BUFFER ON THE CALLER'S ZERO PAGE;
E64F:              *  THE DATA FORMAT IS SHOWN BELOW.  ON ENTRY, X MUST POINT
E64F:              *  TO THE BUFFER.  ON EXIT, ALL REGISTERS ARE CLOBBERED.
E64F:              *  IF AN ERROR OCCURS, CARRY IS SET AND THE BUFFER IS
E64F:              *  SET TO ZERO; OTHERWISE, CARRY IS CLEARED.
E64F:              *
E64F:              *   BITS: 7 6 5 4 3 2 1 0
E64F:              *    X+0  M M M D D D D D
E64F:              *    X+1  Y Y Y Y Y Y Y M
E64F:              *    X+2   -  MINUTE   -
E64F:              *    X+3   - - HOUR  - -
E64F:              *
E64F:              ************************************************************
E64F:              *
E64F:              *  TEMPORARY STORAGE
E64F:              *
E64F:00            OFFSET     DFB   0
E650:00            ERRCNT     DFB   0
E651:        0005  CLKREGS    DS    5
E656:        E651  MIN        EQU   CLKREGS+0
E656:        E652  HOUR       EQU   CLKREGS+1
E656:        E654  DAY        EQU   CLKREGS+3
E656:        E655  MON        EQU   CLKREGS+4
E656:        E653  YEAR       EQU   CLKREGS+2
E656:              *
E656:              *
E656:        E656  DATETIME   EQU   *
E656:8E 4F E6                 STX   OFFSET
E659:AD D0 FF                 LDA   Z.REG
E65C:48                       PHA               ;SAVE ZERO PAGE
E65D:AD DF FF                 LDA   E.REG
E660:48                       PHA               ;  AND ENVIRONMENT
E661:09 C0                    ORA   #BITON7+BITON6   ;SET 1 MHZ AND
E663:8D DF FF                 STA   E.REG       ;  ENABLE I/O SPACE
E666:              *
E666:A0 14                    LDY   #STATUS
E668:8C D0 FF                 STY   Z.REG
E66B:AD 70 C0                 LDA   CLOCK       ;DOES CLOCK EXIST?
E66E:30 25   E695             BMI   DT030       ;  NO
E670:              *
E670:A9 08                    LDA   #8
E672:8D 50 E6                 STA   ERRCNT      ;ALLOW 8 RETRYS
E675:A2 08         DT010      LDX   #CMON+1
E677:08                       PHP
E678:78                       SEI               ;DISABLE INTERRUPTS
E679:              *
E679:CA            DT020      DEX
E67A:E0 03                    CPX   #CMIN
E67C:90 30   E6AE             BCC   DT050
E67E:8E D0 FF                 STX   Z.REG
E681:AD 70 C0                 LDA   CLOCK       ;READ THE CLOCK
```

```
E684:9D 4E E6                 STA   CLKREGS-CMIN,X
E687:8C D0 FF                 STY   Z.REG
E68A:AD 70 C0                 LDA   CLOCK            ;CHECK STATUS
E68D:F0 EA   E679             BEQ   DT020
E68F:              *
E68F:28                       PLP                    ;CLOCK READ ERROR
E690:CE 50 E6                 DEC   ERRCNT
E693:10 E0   E675             BPL   DT010
E695:68            DT030      PLA
E696:8D DF FF                 STA   E.REG            ;RESTORE ENVIRONMENT
E699:68                       PLA
E69A:8D D0 FF                 STA   Z.REG            ;  AND ZERO PAGE
E69D:A2 04                    LDX   #CMON-CMIN
E69F:BD C6 E4      DT040      LDA   PCLOCK+CMIN,X
E6A2:9D 51 E6                 STA   CLKREGS,X
E6A5:CA                       DEX
E6A6:10 F7   E69F             BPL   DT040
E6A8:AE CB E4                 LDX   PCLOCK+8
E6AB:4C C9 E6                 JMP   DT060
E6AE:              *
E6AE:28            DT050      PLP                    ;READ YEAR FROM LATCHES
E6AF:A9 0F                    LDA   #LDAY+1
E6B1:8D D0 FF                 STA   Z.REG
E6B4:AD 70 C0                 LDA   CLOCK
E6B7:38                       SEC
E6B8:2A                       ROL   A
E6B9:2A                       ROL   A
E6BA:CE D0 FF                 DEC   Z.REG
E6BD:2D 70 C0                 AND   CLOCK
E6C0:AA                       TAX
E6C1:              *
E6C1:68                       PLA
E6C2:8D DF FF                 STA   E.REG            ;RESTORE ENVIRONMENT
E6C5:68                       PLA
E6C6:8D D0 FF                 STA   Z.REG            ;  AND ZERO PAGE
E6C9:              *
E6C9:8A            DT060      TXA
E6CA:20 06 E7                 JSR   BCDBIN           ;CONVERT YEAR TO BINARY
E6CD:8D 53 E6                 STA   YEAR
E6D0:AD 55 E6                 LDA   MON              ;CONVERT MONTH AND DAY
E6D3:20 06 E7                 JSR   BCDBIN            ;  TO BINARY THEN
E6D6:0A                       ASL   A                 ;  COMBINE WITH YEAR
E6D7:0A                       ASL   A                 ;  TO FORM DATE STAMP
E6D8:0A                       ASL   A
E6D9:0A                       ASL   A
E6DA:0A                       ASL   A
E6DB:8D 55 E6                 STA   MON
E6DE:2E 53 E6                 ROL   YEAR
E6E1:AD 54 E6                 LDA   DAY
E6E4:20 06 E7                 JSR   BCDBIN
E6E7:0D 55 E6                 ORA   MON
E6EA:AE 4F E6                 LDX   OFFSET
E6ED:95 00                    STA   0,X
E6EF:AD 53 E6                 LDA   YEAR
E6F2:95 01                    STA   1,X
E6F4:AD 51 E6                 LDA   MIN              ;CONVERT MINUTE
```

```
E6F7:20 06 E7                     JSR   BCDBIN
E6FA:95 02                        STA   2,X
E6FC:AD 52 E6                     LDA   HOUR             ;CONVERT HOUR
E6FF:20 06 E7                     JSR   BCDBIN
E702:95 03                        STA   3,X
E704:18                           CLC
E705:60                           RTS
```

```
E706:              ************************************************************
E706:              *
E706:              *  SUBROUTINE BCDBIN
E706:              *
E706:              *  THIS SUBROUTINE CONVERTS A BYTE FROM BCD TO BINARY.
E706:              *  THE BYTE IS PASSED AND RETURNED IN A.  THERE IS NO
E706:              *  ERROR CHECKING.  Y IS DESTROYED AND X IS UNCHANGED.
E706:              *
E706:              ************************************************************
E706:              *
E706:        E706  BCDBIN     EQU   *
E706:48                       PHA
E707:4A                       LSR   A                ;ISOLATE TENS DIGIT FOR
E708:4A                       LSR   A                ;  INDEXING THE TABLE
E709:4A                       LSR   A
E70A:4A                       LSR   A
E70B:A8                       TAY
E70C:68                       PLA
E70D:29 0F                    AND   #$0F             ;GET UNITS
E70F:18                       CLC
E710:79 14 E7                 ADC   TENS,Y           ;ADD IN TENS
E713:60                       RTS
E714:              *
E714:00 0A 14 1E   TENS       DFB   00,10,20,30,40,50,60,70,80,90
```

```
E71E:              ************************************************************
E71E:              *
E71E:              *  SOS CALL $64 -- JOYSTICK INPUT
E71E:              *    JOYSTICK(IN.J.MODE; OUT.J.VALUE)
E71E:              *
E71E:              ************************************************************
E71E:              *
E71E:              *
E71E:        00D0  AD.INPUT   EQU   $D0
E71E:        00D1  AD.TEMP    EQU   $D1
E71E:              *
E71E:        C061  PA.SW0     EQU   $C061            ;PORT A, SWITCH 0
E71E:        C063  PA.SW1     EQU   $C063            ;PORT A, SWITCH 1
E71E:        C062  PB.SW0     EQU   $C062            ;PORT B, SWITCH 0
E71E:        C060  PB.SW1     EQU   $C060            ;PORT B, SWITCH 1
E71E:              *
E71E:        C058  AD.SEL0    EQU   $C058            ;A/D SELECT CONTROLS
E71E:        C05E  AD.SEL1    EQU   $C05E
E71E:        C05A  AD.SEL2    EQU   $C05A
E71E:        C05C  AD.CHRG    EQU   $C05C            ;A/D RAMP CHARGE /
E71E:        C05D  AD.STRT    EQU   $C05D            ;    START TIMEOUT
E71E:        C066  AD.FLAG    EQU   $C066            ;A/D TIMEOUT FLAG
E71E:              *
E71E:        01F4  TCHARGE    EQU   500              ;CHARGE TIME FOR A/D
E71E:        0168  TOFFSET    EQU   360              ;OFFSET TIME TO A/D WINDOW
E71E:              *
E71E:        F4A8  ANALOG     EQU   $F4A8            ;ROM ENTRY FOR ANALOG INPUT
E71E:        F4AB  ANLOG1     EQU   $F4AB            ;  INTERRUPT REENTRY
E71E:        FFD8  D.T2       EQU   $FFD8            ;TIMER
E71E:        FFDB  D.ACR      EQU   $FFDB            ;AUXILIARY CONTROL REGISTER
E71E:        FFDD  D.IFR      EQU   $FFDD            ;INTERRUPT FLAG REGISTER
E71E:              *
E71E:        C0DC  ENSEL      EQU   $C0DC
E71E:        C0DE  ENSIO      EQU   $C0DE
E71E:              *
E71E:              *
E71E:        E71E  JOYSTICK   EQU   *
E71E:A5 C1                    LDA   J.MODE           ;VALIDATE J.MODE
E720:C9 08                    CMP   #$08
E722:90 05   E729             BCC   JS010
E724:A9 00                    LDA   #>BADJMODE
E726:20 00 00      JS.ERR     JSR   SYSERR
E729:              *
E729:20 7C E7      JS010      JSR   AD.SETUP         ;SET UP RESOURCES
E72C:B0 F8   E726             BCS   JS.ERR
E72E:A5 C1                    LDA   J.MODE           ;READ PORT B OR PORT A?
E730:29 04                    AND   #BITON2
E732:D0 0A   E73E             BNE   JS020
E734:AD 62 C0                 LDA   PB.SW0           ;PORT B
E737:AE 60 C0                 LDX   PB.SW1
E73A:A0 01                    LDY   #$01
E73C:D0 08   E746             BNE   JS030
E73E:AD 61 C0      JS020      LDA   PA.SW0           ;PORT A
E741:AE 63 C0                 LDX   PA.SW1
E744:A0 03                    LDY   #$03
E746:84 D0         JS030      STY   AD.INPUT         ;SAVE INPUT SELECT
```

```
E748:29 80         587            AND   #BITON7
E74A:F0 02   E74E  588            BEQ   JS040
E74C:A9 FF         589            LDA   #$FF
E74E:A0 00         590 JS040      LDY   #$00
E750:91 C2         591            STA   (J.VALUE),Y      ;RETURN SWITCH 0
E752:8A            592            TXA
E753:29 80         593            AND   #BITON7
E755:F0 02   E759  594            BEQ   JS050
E757:A9 FF         595            LDA   #$FF
E759:C8            596 JS050      INY
E75A:91 C2         597            STA   (J.VALUE),Y      ;RETURN SWITCH 1
E75C:              598 *
E75C:46 C1         599            LSR   J.MODE
E75E:90 09   E769  600            BCC   JS060
E760:A5 D0         601            LDA   AD.INPUT
E762:20 C7 E7      602            JSR   AD.READ          ;READ A/D
E765:A0 02         603            LDY   #$02
E767:91 C2         604            STA   (J.VALUE),Y      ;RETURN X AXIS
E769:E6 D0         605 JS060      INC   AD.INPUT
E76B:46 C1         606            LSR   J.MODE
E76D:90 09   E778  607            BCC   JS070
E76F:A5 D0         608            LDA   AD.INPUT
E771:20 C7 E7      609            JSR   AD.READ          ;READ A/D
E774:A0 03         610            LDY   #$03
E776:91 C2         611            STA   (J.VALUE),Y      ;RETURN Y AXIS
E778:              612 *
E778:20 B5 E7      613 JS070      JSR   AD.CLNUP         ;CLEAN UP
E77B:60            614            RTS                    ;  AND EXIT
```

```
E77C:              ************************************************************
E77C:              *
E77C:              *  SUBROUTINE AD.SETUP
E77C:              *  THIS SUBROUTINE SETS UP THE ENVIRONMENT AND RESOURCES
E77C:              *  FOR READING THE JOYSTICKS.  IF AN ERROR OCCURS, CARRY
E77C:              *  IS SET AND AN ERROR NUMBER IS RETURNED IN A.
E77C:              *  OTHERWISE, CARRY IS CLEARED.
E77C:              *
E77C:              ************************************************************
E77C:        E77C  AD.SETUP   EQU   *
E77C:A9 0F                    LDA   #JOYSIRSIZ
E77E:A2 A6                    LDX   #>JOYSIRTBL
E780:A0 E7                    LDY   #<JOYSIRTBL
E782:20 00 00                 JSR   ALLOCSIR         ;ALLOCATE RESOURCES
E785:90 03   E78A             BCC   ADS010
E787:A9 00                    LDA   #>XNORESRC
E789:60                       RTS
E78A:AD DF FF      ADS010     LDA   E.REG
E78D:29 7F                    AND   #$7F             ;SET 2 MHZ,
E78F:09 43                    ORA   #$43             ;  ENABLE ROM, & I/O SPACE
E791:8D DF FF                 STA   E.REG
E794:08                       PHP
E795:78                       SEI
E796:AD DB FF                 LDA   D.ACR
E799:29 DF                    AND   #BITOFF5         ;SET UP TIMER
E79B:8D DB FF                 STA   D.ACR
E79E:28                       PLP
E79F:2C DC C0                 BIT   ENSEL            ;DISABLE ENSEL
E7A2:2C DE C0                 BIT   ENSIO            ;SET ENSIO FOR INPUT
E7A5:60                       RTS
E7A6:              *
E7A6:        E7A6  JOYSIRTBL  EQU   *
E7A6:0C 00 00 00              DFB   $0C,0,0,0,0      ;ENSIO
E7AB:0D 00 00 00              DFB   $0D,0,0,0,0      ;ENSEL
E7B0:0E 00 00 00              DFB   $0E,0,0,0,0      ;6522 D.T2
E7B5:        000F  JOYSIRSIZ  EQU   *-JOYSIRTBL
E7B5:              ************************************************************
E7B5:              *
E7B5:              *  SUBROUTINE AD.CLNUP
E7B5:              *  THIS SUBROUTINE RESTORES THE ENVIRONMENT AND RELEASES
E7B5:              *  THE RESOURCES AFTER READING THE JOYSTICKS.
E7B5:              *
E7B5:              ************************************************************
E7B5:        E7B5  AD.CLNUP   EQU   *
E7B5:AD DF FF                 LDA   E.REG
E7B8:29 3C                    AND   #$3C             ;RESTORE RAM AT $C000 & $F000
E7BA:8D DF FF                 STA   E.REG
E7BD:A9 0F                    LDA   #JOYSIRSIZ
E7BF:A2 A6                    LDX   #>JOYSIRTBL
E7C1:A0 E7                    LDY   #<JOYSIRTBL
E7C3:20 00 00                 JSR   DEALCSIR         ;DEALLOCATE RESOURCES
E7C6:60                       RTS
```

```
E7C7:              ************************************************************
E7C7:              *
E7C7:              *  SUBROUTINE AD.READ
E7C7:              *  THIS SUBROUTINE READS A SPECIFIED A/D INPUT AND RETURNS
E7C7:              *  AN 8 BIT RESULT.  IT ASSUMES THAT THE A/D RESOURCES HAVE
E7C7:              *  BEEN ALLOCATED, THE I/O SPACE AND $F000 ROM HAVE BEEN
E7C7:              *  SELECTED, AND THE SYSTEM IS RUNNING IN 2 MHZ MODE.
E7C7:              *
E7C7:              *  PARAMETERS:
E7C7:              *    A:  A/D INPUT PORT (0-7)
E7C7:              *
E7C7:              *  RETURN VALUE:
E7C7:              *    A:  RESULT (0 - 255)
E7C7:              *    X, Y:  UNDEFINED
E7C7:              *
E7C7:              ************************************************************
E7C7:              *
E7C7:        E7C7  AD.READ    EQU   *
E7C7:4A                       LSR   A                ;SELECT THE APPROPRIATE
E7C8:2C 58 C0                 BIT   AD.SEL0          ;  A/D INPUT
E7CB:90 03   E7D0             BCC   ADR010
E7CD:2C 59 C0                 BIT   AD.SEL0+1
E7D0:4A            ADR010     LSR   A
E7D1:2C 5E C0                 BIT   AD.SEL1
E7D4:90 03   E7D9             BCC   ADR020
E7D6:2C 5F C0                 BIT   AD.SEL1+1
E7D9:4A            ADR020     LSR   A
E7DA:2C 5A C0                 BIT   AD.SEL2
E7DD:90 03   E7E2             BCC   ADR030
E7DF:2C 5B C0                 BIT   AD.SEL2+1
E7E2:08            ADR030     PHP
E7E3:              *
E7E3:58            ADR040     CLI
E7E4:2C 5C C0                 BIT   AD.CHRG          ;CHARGE A/D CAPACITOR
E7E7:A9 F4                    LDA   #>TCHARGE
E7E9:8D D8 FF                 STA   D.T2
E7EC:A9 01                    LDA   #<TCHARGE
E7EE:8D D9 FF                 STA   D.T2+1
E7F1:A9 20                    LDA   #BITON5
E7F3:2C DD FF      ADR050     BIT   D.IFR
E7F6:F0 FB   E7F3             BEQ   ADR050
E7F8:              *
E7F8:78                       SEI
E7F9:38                       SEC
E7FA:A9 68                    LDA   #>TOFFSET
E7FC:8D D8 FF                 STA   D.T2             ;SET UP TIMER
E7FF:A9 01                    LDA   #<TOFFSET
E801:2C 5D C0                 BIT   AD.STRT          ;START A/D TIMEOUT
E804:20 A8 F4                 JSR   ANALOG           ;MEASURE CONVERSION TIME
E807:90 0C   E815             BCC   ADR070
E809:              *
E809:58            ADR060     CLI                    ;PROCESS AN INTERRUPT
E80A:78                       SEI
E80B:2C 66 C0                 BIT   AD.FLAG          ;STILL TIMING?
E80E:10 D3   E7E3             BPL   ADR040           ;  NO -- START OVER
E810:20 AB F4                 JSR   ANLOG1           ;  YES -- CONTINUE
```

```
E813:B0 F4   E809              BCS   ADR060
E815:                    *
E815:28                  ADR070     PLP
E816:49 FF                     EOR   #$FF             ;NORMALIZE RESULT
E818:30 13   E82D              BMI   ADR080           ;RESULT < 0
E81A:85 D1                     STA   AD.TEMP
E81C:98                        TYA
E81D:49 FF                     EOR   #$FF
E81F:46 D1                     LSR   AD.TEMP
E821:6A                        ROR   A
E822:46 D1                     LSR   AD.TEMP
E824:6A                        ROR   A
E825:46 D1                     LSR   AD.TEMP
E827:D0 07   E830              BNE   ADR090           ;RESULT > 255
E829:6A                        ROR   A
E82A:69 00                     ADC   #0
E82C:60                        RTS
E82D:A9 00               ADR080     LDA   #0
E82F:60                        RTS
E830:A9 FF               ADR090     LDA   #$FF
E832:60                        RTS
```

```
E833:              ************************************************************
E833:              *
E833:              *  SYSTEM COLD START
E833:              *
E833:              *  THIS ROUTINE IS CALLED TO TELL THE USER TO REBOOT THE
E833:              *  SYSTEM.  IT CLEARS THE SCREEN, DISPLAYS A MESSAGE,
E833:              *  OVERWRITES BANKED MEMORY, AND HANGS UNTIL THE USER
E833:              *  PERFORMS A HARD RESET.
E833:              *
E833:              ************************************************************
E833:              *
E833:              *
E833:        E833  COLDSTRT   EQU   *
E833:78                       SEI                    ;SHUT DOWN INTERRUPTS
E834:A9 40                    LDA   #$40             ;  AND IGNORE NMI
E836:8D CA FF                 STA   $FFCA
E839:A9 67                    LDA   #$67
E83B:8D DF FF                 STA   E.REG            ;DISABLE RESET
E83E:A9 00                    LDA   #$00
E840:8D D0 FF                 STA   Z.REG            ;USE PAGE ZERO
E843:              *
E843:AE 00 00                 LDX   SYSBANK
E846:A9 BF                    LDA   #$BF
E848:A0 00                    LDY   #$00
E84A:84 C1                    STY   MEMORY
E84C:85 C2         CS010      STA   MEMORY+1
E84E:8E EF FF                 STX   B.REG
E851:A9 A0                    LDA   #$A0
E853:91 C1         CS020      STA   (MEMORY),Y       ;SET MEMORY TO BLANKS
E855:88                       DEY
E856:D0 FB   E853             BNE   CS020
E858:C6 C2                    DEC   MEMORY+1
E85A:D0 F7   E853             BNE   CS020
E85C:CA                       DEX
E85D:10 ED   E84C             BPL   CS010
E85F:              *
E85F:A0 06                    LDY   #6
E861:99 50 C0      CS030      STA   $C050,Y          ;SELECT 40 COLUMN
E864:88                       DEY                    ;  BLACK & WHITE TEXT
E865:10 FA   E861             BPL   CS030
E867:              *
E867:A0 1F                    LDY   #BOOTLEN
E869:B9 79 E8      CS040      LDA   BOOTMSG-1,Y      ;PRINT BOOT MESSAGE
E86C:99 2B 06                 STA   BOOTADR-1,Y
E86F:88                       DEY
E870:D0 F7   E869             BNE   CS040
E872:              *
E872:A9 77                    LDA   #$77
E874:8D DF FF                 STA   E.REG            ;ENABLE RESET
E877:4C 77 E8                 JMP   *                ;HANG UNTIL RESET
```

```
E87A:              798            MSB   ON
E87A:C9 CE D3 C5   799 BOOTMSG    ASC   "INSERT          SYSTEM DISKETTE & REBOOT"
E899:        001F  800 BOOTLEN    EQU   *-BOOTMSG
E899:        062C  801 BOOTADR    EQU   40-BOOTLEN/2+$628
E899:              802            MSB   OFF

E899:              803            LST   ON
E899:        E899  804 ZZEND      EQU   *
E899:        040E  805 ZZLEN      EQU   ZZEND-ZZORG
E899:        0000  806            IFNE  ZZLEN-LENUMGR
 S                 807            FAIL  2,"SOSORG        FILE IS INCORRECT FOR UMBR"
E899:              808            FIN
```

```
 C05C AD.CHRG        E7B5 AD.CLNUP       C066 AD.FLAG          D0 AD.INPUT
 E7C7 AD.READ        C058 AD.SEL0        C05E AD.SEL1        C05A AD.SEL2
 E77C AD.SETUP       C05D AD.STRT          D1 AD.TEMP        E7D0 ADR010
 E7D9 ADR020         E7E2 ADR030         E7E3 ADR040         E7F3 ADR050
 E809 ADR060         E815 ADR070         E82D ADR080         E830 ADR090
 E78A ADS010        X000C ALLOCSIR       F4A8 ANALOG         F4AB ANLOG1
 FFEF B.REG         X000A BADJMODE      X0009 BADSCNUM      NE706 BCDBIN
   DF BITOFF5          04 BITON2           20 BITON5           40 BITON6
   80 BITON7        ?2E00 BLABFMI        3200 BLABFM         6B52 BLABUFMG
 6955 BLACFM         5E99 BLADISK3       64D9 BLADMGR        68F4 BLAFMGR
?2CF8 BLAGLOB       ?2AF8 BLAINIT        55C0 BLAIPL         2000 BLALODR
?6E6E BLAMEMMG       5466 BLAOMSG        5466 BLAPATCH       665E BLASCMGR
 6404 BLASERR        5A8B BLAUMGR        062C BOOTADR        001F BOOTLEN
 E87A BOOTMSG       X0007 CEVPRI           D3 CKSUM          E651 CLKREGS
 18D4 CLKTEMP        C070 CLOCK            03 CMIN             07 CMON
NE833 COLDSTRT         12 CRESET         E84C CS010          E853 CS020
 E861 CS030          E869 CS040            02 CSEC           FFDB D.ACR
 FFDD D.IFR          FFD8 D.T2          NE656 DATETIME       E654 DAY
X000D DEALCSIR       E675 DT010          E679 DT020          E695 DT030
 E69F DT040          E6AE DT050          E6C9 DT060          FFDF E.REG
 C0DC ENSEL          C0DE ENSIO          E650 ERRCNT         E4BB GET.FENCE
 E5C1 GET.TIME       E5DD GTIM010        E5E1 GTIM020        E5FD GTIM030
 E614 GTIM040        E61D GTIM050        E62E GTIM060        E63D GTIM070
 E64E GTIM080        E652 HOUR             C1 J.MODE           C2 J.VALUE
 000F JOYSIRSIZ      E7A6 JOYSIRTBL      E71E JOYSTICK       E726 JS.ERR
 E729 JS010          E73E JS020          E746 JS030          E74E JS040
 E759 JS050          E769 JS060          E778 JS070            0E LDAY
 2266 LENBFM        ?0400 LENBFMI        031C LENBUFMG       01FD LENCFM
 056B LENDISK3       0185 LENDMGR          61 LENFMGR       ?01B2 LENINIT
 04CB LENIPL         0AF8 LENLODR       ?0751 LENMEMMG       015A LENOMSG
   00 LENPATCH       0296 LENSCMGR         D5 LENSERR        040E LENUMGR
   C1 MEMORY         E651 MIN            E655 MON            E64F OFFSET
 BC00 ORGBFM         B800 ORGBFMI        F552 ORGBUFMG       F355 ORGCFM
 E899 ORGDISK3       EED9 ORGDMGR        FFBF ORGEND         F2F4 ORGFMGR
?18FC ORGGLOB        28F8 ORGINIT        DFC0 ORGIPL         1E00 ORGLODR
 F86E ORGMEMMG       DE66 ORGOMSG        DE66 ORGPATCH       F05E ORGSCMGR
 EE04 ORGSERR        E48B ORGUMGR        C061 PA.SW0         C063 PA.SW1
 C062 PB.SW0         C060 PB.SW1           D0 PCLK          NE4C3 PCLOCK
   C1 PRIORITY       E4CD RETRY          E4B5 SET.FENCE      E4DA SET.TIME
   14 STATUS         E4E2 STIM010        E4E5 STIM011        E51A STIM015
 E52D STIM016        E549 STIM020        E57F STIM030        E582 STIM040
 E5B8 STIM050       X0006 SYSBANK       X0008 SYSERR         01F4 TCHARGE
 E714 TENS             C1 TIME           0168 TOFFSET          C0 U.REQCODE
   C0 U.TPARMX       E4A4 UMGRERR       NE48B UMGR           0006 USWCNT
 E4A9 USWTBL           D2 WKDAY          E4CE WKMON         X000B XNORESRC
 E653 YEAR           FFD0 Z.REG          E899 ZZEND          040E ZZLEN
 E48B ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   867
** FREE SPACE PAGE COUNT   79
```

```
SOURCE   FILE #01 =>DISK3.SRC
 INCLUDE FILE #02 =>SOSORG
SOURCE   FILE #03 =>DISK3.MAIN.SRC
SOURCE   FILE #04 =>DISK3.WRT.SRC
SOURCE   FILE #05 =>DISK3.SIO.SRC
SOURCE   FILE #06 =>DISK3.USEL.SRC
SOURCE   FILE #07 =>DISK3.SUBS.SRC
SOURCE   FILE #08 =>DISK3.DATA.SRC
```

```
0000:        0000      TEST       EQU   0                ;FOR FUNNY-MODE TESTING
0000:                             INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955   BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52   BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E   BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
0000:        0000              DO    TEST
 S                             ORG   $2000
0000:                          ELSE
0000:                          REL
E899:        E899              ORG   ORGDISK3
E899:                          FIN
E899:        E899   ZZORG      EQU   *
E899:                          CHR   '-'
E899:                          MSB   OFF
E899:               *
E899:               ----------------------------------------
E899:               *    COPYRIGHT (C) APPLE COMPUTER INC.
E899:               *          ALL RIGHTS RESERVED
E899:               ----------------------------------------
E899:               *
E899:        0000   REV0ROM    EQU   0                ;1=SUPPORT REV0 ROM
E899:               *
E899:        0001              DO    1-TEST
E899:        E899              ENTRY DIB1             ;DIB1
E899:        E8B9              ENTRY DIB2             ;DIB2
E899:        E8D9              ENTRY DIB3             ;DIB3
E899:        E8F9              ENTRY DIB4             ;DIB4
E899:        ED57              ENTRY SEEKDSK3         ;SEEK CURRENT DRIVE
E899:               *
E899:        0000              EXTRN SYSERR
E899:               *
E899:        0000              EXTRN XREQCODE
E899:        0000              EXTRN XBADOP
E899:        0000              EXTRN XNODRIVE
E899:        0000              EXTRN XIOERROR
E899:        0000              EXTRN XNOWRITE
E899:        0000              EXTRN XBYTECNT
E899:        0000              EXTRN XBLKNUM
E899:        0000              EXTRN XDISKSW
E899:        0000              EXTRN XCTLCODE
E899:               *
E899:        0000              EXTRN E1908            ; GLOBAL FLAG FOR MOUSE DRIVER
E899:               * TO SAY WE CANNOT BE INTERRUPTED
E899:               *
E899:                          ELSE
 S                  XREQCODE   EQU   $20
 S                  XBADOP     EQU   $26
 S                  XNODRIVE   EQU   $28
 S                  XIOERROR   EQU   $27
 S                  XNOWRITE   EQU   $2B
 S                  XBYTECNT   EQU   $2C
 S                  XBLKNUM    EQU   $2D
 S                  XDISKSW    EQU   $2E
 S                  XCTLCODE   EQU   $21
E899:                          FIN
```

```
E899:               * DISK /// CONTROLLER EQUATES:
E899:               *
E899:               *       MOTOR SELECT BITS:
E899:               *
E899:               *    DRIVE    INT   EXT1   EXT2
E899:               *    -----    ---   ----   ----
E899:               *     .D1      1     X      X
E899:               *     .D2      X     0      1
E899:               *     .D3      X     1      0
E899:               *     .D4      X     1      1
E899:               *
E899:        C0D4   MS.INT     EQU   $C0D4            ;MOTOR   SELECT:INTERNAL DRIVE
E899:        C0D5   MD.INT     EQU   $C0D5            ;MOTOR DESELECT:INTERNAL DRIVE
E899:               *
E899:        C0D3   MS.EXT1    EQU   $C0D3            ;MOTOR   SELECT:EXTERNAL DRIVE
E899:        C0D1   MS.EXT2    EQU   $C0D1            ;MOTOR   SELECT:EXTERNAL DRIVE
E899:        C0D2   MD.EXT1    EQU   $C0D2            ;MOTOR DESELECT:EXTERNAL DRIVE
E899:        C0D0   MD.EXT2    EQU   $C0D0            ;MOTOR DESELECT:EXTERNAL DRIVE
E899:               *
E899:        C0EA   IS.INT     EQU   $C0EA            ;I/O SELECT:INTERNAL DRIVE
E899:        C0EB   IS.EXT     EQU   $C0EB            ;I/O SELECT:EXTERNAL DRIVE
E899:               *
E899:        C0D8   NOSCROLL   EQU   $C0D8            ;SMOOTHSCROLL OFF
E899:               *
E899:        C0E8   MOTOROFF   EQU   $C0E8            ;MOTOR(S) START POWEROFF T/O
E899:        C0E9   MOTORON    EQU   $C0E9            ;MOTOR(S) POWER ON
E899:        C08C   Q6L        EQU   $C08C            ;Q7L,Q6L=READ
E899:        C08D   Q6H        EQU   $C08D            ;Q7L,Q6H=SENSE WPROT
E899:        C08E   Q7L        EQU   $C08E            ;Q7H,Q6L=WRITE
E899:        C08F   Q7H        EQU   $C08F            ;Q7H,Q6H=WRITE STORE
E899:               *
E899:               * OTHER EQUATES:
E899:               *
E899:        FFDF   E.REG      EQU   $FFDF            ;ENVIRONMENT REGISTER
E899:        FFEE   E.IER      EQU   $FFEE            ;INTERRUPT ENABLE REGISTER
E899:               *
E899:               * RETRY COUNTERS:
E899:               *
E899:        0001   R.RECAL    EQU   1                ;MAX RECALIBRATES
E899:               * R.RECAL MUST NOT BECOME ZERO! (MOUSE WILL BE LOCKED OUT)
E899:               * SEE DISK3.SIO.SRC LINE 14 FOR DETAIL
E899:        0003   R.FIND     EQU   3                ;MAX REVS TO FIND A SECTOR
E899:        0004   R.IOERR    EQU   4                ;MAX RETRIES ON READ ERROR
E899:        0006   R.IRQ      EQU   6                ;MAX IRQ'S TOLERATED BEFORE SEI
```

```
E899:              * ZPAGE EQUATES FOR CORE ROUTINES:
E899:              *
0000:                         DSECT
0081:        0081             ORG   $81
0081:        0001  IBSLOT     DS    1                ;SLOT=$60 FOR RTNS
0082:        0007             DS    7                ;N/A
0089:        0001             DS    1                ;RDADR:CHECKSUM
008A:        0001             DS    1                ;N/A
008B:        0001  IMASK      DS    1                ;BIT7 SET IF IRQ ALLOWED
008C:        0001  CURTRK     DS    1                ;SEEK:CURRENT TRACK
008D:        0002             DS    2                ;N/A
008F:        0001  INTRTRY    DS    1                ;READ: IRQ RETRY COUNT
0090:        0005             DS    5                ;N/A
0095:        0001             DS    1                ;RDADR:'MUST FIND' COUNT
0096:        0001             DS    1                ;READ,WRITE: CHECKSUM
0097:        0004  CSSTV      DS    4                ;RDADR:CKSUM,SEC,TRK,VOL
009B:        0099  MONTIMEL   EQU   CSSTV+2          ;MSWAIT:MOTOR-ON TIME
009B:        009A  MONTIMEH   EQU   MONTIMEL+1
009B:        0002  BUF        DS    2                ;PRENIB,POSTNIB:USER BUFFER
009D:        0001             DS    1                ;SEEK:PRIOR PHASE
009E:        0001  TRKN       DS    1                ;SEEK:TARGET TRACK
009F:              *
009F:              * LOCAL TEMPS:
009F:              *
00D0:        00D0             ORG   $D0              ;WE'RE ALLOWED TO $FF
00D0:        0002  BLKTEMP    DS    2                ;LOCAL TEMP FOR BLKNUMBER
00D2:        0002  BUFTEMP    DS    2                ;LOCAL TEMP FOR BUFFER ADDRESS
00D4:        0001  TRACK      DS    1                ;LOCAL TEMP FOR TRACK
00D5:        0001  SECTOR     DS    1                ;LOCAL TEMP FOR SECTOR
00D6:        0001  RETRYADR   DS    1                ;LOCAL TEMP FOR SECTOR-FIND RETRIES
00D7:        0001  RETRYCNT   DS    1                ;LOCAL TEMP FOR I/O RETRIES
00D8:        0001  RECALCNT   DS    1                ;LOCAL TEMP FOR RECAL COUNT
00D9:        0001  BLKCOUNT   DS    1                ;BLKS REQD TO SATISFY BYTECOUNT
00DA:        0001  SEEKWAIT   DS    1                ;<>0 IF SEEK DELAY NEEDED
00DB:        0001  IRQMASK    DS    1                ;ENTRY 'I' BIT
00DC:        0001  TEMP       DS    1                ;JUST A TEMP
E899:                         DEND
```

```
E899:              * DRIVER INTERFACE AREA:
E899:              *
0000:                         DSECT
00C0:        00C0             ORG   $C0
00C0:        0001  D.COMMAND  DS    1                ;COMMAND CODE
00C1:        0001  D.UNITNUM  DS    1                ;UNIT NUMBER
00C2:        0002  D.BUFL     DS    2                ;BUFFER ADDRESS
00C4:        00C3  D.BUFH     EQU   D.BUFL+1
00C4:        00C2  D.STATCODE EQU   D.BUFL           ;DSTATUS CODE
00C4:        00C3  D.STATBUF  EQU   D.BUFH           ;^DSTATUS LIST
00C4:        0002  D.BYTES    DS    2                ;BYTECOUNT
00C6:        0002  D.BLOCK    DS    2                ;REQUESTED BLOCKNUM
00C8:        0002  D.BYTRD    DS    2                ;BYTES READ (READ)
00CA:        0006             DS    6                ;SPARES (OK AS TEMPS)
E899:                         DEND
```

```
E899:        E899  DIB1       EQU   *                ;DIB FOR .D1
E899:B9 E8                    DW    DIB2             ;FLINK
E89B:1D E9                    DW    MAIN             ;ENTRY POINT
E89D:03                       DFB   3                ;NAME LENGTH
E89E:2E 44 31 20              ASC   '.D1                        '
E8AD:80                       DFB   $80              ;DEVNUM: ACTIVE
E8AE:00                       DFB   0                ;SLOT
E8AF:00                       DFB   0                ;UNIT NUMBER
E8B0:E1 01 00                 DFB   $E1,1,0          ;TYPE,SUB,FILLER
E8B3:18 01                    DW    280              ;BLOCKCOUNT
E8B5:01 00                    DW    1                ;MANUFACTURER=APPLE
E8B7:00 11                    DW    $1100            ;VERSION=1.1
E8B9:              *
E8B9:        E8B9  DIB2       EQU   *                ;DIB FOR .D2
E8B9:D9 E8                    DW    DIB3             ;FLINK
E8BB:1D E9                    DW    MAIN             ;ENTRY POINT
E8BD:03                       DFB   3                ;NAME LENGTH
E8BE:2E 44 32 20              ASC   '.D2                        '
E8CD:80                       DFB   $80              ;DEVNUM: ACTIVE
E8CE:00                       DFB   0                ;SLOT
E8CF:01                       DFB   1                ;UNIT NUMBER
E8D0:E1 01 00                 DFB   $E1,1,0          ;TYPE,SUB,FILLER
E8D3:18 01                    DW    280              ;BLOCKCOUNT
E8D5:01 00                    DW    1                ;MANUFACTURER=APPLE
E8D7:00 11                    DW    $1100            ;VERSION=1.1
E8D9:              *
E8D9:        E8D9  DIB3       EQU   *                ;DIB FOR .D3
E8D9:F9 E8                    DW    DIB4             ;FLINK
E8DB:1D E9                    DW    MAIN             ;ENTRY POINT
E8DD:03                       DFB   3                ;NAME LENGTH
E8DE:2E 44 33 20              ASC   '.D3                        '
E8ED:80                       DFB   $80              ;DEVNUM: ACTIVE
E8EE:00                       DFB   0                ;SLOT
E8EF:02                       DFB   2                ;UNIT NUMBER
E8F0:E1 01 00                 DFB   $E1,1,0          ;TYPE,SUB,FILLER
E8F3:18 01                    DW    280              ;BLOCKCOUNT
E8F5:01 00                    DW    1                ;MANUFACTURER=APPLE
E8F7:00 11                    DW    $1100            ;VERSION=1.1
E8F9:              *
E8F9:        E8F9  DIB4       EQU   *                ;DIB FOR .D4
E8F9:00 00                    DW    0                ;NO FLINK
E8FB:1D E9                    DW    MAIN             ;ENTRY POINT
E8FD:03                       DFB   3                ;NAME LENGTH
E8FE:2E 44 34 20              ASC   '.D4                        '
E90D:80                       DFB   $80              ;DEVNUM: ACTIVE
E90E:00                       DFB   0                ;SLOT
E90F:03                       DFB   3                ;UNIT NUMBER
E910:E1 01 00                 DFB   $E1,1,0          ;TYPE,SUB,FILLER
E913:18 01                    DW    280              ;BLOCKCOUNT
E915:01 00                    DW    1                ;MANUFACTURER=APPLE
E917:00 11                    DW    $1100            ;VERSION=1.1
E919:01 00                    DW    1                ;MANUFACTURER=APPLE
E91B:00 11                    DW    $1100            ;VERSION=1.1
E91D:              *
E91D:                         CHN   DISK3.MAIN.SRC
```

```
E91D:              * MAIN ENTRY POINT:
E91D:              *
E91D:              * DISABLE NMI/RESET AND ENABLE ROM/IO SPACE
E91D:              *
E91D:        E91D  MAIN       EQU   *
E91D:AD DF FF                 LDA   E.REG            ;SAVE CALLER'S
E920:29 DF                    AND   #$FF-$20         ;DROP SCREEN BIT
E922:8D F2 ED                 STA   ESAVE            ; ENVIRONMENT
E925:        0001             DO    1-TEST           ;NO RESETLOCK FOR TESTING
E925:AD DF FF                 LDA   E.REG            ;GET EREG AGAIN
E928:29 EF                    AND   #$FF-$10         ;DISABLE NMI/RESET
E92A:                         FIN
E92A:09 03                    ORA   #$03             ;ENABLE ROM/IO SPACE
E92C:8D DF FF                 STA   E.REG
E92F:              *
E92F:AD D8 C0                 LDA   NOSCROLL         ;DISABLE SMOOTHSCROLL
E932:              *
E932:08                       PHP                    ;IF ALREADY SEI'D, THEN WE
E933:68                       PLA                    ; STAY THAT WAY...
E934:6A                       ROR   A
E935:6A                       ROR   A
E936:6A                       ROR   A
E937:6A                       ROR   A
E938:85 DB                    STA   IRQMASK          ;'I' BIT INTO BIT7
E93A:              *
E93A:              * MAKE SURE WE HAVE A VALID COMMAND:
E93A:              *
E93A:A5 C0                    LDA   D.COMMAND        ;GET IT
E93C:30 43   E981             BMI   BADCMD           ;=>WOW!
E93E:F0 46   E986             BEQ   IOSETUP          ;=>ZERO IS A READ
E940:C9 0A                    CMP   #10              ;OFF THE END?
E942:B0 3D   E981             BCS   BADCMD           ;=>YES
E944:C9 09                    CMP   #9               ;REPEAT?
E946:D0 16   E95E             BNE   CMD1             ;=>NOPE
E948:              *
E948:              * REPEAT. SIMPLY GET PRIOR COMMAND:
E948:              *
E948:AD F0 ED                 LDA   PREVUNIT         ;IS THIS REPEAT FOR
E94B:C5 C1                    CMP   D.UNITNUM        ; SAME UNIT?
E94D:D0 2D   E97C             BNE   BADOP            ;=>NO? ILLEGAL!
E94F:AD F1 ED                 LDA   PREVCMD          ;YES, SET COMMAND
E952:F0 04   E958             BEQ   RPTOK            ;=>REPEAT'ED READ IS OK
E954:C9 01                    CMP   #1               ;IF NOT, IS IT REPEAT'ED WRITE?
E956:D0 24   E97C             BNE   BADOP            ;=>CAN'T REPEAT OTHER COMMANDS
E958:        E958  RPTOK      EQU   *
E958:85 C0                    STA   D.COMMAND        ;SAME AS BEFORE
E95A:C9 00                    CMP   #0               ;READ?
E95C:F0 28   E986             BEQ   IOSETUP          ;=>YES
E95E:              * NOW REPEAT GOES LIKE OTHERS:
E95E:              *
E95E:              *
E95E:        E95E  CMD1       EQU   *
E95E:C9 01                    CMP   #1               ;WRITE?
E960:D0 03   E965             BNE   CMD2             ;=>NOPE
E962:4C 86 E9                 JMP   IOSETUP          ;=>YES
E965:        E965  CMD2       EQU   *
```

```
E965:C9 02                         CMP   #2               ;STATUS?
E967:D0 0C   E975                  BNE   CMD3             ;=>NOT STATUS
E969:A5 C2                         LDA   D.STATCODE       ;IS IT 'SENSE'?
E96B:F0 05   E972                  BEQ   GOSTAT           ;=>YES
E96D:A9 00                         LDA   #XCTLCODE        ;ILLEGAL CODE
E96F:4C E9 EA                      JMP   EXIT
E972:        E972       GOSTAT     EQU   *
E972:4C BC E9                      JMP   DRVSETUP         ;=>YES
E975:                   *
E975:        E975       CMD3       EQU   *
E975:C9 08                         CMP   #8               ;INIT?
E977:D0 03   E97C                  BNE   BADOP            ;=>NOPE
E979:4C A4 EA                      JMP   INIT             ;=>YES, DO INIT
E97C:                   *
E97C:        E97C       BADOP      EQU   *
E97C:A9 00                         LDA   #XBADOP          ;ILLEGAL COMMAND
E97E:4C E9 EA                      JMP   EXIT             ;BACK TO YOU
E981:                   *
E981:        E981       BADCMD     EQU   *
E981:A9 00                         LDA   #XREQCODE        ;INVALID COMMAND
E983:4C E9 EA                      JMP   EXIT             ;BACK TO YOU
```

```
E986:               80 * SETUP WHAT WE HAVE TO BEFORE
E986:               81 *  PERFORMING THE I/O OPERATION:
E986:               82 *
E986:        E986   83 IOSETUP    EQU   *
E986:A5 C7          84            LDA   D.BLOCK+1        ;VALIDATE BLOCKNUM
E988:F0 0F   E999   85            BEQ   CHKBYTE          ;=> IF <256, IT'S OK
E98A:C9 02          86            CMP   #2               ;IS IT <512?
E98C:B0 06   E994   87            BCS   BADBLOCK         ;=>BAD BOY!
E98E:A5 C6          88            LDA   D.BLOCK          ;YES, CHECK LO HALF
E990:C9 18          89            CMP   #280-256         ; FOR RANGE
E992:90 05   E999   90            BCC   CHKBYTE          ;=>IT'S OK
E994:        E994   91 BADBLOCK   EQU   *
E994:A9 00          92            LDA   #XBLKNUM         ;BAD BLOCK NUMBER
E996:4C E9 EA       93            JMP   EXIT             ;RETURN BAD NEWS
E999:               94 *
E999:        E999   95 CHKBYTE    EQU   *
E999:A5 C4          96            LDA   D.BYTES          ;GET LO COUNT
E99B:D0 1A   E9B7   97            BNE   BADCOUNT         ;=>ERR, NOT INTEGRAL BLOCK(S)
E99D:A5 C5          98            LDA   D.BYTES+1        ;GET HI COUNT
E99F:4A             99            LSR   A                ;MAKE BLOCK COUNT
E9A0:B0 15   E9B7  100            BCS   BADCOUNT         ;=>BAD IF HALF-BLOCK COUNT
E9A2:85 D9         101            STA   BLKCOUNT         ;SAVE COUNT OF BLOCKS
E9A4:              102 *
E9A4:              103 * DOES REQUESTED BYTECOUNT CAUSE US
E9A4:              104 *  TO RUN OFF END OF DISK?
E9A4:              105 *
E9A4:A5 D9         106            LDA   BLKCOUNT         ;NO. ADD STARTBLOCK
E9A6:18            107            CLC                    ; AND BLKCOUNT AND SEE
E9A7:65 C6         108            ADC   D.BLOCK          ;  IF WE'RE TOO BIG
E9A9:A6 C7         109            LDX   D.BLOCK+1        ;DID IT START OUT > 255?
E9AB:D0 04   E9B1  110            BNE   BLKG255          ;=>YES
E9AD:90 0D   E9BC  111            BCC   DRVSETUP         ;=>DEFINITELY < 256
E9AF:B0 02   E9B3  112            BCS   CHKLO            ;=>IF CARRY,THEN >256
E9B1:        E9B1  113 BLKG255    EQU   *
E9B1:B0 04   E9B7  114            BCS   BADCOUNT         ;>255+CARRY IS NOW >511
E9B3:        E9B3  115 CHKLO      EQU   *
E9B3:C9 19         116            CMP   #280-256+1       ;281..511 ?
E9B5:90 05   E9BC  117            BCC   DRVSETUP         ;=>NO, WE ARE OK
E9B7:        E9B7  118 BADCOUNT   EQU   *
E9B7:A9 00         119            LDA   #XBYTECNT        ;ILLEGAL BYTECOUNT
E9B9:4C E9 EA      120            JMP   EXIT             ;SORRY...
```

```
E9BC:              *
E9BC:              * SELECT THE APPROPRIATE DRIVE:
E9BC:              *
E9BC:        E9BC  DRVSETUP   EQU   *
E9BC:A5 C0                    LDA   D.COMMAND        ;SAVE THIS COMMAND
E9BE:8D F1 ED                 STA   PREVCMD          ; AND DEVICE FOR
E9C1:A5 C1                    LDA   D.UNITNUM        ;  SUBSEQUENT
E9C3:8D F0 ED                 STA   PREVUNIT         ;   'REPEAT' CALL
E9C6:AD DF FF                 LDA   E.REG            ;DOWNSHIFT TO
E9C9:09 80                    ORA   #$80             ; 1MHZ FOR REMAINDER
E9CB:8D DF FF                 STA   E.REG            ;  OF DRIVER EXECUTION
E9CE:20 1D EC                 JSR   UNITSEL          ;SELECT & START IT
E9D1:              *
E9D1:              * SEE IF THE MOTOR STARTED. IF NOT,
E9D1:              *  THEN IT'S EITHER DISKSWITCH OR NODRIVE.
E9D1:              *
E9D1:20 DC EC                 JSR   CHKDRV           ;MOTOR RUNNING?
E9D4:D0 23   E9F9             BNE   DOIO             ;=>YES, GREAT.
E9D6:              *
E9D6:              * IF WE GET A MOTOR WHEN WE MOVE
E9D6:              *  THE HEAD, THEN IT'S DISKSWITCH.
E9D6:              *
E9D6:A6 C1                    LDX   D.UNITNUM        ;FORCE HEAD MOTION
E9D8:FE 00 EE                 INC   DRVTRACK,X       ; EVEN IF ALREADY ON ZERO
E9DB:FE 00 EE                 INC   DRVTRACK,X       ;GIVE HIM A FIRM KNOCKER
E9DE:A9 00                    LDA   #0               ;SEEK TO TRACK ZERO
E9E0:20 60 ED                 JSR   MYSEEK           ; FOR BFM DIR READ
E9E3:20 DC EC                 JSR   CHKDRV           ;RUNNING NOW?
E9E6:D0 0C   E9F4             BNE   DSWITCH          ;=>YES, A SWITCHEROO
E9E8:A9 00                    LDA   #0
E9EA:A4 C1                    LDY   D.UNITNUM        ;FORGET THAT THIS
E9EC:99 F8 ED                 STA   DRIVESEL,Y       ; DRIVE WAS 'SELECTED'
E9EF:A9 00                    LDA   #XNODRIVE        ;NO, A MISSING DRIVE!
E9F1:4C E9 EA                 JMP   EXIT
E9F4:              *
E9F4:        E9F4  DSWITCH    EQU   *
E9F4:A9 00                    LDA   #XDISKSW         ;USER PULLED A FAST ONE
E9F6:4C E9 EA                 JMP   EXIT             ; BUT HE CAN'T FOOL US.
```

```
E9F9:              * PREPARE TO DO THE OPERATION:
E9F9:              *
E9F9:        E9F9  DOIO       EQU   *
E9F9:A5 C2                    LDA   D.BUFL           ;COPY USER BUFFER
E9FB:85 D2                    STA   BUFTEMP          ; AND BLOCK NUMBER
E9FD:A5 C3                    LDA   D.BUFH           ;  TO OUR WORKSPACE
E9FF:85 D3                    STA   BUFTEMP+1
EA01:AD C3 14                 LDA   $1400+D.BUFH
EA04:8D D3 14                 STA   $1400+BUFTEMP+1
EA07:A5 C6                    LDA   D.BLOCK
EA09:85 D0                    STA   BLKTEMP
EA0B:A5 C7                    LDA   D.BLOCK+1
EA0D:85 D1                    STA   BLKTEMP+1
EA0F:              *
EA0F:              * IF CALLER GAVE US A COUNT OF ZERO BYTES,
EA0F:              *  THEN WE'RE ALL DONE!
EA0F:              *
EA0F:A5 C0                    LDA   D.COMMAND        ;IS IT STATUS?
EA11:C9 02                    CMP   #2               ;IF SO, THEN BYTECOUNT
EA13:D0 03   EA18             BNE   DOIO2            ; IS MEANINGLESS
EA15:4C 8B EA                 JMP   STATUS
EA18:        EA18  DOIO2      EQU   *
EA18:A4 D9                    LDY   BLKCOUNT         ;BLKS=0?
EA1A:F0 31   EA4D             BEQ   READOK           ;=>YES, YOU GET GOOD RETURN
EA1C:C9 00                    CMP   #0               ;READ COMMAND?
EA1E:F0 03   EA23             BEQ   READREQ          ;=>YES
EA20:4C 55 EA                 JMP   WRITEREQ
```

```
EA23:              ----------------------------------------
EA23:              *  -- READ --
EA23:              ----------------------------------------
EA23:        EA23  READREQ    EQU   *
EA23:A9 00                    LDA   #0               ;CLEAR COUNT OF
EA25:A0 00                    LDY   #0
EA27:91 C8                    STA   (D.BYTRD),Y      ; BYTES READ
EA29:C8                       INY
EA2A:91 C8                    STA   (D.BYTRD),Y
EA2C:        EA2C  READREQ2   EQU   *
EA2C:20 98 ED                 JSR   BLK2SECT         ;COMPUTE TRK/SECTOR THIS BLOCK
EA2F:              *
EA2F:20 0E EB                 JSR   SECTORIO         ;READ IT PLEASE
EA32:B0 1E   EA52             BCS   READERR          ;=>WE LOSE.
EA34:E6 D5                    INC   SECTOR           ;BUMP TO NEXT
EA36:E6 D5                    INC   SECTOR           ; LOGICAL SECTOR
EA38:E6 9C                    INC   BUF+1            ;BUMP SECTOR BUFFER
EA3A:20 0E EB                 JSR   SECTORIO         ;READ IT TOO
EA3D:B0 13   EA52             BCS   READERR          ;=>WE LOSE.
EA3F:A0 01                    LDY   #1
EA41:B1 C8                    LDA   (D.BYTRD),Y      ;BUMP COUNT OF
EA43:18                       CLC
EA44:69 02                    ADC   #2
EA46:91 C8                    STA   (D.BYTRD),Y      ; BYTES READ
EA48:              *
EA48:              * MORE BLOCKS TO GO?
EA48:              *
EA48:20 DB ED                 JSR   MOREBLKS         ;SETUP FOR NEXT BLOCK
EA4B:D0 DF   EA2C             BNE   READREQ2         ;=>MORE TO READ...
EA4D:        EA4D  READOK     EQU   *
EA4D:A9 00                    LDA   #0               ;GOOD RETURN
EA4F:4C E9 EA                 JMP   EXIT             ;TELL HAPPY USER
EA52:              *
EA52:        EA52  READERR    EQU   *
EA52:4C E9 EA                 JMP   EXIT             ;RETURN ERROR CODE
EA55:                         CHN   DISK3.WRT.SRC
```

```
EA55:                ---------------------------------------
EA55:                * --- WRITE ---
EA55:                ---------------------------------------
EA55:                *
EA55:        EA55    WRITEREQ   EQU   *
EA55:20 98 ED                   JSR   BLK2SECT         ;COMPUTE TRK/SECTOR THIS BLOCK
EA58:AD DF FF                   LDA   E.REG            ;SET 2 MHZ
EA5B:29 7F                      AND   #$7F
EA5D:8D DF FF                   STA   E.REG
EA60:20 C4 F2                   JSR   PRENIB           ;PRENIBBLIZE FOR WRITE
EA63:20 0E EB                   JSR   SECTORIO         ;WRITE IT OUT...
EA66:B0 20   EA88               BCS   WRITERR          ;=>SOMETHING'S WRONG
EA68:                *
EA68:E6 D5                      INC   SECTOR           ;BUMP TO NEXT
EA6A:E6 D5                      INC   SECTOR           ; LOGICAL SECTOR
EA6C:E6 9C                      INC   BUF+1            ;BUMP SECTOR BUFFER ADDRESS
EA6E:AD DF FF                   LDA   E.REG            ;SET 2 MHZ
EA71:29 7F                      AND   #$7F
EA73:8D DF FF                   STA   E.REG
EA76:20 C4 F2                   JSR   PRENIB           ;PRENIBBLIZE FOR WRITE
EA79:20 0E EB                   JSR   SECTORIO         ;WRITE IT OUT
EA7C:B0 0A   EA88               BCS   WRITERR          ;=>SOMETHING'S WRONG
EA7E:                *
EA7E:                * MORE BYTES TO DO?
EA7E:                *
EA7E:20 DB ED                   JSR   MOREBLKS         ;SETUP FOR NEXT
EA81:D0 D2   EA55               BNE   WRITEREQ         ;=>MORE TO DO
EA83:A9 00                      LDA   #0               ;GOOD RETURN
EA85:4C E9 EA                   JMP   EXIT
EA88:                *
EA88:        EA88    WRITERR    EQU   *
EA88:4C E9 EA                   JMP   EXIT             ;RETURN ERROR CODE
```

```
EA8B:                ---------------------------------------
EA8B:                *  --- STATUS ---
EA8B:                ---------------------------------------
EA8B:                *
EA8B:        EA8B    STATUS     EQU   *
EA8B:A2 60                      LDX   #$60             ;DUMMY SLOT
EA8D:BD 8D C0                   LDA   Q6H,X            ;SENSE WRITE PROTECT
EA90:BD 8E C0                   LDA   Q7L,X
EA93:0A                         ASL   A                ;PRESERVE IT IN CARRY
EA94:BD 8C C0                   LDA   Q6L,X            ;BACK TO READ MODE
EA97:A9 00                      LDA   #0               ;NOW MOVE BIT TO
EA99:2A                         ROL   A                ; PROPER POSITION
EA9A:2A                         ROL   A                ; ($02)
EA9B:A0 00                      LDY   #0
EA9D:91 C3                      STA   (D.STATBUF),Y    ;RETURN IT
EA9F:A9 00                      LDA   #0               ;GOOD RETURN
EAA1:4C E9 EA                   JMP   EXIT             ;DONE
```

```
EAA4:              ----------------------------------------
EAA4:              * --- INIT ---
EAA4:              ----------------------------------------
EAA4:              *
EAA4:        EAA4  INIT       EQU   *
EAA4:AD F4 ED                 LDA   INITFLAG         ;INIT'ED YET?
EAA7:30 3B   EAE4             BMI   GOODINIT         ;=>YES, DONE
EAA9:              *
EAA9:A9 60                    LDA   #$60             ;SETUP SLOT FOR
EAAB:85 81                    STA   IBSLOT           ; CORE ROUTINES
EAAD:A9 FF                    LDA   #$FF             ;PREVENT SECOND
EAAF:8D F4 ED                 STA   INITFLAG         ; INIT
EAB2:A9 00                    LDA   #0               ;CLEAR STUFF OUT
EAB4:8D F0 ED                 STA   PREVUNIT         ;SOSBOOT JUST USED .D1
EAB7:A0 04                    LDY   #4
EAB9:        EAB9  CLRDRVS    EQU   *
EAB9:A9 00                    LDA   #0
EABB:99 F7 ED                 STA   DRIVESEL-1,Y     ;NOBODY SELECTED
EABE:99 FB ED                 STA   UPTIME-1,Y       ;ALL OFF
EAC1:99 FF ED                 STA   DRVTRACK-1,Y
EAC4:88                       DEY
EAC5:D0 F2   EAB9             BNE   CLRDRVS
EAC7:        0001             DO    1-TEST           ;ONLY IF NOT TESTING
EAC7:              *
EAC7:              * SET UP .D1 SINCE LOADER'S USING IT:
EAC7:              *
EAC7:AD DF FF                 LDA   E.REG            ;SET 1MHZ FOR THE
EACA:09 80                    ORA   #$80             ; STATEMACHINE I/O
EACC:8D DF FF                 STA   E.REG
EACF:20 DC EC                 JSR   CHKDRV           ;IS .D1 MOTOR SPINNING?
EAD2:F0 05   EAD9             BEQ   INIT2            ;=>NO, MOTOR'S OFF
EAD4:A9 08                    LDA   #T200MS          ;UPTIME GOOD FOR READS
EAD6:8D FC ED                 STA   UPTIME+0
EAD9:        EAD9  INIT2      EQU   *
EAD9:A9 01                    LDA   #1
EADB:8D F8 ED                 STA   DRIVESEL+0       ;.D1 IS THE CURRENT DRIVE
EADE:AD 8C 03                 LDA   $0300+CURTRK     ;RETRIEVE CURRENT TRACK
EAE1:8D 00 EE                 STA   DRVTRACK+0       ;REMEMBER IT
EAE4:                         FIN
EAE4:              *
EAE4:              * SET UP JMP TABLE FOR CORRECT ROM:
EAE4:              *
EAE4:        0000             DO    REV0ROM          ;ONLY IF SUPPORTING IT!
 S                            LDA   $F1B9            ;LOOK FOR START OF RDADR
 S                            CMP   #$A0             ;IS IT RDADR (REV1)?
 S                            BEQ   INITREV1         ;=>YES
 S                            CMP   #$60             ;IS IT END OF READ (REV0)?
 S                            BNE   INITERR          ;=>NEITHER!
 S                            LDY   #0               ;REV=0
 S                            BEQ   INITVECT         ;(ALWAYS TAKEN)
 S                 INITREV1   EQU   *
 S                            LDY   #VSIZE
 S                 INITVECT   EQU   *
 S                            STY   ROMREV           ;SET ROM REVISION INDICATOR
 S                            LDX   #VSIZE
 S                 MOVEVECT   EQU   *
 S                            LDA   REV0,Y           ;GET A BYTE
 S                            STA   JMPTAB,Y         ;MOVE IT
 S                            INY
 S                            DEX
 S                            BNE   MOVEVECT
EAE4:                         FIN
EAE4:        EAE4  GOODINIT   EQU   *
EAE4:A9 00                    LDA   #0               ;RETCODE=GOOD, IF YOU CARE
EAE6:18                       CLC                    ;SAY 'GOOD INIT'
EAE7:90 00   EAE9             BCC   EXIT             ;(ALWAYS TAKEN)
EAE9:        0000             DO    REV0ROM
 S                 INITERR    EQU   *
 S                            SEC   ;SAY             'BAD INIT'
 S                 *          FALL  THRU             TO EXIT
EAE9:                         FIN
```

```
EAE9:             ---------------------------------------
EAE9:             * -- EXIT PATH --
EAE9:             ---------------------------------------
EAE9:             *
EAE9:        EAE9 EXIT       EQU   *
EAE9:48                      PHA                     ;SAVE RETURN CODE
EAEA:             *
EAEA:             * UPDATE UPTIME BY 50 MS (3 SECTOR-TIMES)
EAEA:             *  TO ACCOUNT FOR READ/WRITE TIME:
EAEA:             *
EAEA:A5 C0                   LDA   D.COMMAND         ;GET COMMAND
EAEC:C9 02                   CMP   #2                ;SENSE OR INIT?
EAEE:B0 05   EAF5            BCS   EXIT2             ;=>YES, NO TIME USED UP
EAF0:A9 02                   LDA   #2                ;TIME=50 MS (2 UNITS)
EAF2:20 0A ED                JSR   ADDTIME           ;BUMP UPTIME(S)
EAF5:             *
EAF5:             * RESTORE CALLER ENVIRONMENT:
EAF5:             *
EAF5:        EAF5 EXIT2      EQU   *
EAF5:AD DF FF                LDA   E.REG             ;GET CURRENT STATE
EAF8:29 20                   AND   #$20              ; OF THE SCREEN
EAFA:0D F2 ED                ORA   ESAVE             ;MERGE WITH CALLER STATE
EAFD:8D DF FF                STA   E.REG
EB00:20 E8 ED                JSR   FIXIRQ            ;RE-ENABLE IRQ IF OK
EB03:AD E8 C0                LDA   MOTOROFF          ;START MOTOR-OFF TIMEOUT
EB06:68                      PLA                     ;RESTORE RETURN CODE
EB07:        0000            DO    TEST              ;IF TEST, NO SYSERR
 S                           RTS
EB07:                        ELSE
EB07:D0 02   EB0B            BNE   GOERR             ;=>ERROR RETURN VIA SYSERR
EB09:18                      CLC
EB0A:60                      RTS                     ;GOOD RETURN W/CARRY CLEAR
EB0B:        EB0B GOERR      EQU   *
EB0B:20 00 00                JSR   SYSERR            ;RETURN VIA SYSERR
EB0E:                        FIN
EB0E:                        CHN   DISK3.SIO.SRC
```

```
EB0E:               ---------------------------------------
EB0E:               * NAME    : SECTORIO
EB0E:               * FUNCTION: READ OR WRITE A SECTOR
EB0E:               * INPUT   : IBSTRK, IBSECT, MONTIME,
EB0E:               * RETURNS : CARRY CLEAR IF OK (AC=00)
EB0E:               *         : CARRY SET   IF ERROR (AC=ERRCODE)
EB0E:               *         : SEEKWAIT  ALL SETUP
EB0E:               * DESTROYS: ALL REGISTERS
EB0E:               ---------------------------------------
EB0E:               *
EB0E:        EB0E   SECTORIO   EQU   *
EB0E:A9 01                     LDA   #R.RECAL         ;SETUP THE
EB10:               * R.RECAL MUST BE NON-ZERO!! (SEE BELOW)
EB10:85 D8                     STA   RECALCNT         ; RECAL TRIES
EB12:EA                        NOP                    ; PAD ONE BYTE
EB13:8D 00 00                  STA   E1908            ; A-REG MUST BE NON-ZERO !!!
EB16:               * E1908 = NON-ZERO LOCKOUT MOUSE
EB16:               *
EB16:A4 C1                     LDY   D.UNITNUM        ;ARE WE ON-TRACK?
EB18:A5 D4                     LDA   TRACK
EB1A:D9 00 EE                  CMP   DRVTRACK,Y
EB1D:F0 1B   EB3A              BEQ   SOUGHT           ;=>IF SO, FORGET SEEK & DELAY!
EB1F:               *
EB1F:               * WAIT BEFORE STEPPING:
EB1F:               *
EB1F:A5 DA                     LDA   SEEKWAIT         ;SEEK DELAY NEEDED?
EB21:F0 12   EB35              BEQ   GOSEEK           ;=>NAW...
EB23:A9 00                     LDA   #0
EB25:85 DA                     STA   SEEKWAIT         ;CLEAR THE FLAG
EB27:A9 04                     LDA   #4               ;ADD SEEKDELAY TO
EB29:20 0A ED                  JSR   ADDTIME          ; THE TOTAL UPTIME(S)
EB2C:A8                        TAY                    ;4*25 MS DELAY
EB2D:        EB2D   SEEKDEL    EQU   *
EB2D:A9 00                     LDA   #0
EB2F:20 56 F4                  JSR   MSWAIT
EB32:88                        DEY
EB33:D0 F8   EB2D              BNE   SEEKDEL
EB35:               *
EB35:               * ISSUE THE SEEK:
EB35:               *
EB35:        EB35   GOSEEK     EQU   *
EB35:A5 D4                     LDA   TRACK            ;GET DESTINATION TRACK
EB37:20 60 ED                  JSR   MYSEEK           ;=>..AND YOU SHALL FIND...
EB3A:               *
EB3A:        EB3A   SOUGHT     EQU   *
EB3A:A5 DB                     LDA   IRQMASK          ;SET IRQ MASK FOR
EB3C:85 8B                     STA   IMASK            ; CORE ROUTINES
EB3E:A9 06                     LDA   #R.IRQ           ;SETUP IRQ RETRIES
EB40:85 8F                     STA   INTRTRY
EB42:A9 04                     LDA   #R.IOERR         ; AND ERROR RETRIES
EB44:85 D7                     STA   RETRYCNT
EB46:               *
EB46:               * DELAY FOR ANY REMAINING MOTOR-UP TIME:
EB46:               *
EB46:        EB46   MDELAY     EQU   *
EB46:A5 9A                     LDA   MONTIMEH         ;ANY TIME REMAINING?
```

```
EB48:10 0D   EB57              BPL   FINDIT           ;=>NO, WE'RE UP TO SPEED.
EB4A:A9 01                     LDA   #1               ;YES, SO BUMP A SLICE OF
EB4C:20 0A ED                  JSR   ADDTIME          ; UPTIME WHILE WE WAIT
EB4F:A9 00                     LDA   #0
EB51:20 56 F4                  JSR   MSWAIT
EB54:4C 46 EB                  JMP   MDELAY           ;=>GO TILL ENOUGH
EB57:               *
EB57:               * FIND THE DESIRED SECTOR:
EB57:               *
EB57:               * NOTE: FINDSECT RETURNS WITH
EB57:               *       IRQ INHIBITED!
EB57:               *
EB57:        EB57   FINDIT     EQU   *
EB57:08                        PHP                    ;INHIBIT IRQ WHILE
EB58:78                        SEI                    ; MESSING WITH VBL FLAGS
EB59:AD EE FF                  LDA   E.IER            ;DISABLE VBL IRQ
EB5C:29 18                     AND   #$18             ; DURING SECTOR I/O
EB5E:8D EE FF                  STA   E.IER
EB61:09 80                     ORA   #$80             ;FOR 'SET' LATER
EB63:8D F3 ED                  STA   VBLSAVE
EB66:28                        PLP                    ;RESTORE IRQ STATUS
EB67:20 D5 EB                  JSR   FINDSECT         ;FIND ME PLEASE
EB6A:B0 3A   EBA6              BCS   TRYRECAL         ;=>NO? RECAL OR GIVE UP!
EB6C:A2 60                     LDX   #$60             ;SET UP SLOT FOR CORE RTNS
EB6E:A5 C0                     LDA   D.COMMAND        ;WHAT'S YOUR PLEASURE?
EB70:D0 1E   EB90              BNE   SIOWRITE         ;=>WRITE
EB72:               *
EB72:               ---------------------------------------
EB72:               * READ A SECTOR:
EB72:               *
EB72:20 48 F1                  JSR   READ             ;READ THAT SECTOR
EB75:20 E8 ED                  JSR   FIXIRQ           ;ENABLE IRQ IF OK
EB78:AD F3 ED                  LDA   VBLSAVE          ;ALLOW VBL DURING
EB7B:8D EE FF                  STA   E.IER            ; POSTNIB
EB7E:B0 20   EBA0              BCS   BADIO            ;=>I/O ERR OR IRQ
EB80:AD DF FF                  LDA   E.REG            ;SET 2MHZ FOR POSTNIB
EB83:29 7F                     AND   #$7F
EB85:8D DF FF                  STA   E.REG
EB88:20 0F F3                  JSR   POSTNIB          ;POSTNIB/CHECKSUM
EB8B:B0 15   EBA2              BCS   IORETRY          ;=>I/O ERR:BAD CHKSUM
EB8D:4C CC EB                  JMP   SIOGOOD          ;=>GOOD READ
EB90:               *
EB90:               ---------------------------------------
EB90:               * WRITE A SECTOR:
EB90:               *
EB90:        EB90   SIOWRITE   EQU   *
EB90:20 16 F2                  JSR   WRITE            ;WRITE THE DATA
EB93:20 E8 ED                  JSR   FIXIRQ           ;RE-ENABLE IRQ IF OK
EB96:AD F3 ED                  LDA   VBLSAVE          ;RESTORE
EB99:8D EE FF                  STA   E.IER            ; VBL IRQ
EB9C:90 2E   EBCC              BCC   SIOGOOD          ;=>GOOD WRITE
EB9E:50 27   EBC7              BVC   SIOWPROT         ;=>WRITE PROTECTED
EBA0:               *
EBA0:               ---------------------------------------
EBA0:               * IT DIDN'T GO WELL FOR US:
EBA0:               *
```

```
EBA0:        EBA0  BADIO      EQU   *
EBA0:        0001             DO    1-REV0ROM        ;FOR REV1
EBA0:70 B5   EB57             BVS   FINDIT           ;=>IRQ. JUST RETRY IT.
EBA2:                         ELSE  ;FOR             REV0
 S                 *
 S                 *          THE   REV1             ROM TAKES CARE OF THE
 S                 *                IRQ              RETRY COUNT, BUT REV0 DOESN'T:
 S                 *
 S                            BVC   IORETRY          ;=>I/O ERROR. RETRY IT
 S                            LDA   ROMREV           ;WHICH ROM?
 S                            BNE   FINDIT           ;=>REV1. HE DOES IT.
 S                            LDA   INTRTRY          ;REV0. OUT OF RETRIES?
 S                            BPL   BADIO2           ;=>NO.
 S                            STA   IMASK            ;SET HI BIT FOR IRQ MASK
 S                 BADIO2     EQU   *
 S                            DEC   INTRTRY          ;ONE LESS RETRY
 S                            JMP   FINDIT           ;=>RETRY AFTER IRQ
EBA2:                         FIN
EBA2:              *
EBA2:              * RETRY AFTER AN I/O ERROR:
EBA2:              *
EBA2:        EBA2  IORETRY    EQU   *
EBA2:C6 D7                    DEC   RETRYCNT         ;ANY RETRIES LEFT?
EBA4:D0 B1   EB57             BNE   FINDIT           ;=>YEAH, RETRY AFTER ERROR
EBA6:              *
EBA6:              * RETRIES EXHAUSTED. RECALIBRATE:
EBA6:              *
EBA6:        EBA6  TRYRECAL   EQU   *
EBA6:AD F3 ED                 LDA   VBLSAVE          ;ALLOW VBL IF RECAL
EBA9:8D EE FF                 STA   E.IER            ; OR UNRECOVERABLE ERROR
EBAC:C6 D8                    DEC   RECALCNT         ;HAVE WE RECALIBRATED YET?
EBAE:30 12   EBC2             BMI   SIOERR           ;=>YUP. WE'RE DEAD.
EBB0:20 26 ED                 JSR   RECAL            ;NO, TRY OUR LUCK
EBB3:A4 C1                    LDY   D.UNITNUM        ;ARE WE ON-TRACK?
EBB5:A5 D4                    LDA   TRACK
EBB7:D9 00 EE                 CMP   DRVTRACK,Y
EBBA:D0 03   EBBF             BNE   NOTSAME
EBBC:4C 3A EB                 JMP   SOUGHT           ;=>IF SO, FORGET RESEEK
EBBF:        EBBF  NOTSAME    EQU   *
EBBF:4C 35 EB                 JMP   GOSEEK           ;TRY AGAIN ON TARGET TRACK
EBC2:              *
EBC2:              ---------------------------------------
EBC2:        EBC2  SIOERR     EQU   *
EBC2:A9 00                    LDA   #XIOERROR        ;RETURN CODE
EBC4:38                       SEC                    ;INDICATE HARD ERROR
EBC5:B0 08   EBCF             BCS   SIORET
EBC7:        EBC7  SIOWPROT   EQU   *
EBC7:A9 00                    LDA   #XNOWRITE        ;RETURN CODE
EBC9:38                       SEC                    ;INDICATE HARD ERROR
EBCA:B0 03   EBCF             BCS   SIORET
EBCC:        EBCC  SIOGOOD    EQU   *
EBCC:A9 00                    LDA   #0
EBCE:18                       CLC                    ;INDICATE GOOD COMPLETION
EBCF:A2 00         SIORET     LDX   #0               ; SAY OK TO MOUSE
EBD1:8E 00 00                 STX   E1908            ; WITH THIS GLOBAL $1908
EBD4:60                       RTS
```

```
EBD5:              ---------------------------------------
EBD5:              * NAME    : FINDSECT
EBD5:              * FUNCTION: LOCATE A DESIRED SECTOR
EBD5:              * INPUT   : IBTRK, IBSECT SETUP
EBD5:              * RETURNS : CARRY CLEAR IF OK,
EBD5:              *         : CARRY SET   IF ERROR.
EBD5:              * DESTROYS: ALL REGISTERS & 'TEMP'
EBD5:              * NOTE    : RETURNS WITH IRQ DISABLED IF NO ERROR!
EBD5:              ---------------------------------------
EBD5:              *
EBD5:        EBD5  FINDSECT   EQU   *
EBD5:A9 30                    LDA   #R.FIND*16       ;SETUP NUMBER OF REVS
EBD7:85 D6                    STA   RETRYADR         ; ALLOWED TO FIND SECTOR
EBD9:46 DC                    LSR   TEMP             ;COMPUTE LATENCY FIRST TIME THRU
EBDB:        EBDB  FINDSEC2   EQU   *
EBDB:A2 60                    LDX   #$60             ;FAKE SLOT FOR CORE ROUTINES
EBDD:20 B9 F1                 JSR   RDADR            ;GET NEXT ADDRESS FIELD
EBE0:B0 1D   EBFF             BCS   RDADERR          ;=>UGH! AN ERROR!
EBE2:              *
EBE2:              * MAKE SURE WE'RE ON THE CORRECT TRACK:
EBE2:              *
EBE2:A5 D4                    LDA   TRACK            ;IS IT
EBE4:C5 99                    CMP   CSSTV+2          ; CORRECT TRACK?
EBE6:D0 2C   EC14             BNE   FINDERR          ;=>NO?!? IT'S USELESS!
EBE8:A5 D5                    LDA   SECTOR           ;IS IT
EBEA:C5 98                    CMP   CSSTV+1          ; DESIRED SECTOR?
EBEC:F0 20   EC0E             BEQ   FINDGOOD         ;=>YEAH. GOT IT!
EBEE:              *
EBEE:              * COMPUTE LATENCY. EACH TWO-SECTOR
EBEE:              *  DISTANCE IS 25 MS OF UPTIME.
EBEE:              *
EBEE:A5 DC                    LDA   TEMP             ;LATENCY ALREADY COMPUTED?
EBF0:30 0D   EBFF             BMI   RDADERR          ;=>YES.
EBF2:A5 D5                    LDA   SECTOR           ;HOW FAR AWAY IS OUR
EBF4:38                       SEC                    ; DESIRED SECTOR?
EBF5:66 DC                    ROR   TEMP             ;PREVENT RECOMPUTATION
EBF7:E5 98                    SBC   CSSTV+1
EBF9:29 0F                    AND   #$0F
EBFB:4A                       LSR   A                ;EACH 2-SECTORS IS 25 MS
EBFC:20 0A ED                 JSR   ADDTIME
EBFF:              *
EBFF:              * KEEP LOOKING TILL WE FIND IT:
EBFF:              *
EBFF:        EBFF  RDADERR    EQU   *
EBFF:20 E8 ED                 JSR   FIXIRQ           ;ENABLE IRQ IF APPROPRIATE
EC02:C6 D6                    DEC   RETRYADR         ;ANY RETRIES LEFT?
EC04:F0 0E   EC14             BEQ   FINDERR          ;=>NO, WE CAN'T FIND IT.
EC06:              *
EC06:              * COMPENSATE FOR A BUG IN RDADR: IF WE TRY
EC06:              *  TO CALL RDADR AGAIN BEFORE THE DATA MARK
EC06:              *  GOES BY, THEN RDADR WILL ACCIDENTALLY CALL
EC06:              *  THAT AN ERROR. WE CAN AVOID THIS 'FAKE'
EC06:              *  ERROR BY DELAYING PAST THE DATA MARK.
EC06:A0 C8                    LDY   #200             ;1 MS IS PLENTY
EC08:        EC08  ADRDELAY   EQU   *
EC08:88                       DEY
```

```
EC09:D0 FD   EC08              BNE   ADRDELAY
EC0B:4C DB EB                  JMP   FINDSEC2         ;=>NOW TRY LOOKING AGAIN
EC0E:                *
EC0E:                ----------------------------------------
EC0E:        EC0E    FINDGOOD  EQU   *
EC0E:A9 00                     LDA   #0               ;CLEAR VOLNUM OUT OF
EC10:85 9A                     STA   MONTIMEH         ; MOTORTIME!
EC12:18                        CLC                    ;INDICATE NO ERROR
EC13:60                        RTS
EC14:                *
EC14:        EC14    FINDERR   EQU   *
EC14:20 E8 ED                  JSR   FIXIRQ           ;ENABLE IRQ IF APPROPRIATE
EC17:A9 00                     LDA   #0               ;CLEAR VOLNUM OUT OF
EC19:85 9A                     STA   MONTIMEH         ; MOTORTIME!
EC1B:38                        SEC                    ;INDICATE THE ERROR
EC1C:60                        RTS
EC1D:                          CHN   DISK3.USEL.SRC
```

```
EC1D:                   ----------------------------------------
EC1D:                   * NAME    : UNITSEL
EC1D:                   * FUNCTION: SELECT & START A DRIVE,
EC1D:                   *           SET UP MOTOR & SEEK DELAYS
EC1D:                   * INPUT   : NONE
EC1D:                   * OUTPUT  : MONTIME,SEEKTIME
EC1D:                   * DESTROYS: ALL REGISTERS
EC1D:                   ----------------------------------------
EC1D:                   *
EC1D:         EC1D      UNITSEL    EQU   *
EC1D:A4 C1                         LDY   D.UNITNUM        ;GET DRIVENUM
EC1F:A9 00                         LDA   #0               ;ASSUME NO SEEKWAIT
EC21:85 DA                         STA   SEEKWAIT         ; WILL BE NEEDED
EC23:85 99                         STA   MONTIMEL         ;CLEAR MONTIME
EC25:85 9A                         STA   MONTIMEH
EC27:                   *
EC27:                   * SEE IF MOTOR(S) STILL SPINNING:
EC27:                   *
EC27:20 DC EC                      JSR   CHKDRV           ;MOTOR(S) POWERED UP?
EC2A:D0 11   EC3D                  BNE   SPINNING         ;=>YES. WHO IS IT?
EC2C:                   *
EC2C:                   * NO MOTOR(S) SPINNING. DESELECT
EC2C:                   *  ALL MOTORS AND START AFRESH:
EC2C:                   *
EC2C:AE D5 C0                      LDX   MD.INT           ;DESELECT ALL
EC2F:A9 00                         LDA   #0               ;SHOW INTERNAL AS
EC31:8D F8 ED                      STA   DRIVESEL+0       ; NOT SELECTED
EC34:8D FC ED                      STA   UPTIME+0         ;INDICATE DRIVE IS FULLY STOPPED
EC37:20 C8 EC                      JSR   EXTDESEL         ;DESELECT ALL EXTERNALS TOO
EC3A:4C 6B EC                      JMP   SETTIME          ;GO SETUP MOTOR DELAY
EC3D:                   ----------------------------------------
EC3D:                   * MOTOR(S) SPINNING: OURS?
EC3D:                   *
EC3D:         EC3D      SPINNING   EQU   *
EC3D:B9 F8 ED                      LDA   DRIVESEL,Y       ;HAD WE BEEN SELECTED?
EC40:D0 19   EC5B                  BNE   GOFORIT          ;=>YES, GO FOR IT RIGHT AWAY.
EC42:                   *
EC42:                   * WE AREN'T SPINNING. SHUTDOWN ANOTHER
EC42:                   *  DRIVE, IF NECESSARY, TO GET GOING:
EC42:                   *
EC42:C0 00                         CPY   #0               ;ARE WE THE INTERNAL DRIVE?
EC44:F0 25   EC6B                  BEQ   SETTIME          ;=>YES, LEAVE EXT MOTOR ALONE
EC46:                   *
EC46:                   * WE'RE AN EXTERNAL DRIVE. STOP ALL EXTERNAL MOTORS
EC46:                   *  UNCONDITIONALLY, BUT LEAVE THE INTERNAL MOTOR ALONE.
EC46:                   * IF WE *DID* HAVE TO STOP ANOTHER EXTERNAL, THEN
EC46:                   *  MAKE SURE WE SET THE CORRECT PRE-SEEK DELAY!
EC46:                   *
EC46:A9 00                         LDA   #0               ;SEE IF ANOTHER EXTERNAL
EC48:0D FB ED                      ORA   DRIVESEL+3       ; HAD BEEN
EC4B:0D FA ED                      ORA   DRIVESEL+2       ;  SELECTED
EC4E:0D F9 ED                      ORA   DRIVESEL+1       ;   BEFORE...
EC51:F0 18   EC6B                  BEQ   SETTIME          ;=>NO, SEEK DELAY IS UNNECESSARY
EC53:E6 DA                         INC   SEEKWAIT         ;YES, DELAY BEFORE STEPPING
EC55:20 C8 EC                      JSR   EXTDESEL         ;DESELECT ALL EXTERNALS
EC58:4C 6B EC                      JMP   SETTIME          ;=>GO SETUP MOTOR DELAY
```

```
EC5B:               ----------------------------------------
EC5B:               * OUR DRIVE IS SPINNING. GO FOR IT!
EC5B:               * DEPENDING OF HOW LONG THE MOTOR'S BEEN ON,
EC5B:               *  THIS COMMAND MAY REQUIRE A MOTOR DELAY.
EC5B:               *
EC5B:        EC5B   GOFORIT    EQU   *
EC5B:A6 C0                     LDX   D.COMMAND        ;GET CURRENT COMMAND
EC5D:BD F5 ED                  LDA   MTIMES,X         ;GET REQUIRED UPTIME FOR IT
EC60:38                        SEC
EC61:F9 FC ED                  SBC   UPTIME,Y         ;DRIVE RUNNING LONG ENOUGH?
EC64:B0 0F   EC75              BCS   SELECT           ;=>NO, AC NOW HAS DELTA-T
EC66:A9 00                     LDA   #0               ;OTHERWISE, WAIT=0
EC68:4C 75 EC                  JMP   SELECT           ;SET MONTIME & SELECT DRIVE
EC6B:               ----------------------------------------
EC6B:               *
EC6B:               * ALL MOTORS WERE OFF. CHOOSE THE
EC6B:               *  APPROPRIATE MOTOR-ON TIME:
EC6B:               *
EC6B:        EC6B   SETTIME    EQU   *
EC6B:A9 00                     LDA   #0               ;INDICATE THAT
EC6D:99 FC ED                  STA   UPTIME,Y         ; THE DRIVE WAS OFF
EC70:A6 C0                     LDX   D.COMMAND        ;GET CURRENT COMMAND
EC72:BD F5 ED                  LDA   MTIMES,X         ;GET CORRECT DELAY TIME
EC75:               ----------------------------------------
EC75:               *
EC75:               * SELECT THE DRIVE & START IT:
EC75:               *
EC75:        EC75   SELECT     EQU   *
EC75:85 9A                     STA   MONTIMEH         ;NEGATE IT BECAUSE
EC77:A9 00                     LDA   #0               ; IT GETS INCREMENTED
EC79:38                        SEC                    ;  INSTEAD OF
EC7A:E5 9A                     SBC   MONTIMEH         ;   DECREMENTED
EC7C:85 9A                     STA   MONTIMEH         ;STUFF MOTOR DELAY
EC7E:C0 01                     CPY   #1               ;ARE WE THE INTERNAL DRIVE?
EC80:B0 09   EC8B              BCS   SELEXT           ;=>NO, AN EXTERNAL
EC82:AD EA C0                  LDA   IS.INT           ;I/O SELECT INTERNAL
EC85:AD D4 C0                  LDA   MS.INT           ;MOTOR SELECT INTERNAL
EC88:4C AC EC                  JMP   UNITRET          ;=>ALL DONE!
EC8B:               *
EC8B:        EC8B   SELEXT     EQU   *
EC8B:AD EB C0                  LDA   IS.EXT           ;I/O SELECT EXTERNAL
EC8E:C0 02                     CPY   #2               ;ARE WE 2, 3, OR 4 ?
EC90:B0 09   EC9B              BCS   NOTD2            ;=>DEFINITELY 3 OR 4
EC92:AD D2 C0                  LDA   MD.EXT1          ;MOTOR SELECT
EC95:AD D1 C0                  LDA   MS.EXT2          ; ONLY .D2
EC98:4C AC EC                  JMP   UNITRET          ;=>ALL DONE!
EC9B:               *
EC9B:        EC9B   NOTD2      EQU   *
EC9B:D0 09   ECA6              BNE   ISD4             ;=>DEFINITELY NOT 3
EC9D:AD D3 C0                  LDA   MS.EXT1          ;MOTOR SELECT
ECA0:AD D0 C0                  LDA   MD.EXT2          ; ONLY .D3
ECA3:4C AC EC                  JMP   UNITRET          ;=>ALL DONE!
ECA6:               *
ECA6:        ECA6   ISD4       EQU   *
ECA6:AD D3 C0                  LDA   MS.EXT1          ;MOTOR SELECT
ECA9:AD D1 C0                  LDA   MS.EXT2          ; ONLY .D4
```

```
ECAC:              *
ECAC:              *
ECAC:        ECAC  UNITRET    EQU   *
ECAC:AD E9 C0                 LDA   MOTORON          ;PROVIDE MOTOR POWER
ECAF:A9 01                    LDA   #1               ;SAY WE'VE SELECTED
ECB1:99 F8 ED                 STA   DRIVESEL,Y       ; THIS DRIVE
ECB4:              *
ECB4:              * IF WE HAVE MOTORTIME TO BURN,
ECB4:              *  THEN DELAY 50 MS. THIS ENSURES
ECB4:              *  A GOOD SOLID CHKDRV AFTER
ECB4:              *  TURNING ON THE MOTOR.
ECB4:              *
ECB4:A5 9A                    LDA   MONTIMEH         ;ANY MOTORTIME?
ECB6:10 0F   ECC7             BPL   UNITRTS          ;=>NO, WE GO FOR IT.
ECB8:A0 05                    LDY   #5               ;5*10 MS
ECBA:        ECBA  UNITDEL    EQU   *
ECBA:A9 64                    LDA   #100             ;100*100US IS 10MS
ECBC:20 56 F4                 JSR   MSWAIT
ECBF:88                       DEY
ECC0:D0 F8   ECBA             BNE   UNITDEL
ECC2:A9 02                    LDA   #2               ;INCLUDE THE 50MS
ECC4:20 0A ED                 JSR   ADDTIME          ; IN MOTOR UPTIME(S)
ECC7:        ECC7  UNITRTS    EQU   *
ECC7:60                       RTS



ECC8:              ----------------------------------------
ECC8:              * NAME    : EXTDESEL
ECC8:              * FUNCTION: DESELECT ALL EXTERNAL DRIVE MOTORS
ECC8:              * INPUT   : NONE
ECC8:              * DESTROYS: AC,X
ECC8:              ----------------------------------------
ECC8:              *
ECC8:        ECC8  EXTDESEL   EQU   *
ECC8:AD D2 C0                 LDA   MD.EXT1          ;DESELECT ALL EXTERNAL
ECCB:AD D0 C0                 LDA   MD.EXT2          ; DRIVE MOTORS
ECCE:A2 03                    LDX   #3               ;SHOW THAT THEY ARE
ECD0:A9 00                    LDA   #0               ; ARE ALL DEAD DUCKS
ECD2:9D F8 ED      EDS1       STA   DRIVESEL,X
ECD5:9D FC ED                 STA   UPTIME,X         ;DRIVE MOTORS ARE OFF
ECD8:CA                       DEX
ECD9:D0 F7   ECD2             BNE   EDS1
ECDB:60                       RTS
ECDC:                         CHN   DISK3.SUBS.SRC
```

```
ECDC:               ---------------------------------------
ECDC:               * NAME    : CHKDRV
ECDC:               * FUNCTION: CHECK IF MOTOR(S) RUNNING
ECDC:               * INPUT   : NONE
ECDC:               * RETURNS : 'BNE' IF RUNNING
ECDC:               *         : 'BEQ' IF NOT
ECDC:               * DESTROYS: AC,X
ECDC:               ---------------------------------------
ECDC:               * NOTES: DUE TO A FLOATING PIN, THERE
ECDC:               *  COULD BE A GLITCH WHICH CAUSES THE
ECDC:               *  SHIFTER TO 'FLASH' ONTO THE BUS
ECDC:               *  INSTEAD OF ALWAYS BEING TRISTATED.
ECDC:               *  THIS COULD CAUSE CHKDRV TO THINK
ECDC:               *  THAT THE MOTOR IS SPINNING WHEN IT
ECDC:               *  IS NOT. THUS WE WILL SAMPLE THE SHIFTER
ECDC:               *  FOR 40 US AT 6-US INTERVALS. IF, AFTER
ECDC:               *  THREE (3) CONSECUTIVE PASSES, ANY OF
ECDC:               *  THE PASSES SEES A 'LOCKED' SHIFTER,
ECDC:               *  THEN WE SAY THE DRIVE IS STOPPED.
ECDC:               *
ECDC:               *
ECDC:        ECDC   CHKDRV     EQU   *
ECDC:A2 03                     LDX   #3                ;CHECK SHIFTER SEVERAL TIMES
ECDE:        ECDE   CHKD1      EQU   *
ECDE:AD EC C0                  LDA   Q6L+$60           ;GET DATA
ECE1:CD EC C0                  CMP   Q6L+$60           ;HAS IT CHANGED?
ECE4:D0 1F   ED05              BNE   CHANGED           ;=>YES
ECE6:CD EC C0                  CMP   Q6L+$60           ;HAS IT CHANGED?
ECE9:D0 1A   ED05              BNE   CHANGED           ;=>YES
ECEB:CD EC C0                  CMP   Q6L+$60           ;HAS IT CHANGED?
ECEE:D0 15   ED05              BNE   CHANGED           ;=>YES
ECF0:CD EC C0                  CMP   Q6L+$60           ;HAS IT CHANGED?
ECF3:D0 10   ED05              BNE   CHANGED           ;=>YES
ECF5:CD EC C0                  CMP   Q6L+$60           ;HAS IT CHANGED?
ECF8:D0 0B   ED05              BNE   CHANGED           ;=>YES
ECFA:CD EC C0                  CMP   Q6L+$60           ;HAS IT CHANGED?
ECFD:D0 06   ED05              BNE   CHANGED           ;=>YES
ECFF:CD EC C0                  CMP   Q6L+$60           ;HAS IT CHANGED?
ED02:D0 01   ED05              BNE   CHANGED           ;=>YES
ED04:60                        RTS                     ;IF EVER LOCKED, IT'S STOPPED
ED05:               *
ED05:        ED05   CHANGED    EQU   *
ED05:CA                        DEX
ED06:D0 D6   ECDE              BNE   CHKD1             ;TRY SEVERAL TIMES
ED08:CA                        DEX                     ;SET CC=BNE
ED09:60                        RTS                     ;RETURN ZFLAG APPROPRIATELY
```

```
ED0A:               ----------------------------------------
ED0A:               * NAME    : ADDTIME
ED0A:               * FUNCTION: ADD TO MOTOR UPTIME(S)
ED0A:               * INPUT   : AC=NO. OF 25 MS INCREMENTS
ED0A:               * DESTROYS: Y
ED0A:               ----------------------------------------
ED0A:               *
ED0A:        ED0A   ADDTIME    EQU   *
ED0A:48                        PHA                    ;PRESERVE AC
ED0B:A0 04                     LDY   #4               ;TABLE INDEX/COUNT
ED0D:        ED0D   ADD2       EQU   *
ED0D:B9 F7 ED                  LDA   DRIVESEL-1,Y     ;IS IT SELECTED?
ED10:F0 0F   ED21              BEQ   ADD3             ;=>NOPE
ED12:68                        PLA
ED13:48                        PHA                    ;RECOVER DELTA-T
ED14:18                        CLC
ED15:79 FB ED                  ADC   UPTIME-1,Y       ;ADD TO MOTOR UPTIME
ED18:C9 29                     CMP   #T1SEC+2         ;IS IT AT MAX TIME?
ED1A:90 02   ED1E              BCC   ADD2A            ;=>NO, STORE NEW TIME
ED1C:A9 28                     LDA   #T1SEC+1         ;YES, SET TO >1 SEC
ED1E:        ED1E   ADD2A      EQU   *
ED1E:99 FB ED                  STA   UPTIME-1,Y
ED21:        ED21   ADD3       EQU   *
ED21:88                        DEY
ED22:D0 E9   ED0D              BNE   ADD2             ;=>DO ALL 4 DRIVES
ED24:               *
ED24:68                        PLA                    ;RESTORE AC
ED25:60                        RTS
```

```
ED26:              ---------------------------------------
ED26:              * NAME    : RECAL
ED26:              * FUNCTION: RECALIBRATE DRIVE HEAD
ED26:              * INPUT   : NONE
ED26:              * DESTROYS: ALL REGISTERS
ED26:              * NOTE    : A 'QUIET' RECALIBRATE IS DONE
ED26:              *         : USING TWO ITERATIONS. IF WE ARE
ED26:              *         : LOST, THEN SEEK 48-TRACKS
ED26:              *         : TOWARD TRACK ZERO. IF WE KNOW
ED26:              *         : WHAT TRACK WE'RE CURRENTLY
ED26:              *         : ON (+- 1/2 TRACK), THEN JUST
ED26:              *         : ADD A LITTLE EXTRA AND SEEK
ED26:              *         : TO TRACK ZERO. A 48-TRACK
ED26:              *         : SEEK WILL ALWAYS GET US BACK
ED26:              *         : ONTO THE MEDIA, EVEN IF WE
ED26:              *         : WERE "OFF THE CAM". FROM THAT
ED26:              *         : POINT, THE 2ND SEEK GETS US
ED26:              *         : BACK TO TRACK ZERO QUIETLY.
ED26:              ---------------------------------------
ED26:              *
ED26:        ED26  RECAL      EQU   *
ED26:A9 02                    LDA   #2               ;TWO ITERATIONS, PLEASE
ED28:        ED28  RECAL1     EQU   *
ED28:48                       PHA                    ;SAVE LOOPCOUNT
ED29:A2 60                    LDX   #$60             ;SETUP SLOT FOR CORE RTNS
ED2B:20 B9 F1                 JSR   RDADR            ;WHERE ARE WE?
ED2E:90 0A   ED3A             BCC   RECAL2           ;=>NOW WE KNOW
ED30:20 B9 F1                 JSR   RDADR            ;GIVE SECOND SHOT
ED33:90 05   ED3A             BCC   RECAL2           ;=>THAT GOT IT
ED35:A9 30                    LDA   #48              ;LOST? TRY 48-TRACK SEEK
ED37:4C 3F ED                 JMP   RECAL3
ED3A:        ED3A  RECAL2     EQU   *
ED3A:A5 99                    LDA   CSSTV+2          ;HERE'S WHERE WE ARE
ED3C:18                       CLC                    ;ADD SOME SO WE GET A
ED3D:69 03                    ADC   #3               ; HARDER SEEK TO ZERO
ED3F:        ED3F  RECAL3     EQU   *
ED3F:A4 C1                    LDY   D.UNITNUM        ;THIS IS NOW WHERE
ED41:99 00 EE                 STA   DRVTRACK,Y       ; WE ARE
ED44:20 E8 ED                 JSR   FIXIRQ           ;ENABLE IRQ IF OK
ED47:              *
ED47:A9 00                    LDA   #0               ;DESTINATION TRACK IS 00
ED49:85 9A                    STA   MONTIMEH         ;CLEAR MOTOR-UP TIME SO
ED4B:85 99                    STA   MONTIMEL         ; SEEK KNOWS HOW LONG RECAL TAKES
ED4D:20 60 ED                 JSR   MYSEEK           ;=>SLAM IT BACK!
ED50:68                       PLA                    ;HAVE WE DONE IT TWICE?
ED51:A8                       TAY
ED52:88                       DEY
ED53:98                       TYA
ED54:D0 D2   ED28             BNE   RECAL1           ;=>DO TWO ITERATIONS
ED56:60                       RTS
```

```
ED57:              ----------------------------------------
ED57:              * NAME    : SEEKDSK3
ED57:              * FUNCTION: SEEK CURRENT DRIVE
ED57:              * INPUT   : AC=DESTINATION TRACK
ED57:              * OUTPUT  : NONE
ED57:              * DESTROYS: ALL REGISTERS
ED57:              * NOTE    : MUST BE CALLED WHILE
ED57:              *         :  MOTOR IS RUNNING, IN
ED57:              *         :  1MHZ+ROM+IO MODE
ED57:              ----------------------------------------
ED57:        ED57  SEEKDSK3   EQU   *
ED57:AC F0 ED                 LDY   PREVUNIT         ;GET DRIVENUM
ED5A:84 C1                    STY   D.UNITNUM        ;SET IT UP
ED5C:20 60 ED                 JSR   MYSEEK           ;MOVE IT!
ED5F:60                       RTS
ED60:              ----------------------------------------
ED60:              * NAME    : MYSEEK
ED60:              * FUNCTION: SEEK TO DESIRED TRACK
ED60:              * INPUT   : AC=DESTINATION TRACK
ED60:              * DESTROYS: ALL REGISTERS
ED60:              ----------------------------------------
ED60:        ED60  MYSEEK     EQU   *
ED60:85 9E                    STA   TRKN             ;TEMP HOLD OF AC
ED62:A4 C1                    LDY   D.UNITNUM        ;GET DRIVENUM
ED64:B9 00 EE                 LDA   DRVTRACK,Y       ;SETUP CURRENT TRACK
ED67:0A                       ASL   A                ;SET IN HALFTRACKS FOR SEEK
ED68:85 8C                    STA   CURTRK           ; FOR SEEK ROUTINE
ED6A:A2 60                    LDX   #$60             ;SET UP SLOT FOR CORE RTNS
ED6C:A5 9A                    LDA   MONTIMEH         ;GET STARTING MOTOR TIME
ED6E:85 DC                    STA   TEMP
ED70:              *
ED70:              * NOTE: IRQ'S WHICH SUSPEND SEEK MAY CAUSE A
ED70:              *  SEEK FAILURE. WE WILL HAVE TO RECALIBRATE
ED70:              *  SINCE WE WON'T BE ON-TRACK. WE CAN NOT GET
ED70:              *  ON A HALFTRACK SINCE SEEK ALLOWS SETTLING
ED70:              *  TIME OF THE PHASE. BECAUSE VBL IS A SERIOUS
ED70:              *  OFFENDER, WE INHIBIT HIM.
ED70:              *
ED70:08                       PHP                    ;INHIBIT IRQ WHILE
ED71:78                       SEI                    ; MESSING WITH VBL FLAGS
ED72:AD EE FF                 LDA   E.IER
ED75:29 18                    AND   #$18
ED77:8D F3 ED                 STA   VBLSAVE
ED7A:8D EE FF                 STA   E.IER
ED7D:28                       PLP                    ;RESTORE IRQ STATUS
ED7E:A5 9E                    LDA   TRKN             ;RESTORE DESTINATION TRACK
ED80:99 00 EE                 STA   DRVTRACK,Y       ;DEST IS NOW CURRENT
ED83:0A                       ASL   A                ;MAKE IT IN HALFTRACKS
ED84:20 00 F4                 JSR   SEEK             ;GO MOVE THE HEAD...
ED87:AD F3 ED                 LDA   VBLSAVE          ;NOW ALLOW THAT
ED8A:09 80                    ORA   #$80             ; NASTY
ED8C:8D EE FF                 STA   E.IER            ;  VBL INTERRUPT
ED8F:              *
ED8F:              * COMPUTE THE TIME USED BY SEEK:
ED8F:              *
ED8F:A5 9A                    LDA   MONTIMEH         ;INCLUDE SEEKTIME IN
```

```
ED91:38            185            SEC
ED92:E5 DC         186            SBC   TEMP
ED94:20 0A ED      187            JSR   ADDTIME          ; TOTAL MOTOR UPTIME(S)
ED97:60            188            RTS
```

```
ED98:              ----------------------------------------
ED98:              * NAME    : BLK2SECT
ED98:              * FUNCTION: COMPUTE TRACK/SECTOR FOR A BLOCK
ED98:              *           AND ADJUST BUFFER ADDRESS
ED98:              * INPUT   : D.BLOCK, D.BUF
ED98:              * OUTPUT  : TRACK, SECTOR, D.BUF
ED98:              * DESTROYS: AC,Y
ED98:              ----------------------------------------
ED98:              *
ED98:        ED98  BLK2SECT   EQU   *
ED98:A5 D1                    LDA   BLKTEMP+1        ;GET HI BLK HALF
ED9A:6A                       ROR   A                ;MOVE LO BIT TO CARRY
ED9B:A5 D0                    LDA   BLKTEMP          ;GET LO HALF
ED9D:6A                       ROR   A                ;COMBINE WITH HI BIT
ED9E:4A                       LSR   A
ED9F:4A                       LSR   A                ;FINISH OFF DIVIDE-BY-8
EDA0:85 D4                    STA   TRACK            ;THAT'S THE TRACK
EDA2:A5 D0                    LDA   BLKTEMP          ;GET LO HALF AGAIN
EDA4:29 07                    AND   #7
EDA6:A8                       TAY
EDA7:B9 D3 ED                 LDA   SECTABLE,Y       ;GET START SECTOR
EDAA:85 D5                    STA   SECTOR
EDAC:              *
EDAC:              * ADJUST BUFFER ADDRESS SO THAT I/O
EDAC:              *  WON'T WRAPAROUND IN THE BANK:
EDAC:              * (THIS ALGORITHM RIPPED OFF FROM 1.0)
EDAC:              *
EDAC:A5 D3                    LDA   BUFTEMP+1        ;GET BUFFER HI ADDRESS
EDAE:AC D3 14                 LDY   $1400+BUFTEMP+1  ; AND XTND BYTE
EDB1:C9 82                    CMP   #$82             ;IF RAM ADDR >=8200 THEN BUMP TO
EDB3:90 0F   EDC4             BCC   NOADJ            ; NEXT BANK PAIR
EDB5:C0 80                    CPY   #$80
EDB7:90 0B   EDC4             BCC   NOADJ            ;=>NOT USING BANKPAIR
EDB9:C0 8F                    CPY   #$8F             ;SPECIAL BANK 0?
EDBB:F0 07   EDC4             BEQ   NOADJ            ;=>YES
EDBD:29 7F                    AND   #$7F             ;DROP HI ADDRESS AND
EDBF:85 D3                    STA   BUFTEMP+1        ; BUMP BANK NUMBER
EDC1:EE D3 14                 INC   $1400+BUFTEMP+1
EDC4:              *
EDC4:        EDC4  NOADJ      EQU   *
EDC4:A5 D3                    LDA   BUFTEMP+1        ;COPY BUFFER ADDRESS
EDC6:85 9C                    STA   BUF+1            ; FOR PRE & POSTNIB
EDC8:A5 D2                    LDA   BUFTEMP
EDCA:85 9B                    STA   BUF
EDCC:AD D3 14                 LDA   $1400+BUFTEMP+1
EDCF:8D 9C 14                 STA   $1400+BUF+1
EDD2:60                       RTS
EDD3:              *
EDD3:00 04 08 0C   SECTABLE   DFB   $00,$04,$08,$0C,$01,$05,$09,$0D
```

```
EDDB:              ----------------------------------------
EDDB:              * NAME    : MOREBLKS
EDDB:              * FUNCTION: SETUP TO DO NEXT BLOCK
EDDB:              * INPUT   : NONE
EDDB:              * RETURNS : 'BNE' IF MORE TO DO
EDDB:              *         : 'BEQ' IF NO MORE TO DO
EDDB:              * DESTROYS:NOTHING
EDDB:              ----------------------------------------
EDDB:              *
EDDB:        EDDB  MOREBLKS   EQU   *
EDDB:E6 D3                    INC   BUFTEMP+1        ;BUMP BUFFER ADDRESS
EDDD:E6 D3                    INC   BUFTEMP+1
EDDF:E6 D0                    INC   BLKTEMP          ;BUMP BLOCK NUMBER
EDE1:D0 02   EDE5             BNE   MORE2
EDE3:E6 D1                    INC   BLKTEMP+1
EDE5:        EDE5  MORE2      EQU   *
EDE5:C6 D9                    DEC   BLKCOUNT         ;MORE BLOCKS TO GO?
EDE7:60                       RTS                    ;RETURN RESULT OF DEC



EDE8:              ----------------------------------------
EDE8:              * NAME    : FIXIRQ
EDE8:              * FUNCTION: ENABLE IRQ IF APPROPRIATE
EDE8:              * INPUT   : NONE
EDE8:              * DESTROYS: NOTHING
EDE8:              ----------------------------------------
EDE8:              *
EDE8:        EDE8  FIXIRQ     EQU   *
EDE8:48                       PHA
EDE9:A5 DB                    LDA   IRQMASK          ;SHOULD IRQ BE ENABLED?
EDEB:30 01   EDEE             BMI   FIXRET           ;=>NO, LEAVE IT ALONE
EDED:58                       CLI                    ;ENABLE IRQ
EDEE:        EDEE  FIXRET     EQU   *
EDEE:68                       PLA
EDEF:60                       RTS
EDF0:                         CHN   DISK3.DATA.SRC
```

```
EDF0:              * GENERAL DATA:
EDF0:              *
EDF0:        0001  PREVUNIT   DS    1                ;PRIOR UNIT ACCESSED (FOR REPEAT)
EDF1:        0001  PREVCMD    DS    1                ;PRIOR CMD (FOR REPEAT)
EDF2:              *
EDF2:        0001  ESAVE      DS    1                ;SAVED E.REG
EDF3:        0001  VBLSAVE    DS    1                ;SAVED E.IER
EDF4:00            INITFLAG   DFB   0                ;<0 IS INITTED
EDF5:        0000             DO    REV0ROM
 S                 ROMREV     DS    1                ;0=REV0, <>0=REV1
EDF5:                         FIN
EDF5:              *
EDF5:              * MOTOR-UP TIMES PER COMMAND
EDF5:        0002  T50MS      EQU   $02              ; 50MS FOR MONTIMEH
EDF5:        0008  T200MS     EQU   $08              ;200 MS FOR MONTIMEH
EDF5:        0027  T1SEC      EQU   $27              ;1-SEC FOR MONTIMEH
EDF5:              *
EDF5:08 27 02      MTIMES     DFB   T200MS,T1SEC,T50MS ;READ,WRITE,SENSE
EDF8:              *
EDF8:              ----------------------------------------
EDF8:              * DRIVE TABLES:
EDF8:              *
EDF8:        0004  DRIVESEL   DS    4                ;NONZERO IF SELECTED
EDFC:              *
EDFC:        0004  UPTIME     DS    4                ;MOTOR RUNTIME SINCE STARTED
EE00:        0004  DRVTRACK   DS    4                ;CURRENT HEAD POSITION
```

```
EE04:        0000                DO    REV0ROM          ;ONLY IF SUPPORTING IT!
 S                  *          JUMP  TABLE            TO MONITOR ROUTINES.
 S                  *                THIS             TABLE FILLED IN BY 'INIT'.
 S                  *
 S                  JMPTAB     EQU   *
 S                  RDADR      JMP   *
 S                  READ       JMP   *
 S                  WRITE      JMP   *
 S                  SEEK       JMP   *
 S                  MSWAIT     JMP   *
 S                  PRENIB     JMP   *
 S                  POSTNIB    JMP   *
 S                  *
 S                  REV0       EQU   *                ;REV0 ADDRESSES
 S                             JMP   $F1BD            ;RDADR
 S                             JMP   $F148            ;READ
 S                             JMP   $F219            ;WRITE
 S                             JMP   $F400            ;SEEK
 S                             JMP   $F456            ;MSWAIT
 S                             JMP   $F2C6            ;PRENIB
 S                             JMP   $F311            ;POSTNIB
 S                  VSIZE      EQU   *-REV0           ;TABLE SIZE
 S                  *
 S                  REV1       EQU   *                ;REV1 ADDRESSES
 S                             JMP   $F1B9            ;RDADR
 S                             JMP   $F148            ;READ
 S                             JMP   $F216            ;WRITE
 S                             JMP   $F400            ;SEEK
 S                             JMP   $F456            ;MSWAIT
 S                             JMP   $F2C4            ;PRENIB
 S                             JMP   $F30F            ;POSTNIB
EE04:                          ELSE  ;FOR             REV1 WE USE EQUATES
EE04:        F1B9   RDADR      EQU   $F1B9            ;RDADR
EE04:        F148   READ       EQU   $F148            ;READ
EE04:        F216   WRITE      EQU   $F216            ;WRITE
EE04:        F400   SEEK       EQU   $F400            ;SEEK
EE04:        F456   MSWAIT     EQU   $F456            ;MSWAIT
EE04:        F2C4   PRENIB     EQU   $F2C4            ;PRENIB
EE04:        F30F   POSTNIB    EQU   $F30F            ;POSTNIB
EE04:                          FIN
EE04:        EE04   ZZEND      EQU   *
EE04:        056B   ZZLEN      EQU   *-ZZORG
EE04:        0000              IFNE  ZZLEN-LENDISK3
 S                             FAIL  2,"SOSORG        FILE IS INCORRECT FOR DISK3"
EE04:                          FIN
```

```
 ED0D ADD2           ED1E ADD2A          ED21 ADD3           ED0A ADDTIME
 EC08 ADRDELAY       E994 BADBLOCK       E981 BADCMD         E9B7 BADCOUNT
 EBA0 BADIO          E97C BADOP         ?2E00 BLABFMI        3200 BLABFM
 6B52 BLABUFMG       6955 BLACFM         5E99 BLADISK3       64D9 BLADMGR
 68F4 BLAFMGR       ?2CF8 BLAGLOB       ?2AF8 BLAINIT        55C0 BLAIPL
 2000 BLALODR       ?6E6E BLAMEMMG       5466 BLAOMSG        5466 BLAPATCH
 665E BLASCMGR       6404 BLASERR        5A8B BLAUMGR        ED98 BLK2SECT
   D9 BLKCOUNT       E9B1 BLKG255          D0 BLKTEMP          9B BUF
   D2 BUFTEMP        ED05 CHANGED        E999 CHKBYTE        ECDE CHKD1
 ECDC CHKDRV         E9B3 CHKLO          EAB9 CLRDRVS        E95E CMD1
 E965 CMD2           E975 CMD3             97 CSSTV            8C CURTRK
   C6 D.BLOCK          C3 D.BUFH           C2 D.BUFL           C4 D.BYTES
   C8 D.BYTRD          C0 D.COMMAND        C3 D.STATBUF        C2 D.STATCODE
   C1 D.UNITNUM     NE899 DIB1          NE8B9 DIB2          NE8D9 DIB3
NE8F9 DIB4           E9F9 DOIO           EA18 DOIO2          EDF8 DRIVESEL
 E9BC DRVSETUP       EE00 DRVTRACK       E9F4 DSWITCH        FFEE E.IER
 FFDF E.REG         X0010 E1908          ECD2 EDS1           EDF2 ESAVE
 EAF5 EXIT2          EAE9 EXIT           ECC8 EXTDESEL       EC14 FINDERR
 EC0E FINDGOOD       EB57 FINDIT         EBDB FINDSEC2       EBD5 FINDSECT
 EDE8 FIXIRQ         EDEE FIXRET         EB0B GOERR          EC5B GOFORIT
 EAE4 GOODINIT       EB35 GOSEEK         E972 GOSTAT           81 IBSLOT
   8B IMASK          EDF4 INITFLAG       EAA4 INIT           EAD9 INIT2
   8F INTRTRY        EBA2 IORETRY        E986 IOSETUP          DB IRQMASK
 C0EB IS.EXT         C0EA IS.INT         ECA6 ISD4          ?0400 LENBFMI
 2266 LENBFM         031C LENBUFMG       01FD LENCFM         056B LENDISK3
 0185 LENDMGR          61 LENFMGR       ?01B2 LENINIT        04CB LENIPL
 0AF8 LENLODR       ?0751 LENMEMMG       015A LENOMSG          00 LENPATCH
 0296 LENSCMGR         D5 LENSERR        040E LENUMGR        E91D MAIN
 C0D2 MD.EXT1        C0D0 MD.EXT2        C0D5 MD.INT         EB46 MDELAY
   9A MONTIMEH         99 MONTIMEL       EDE5 MORE2          EDDB MOREBLKS
 C0E8 MOTOROFF       C0E9 MOTORON        C0D3 MS.EXT1        C0D1 MS.EXT2
 C0D4 MS.INT         F456 MSWAIT         EDF5 MTIMES         ED60 MYSEEK
 EDC4 NOADJ          C0D8 NOSCROLL       EC9B NOTD2          EBBF NOTSAME
 BC00 ORGBFM         B800 ORGBFMI        F552 ORGBUFMG       F355 ORGCFM
 E899 ORGDISK3       EED9 ORGDMGR        FFBF ORGEND         F2F4 ORGFMGR
?18FC ORGGLOB        28F8 ORGINIT        DFC0 ORGIPL         1E00 ORGLODR
 F86E ORGMEMMG       DE66 ORGOMSG        DE66 ORGPATCH       F05E ORGSCMGR
 EE04 ORGSERR        E48B ORGUMGR        F30F POSTNIB        F2C4 PRENIB
 EDF1 PREVCMD        EDF0 PREVUNIT       C08D Q6H            C08C Q6L
?C08F Q7H            C08E Q7L              03 R.FIND           04 R.IOERR
   06 R.IRQ            01 R.RECAL        EBFF RDADERR        F1B9 RDADR
 EA4D READOK         F148 READ           EA52 READERR        EA2C READREQ2
 EA23 READREQ        ED28 RECAL1         ED3A RECAL2         ED3F RECAL3
   D8 RECALCNT       ED26 RECAL            D6 RETRYADR         D7 RETRYCNT
   00 REV0ROM        E958 RPTOK          EDD3 SECTABLE       EB0E SECTORIO
   D5 SECTOR         F400 SEEK           EB2D SEEKDEL       NED57 SEEKDSK3
   DA SEEKWAIT       EC75 SELECT         EC8B SELEXT         EC6B SETTIME
 EBC2 SIOERR         EBCC SIOGOOD        EBCF SIORET         EBC7 SIOWPROT
 EB90 SIOWRITE       EB3A SOUGHT         EC3D SPINNING       EA8B STATUS
X0006 SYSERR         0027 T1SEC          0008 T200MS           02 T50MS
   DC TEMP             00 TEST             D4 TRACK            9E TRKN
 EBA6 TRYRECAL       ECBA UNITDEL        ECAC UNITRET        ECC7 UNITRTS
 EC1D UNITSEL        EDFC UPTIME         EDF3 VBLSAVE        EA55 WRITEREQ
 F216 WRITE          EA88 WRITERR       X0008 XBADOP        X000D XBLKNUM
X000C XBYTECNT      X000F XCTLCODE      X000E XDISKSW       X000A XIOERROR
X0009 XNODRIVE      X000B XNOWRITE      X0007 XREQCODE      ?EE04 ZZEND
```

```
 056B ZZLEN          E899 ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED  1398
** FREE SPACE PAGE COUNT   77
```

```
SOURCE   FILE #01 =>SYSERR.SRC
 INCLUDE FILE #02 =>SOSORG
```

```
0000:                             REL
0000:                             INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955      BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52      BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E      BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
EE04:        EE04                 ORG   ORGSERR
EE04:        EE04      ZZORG      EQU   *
EE04:                             MSB   OFF
EE04:                  ************************************************************
EE04:                  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
EE04:                  *                    ALL RIGHTS RESERVED
EE04:                  ************************************************************
EE04:                  *
EE04:                  * SYSTEM ERROR ROUTINES  (VERSION = 1.1O   )
EE04:                  *                        (DATE    = 12/02/81)
EE04:                  *
EE04:                  * THIS MODULE CONTAINS THE SYSTEM ERROR AND SYSTEM FAILURE ROUTINES.
EE04:                  *
EE04:                  ************************************************************
EE04:                  *
EE04:        EE17                 ENTRY SYSERR
EE04:        EE2A                 ENTRY SYSDEATH
EE04:                  *
EE04:        0000                 EXTRN SERR
EE04:        0000                 EXTRN SDEATH.REGS
EE04:        0000                 EXTRN SCRNMODE
```

```
EE04:               ************************************************************
EE04:               *
EE04:               * DATA DECLARATIONS
EE04:               *
EE04:               ************************************************************
EE04:               *
EE04:        FFDF   E.REG      EQU   $FFDF
EE04:        FFD0   Z.REG      EQU   $FFD0
EE04:        FFEF   B.REG      EQU   $FFEF
EE04:               *
EE04:        0009   S.SAVE     EQU   $09              ; REGISTER SAVE AREA
EE04:        0008   PCH.SAVE   EQU   $08
EE04:        0007   PCL.SAVE   EQU   $07
EE04:        0006   P.SAVE     EQU   $06
EE04:        0005   A.SAVE     EQU   $05
EE04:        0004   X.SAVE     EQU   $04
EE04:        0003   Y.SAVE     EQU   $03
EE04:        0002   E.SAVE     EQU   $02
EE04:        0001   Z.SAVE     EQU   $01
EE04:        0000   B.SAVE     EQU   $00
EE04:               *
EE04:        FFFA   NMI.VECTOR EQU   $FFFA
EE04:               *
EE04:        C050   TXT.CLR    EQU   $C050
EE04:        C052   MIX.CLR    EQU   $C052
EE04:        C056   HIRES.CLR  EQU   $C056
EE04:               *
EE04:        C054   PG2.CLR    EQU   $C054
EE04:               *
EE04:        07E4   MSGBASE    EQU   $7E4
EE04:        0BE4   MSGBASE2   EQU   $BE4
EE04:20 53 59 53    MSG        ASC   '                SYSTEM FAILURE = $'
EE17:        0013   MSGLEN     EQU   *-MSG
```

```
EE17:               ************************************************************
EE17:               *
EE17:               * SYSTEM ERROR ROUTINE
EE17:               *
EE17:               * THIS ROUTINE IS CALLED WHEN AN ERROR CONDITION HAS BEEN
EE17:               * ENCOUNTERED.  THE ERROR NUMBER IS PASSED IN THE A REG
EE17:               * AND THE CALL TO THIS ROUTINE MUST ALWAYS BE A JSR.
EE17:               *
EE17:               ************************************************************
EE17:         EE17  SYSERR     EQU   *
EE17:               *
EE17:8D 00 00                  STA   SERR
EE1A:68                        PLA
EE1B:8D 07 00                  STA   SDEATH.REGS+PCL.SAVE
EE1E:68                        PLA
EE1F:8D 08 00                  STA   SDEATH.REGS+PCH.SAVE
EE22:38                        SEC
EE23:AD 00 00                  LDA   SERR
EE26:D0 01   EE29              BNE   SERR.EXIT
EE28:18                        CLC
EE29:60             SERR.EXIT  RTS                    ; RETURNS ONE LEVEL BEYOND CALLER
```

```
EE2A:              ************************************************************
EE2A:              *
EE2A:              * SYSTEM DEATH ROUTINE
EE2A:              *
EE2A:              * CALLED TO IMMEDIATELY TERMINATE EXECUTION OF THE MACHINE
EE2A:              * BECAUSE A FATAL ERROR HAS BEEN DETECTED BY THE OPERATING
EE2A:              * SYSTEM.  THE ERROR CODE IS PASSED IN THE A REG.  THE
EE2A:              * CALL TO THIS ROUTINE MUST ALWAYS BE A JSR.
EE2A:              *
EE2A:              ************************************************************
EE2A:        EE2A  SYSDEATH   EQU   *
EE2A:              *
EE2A:8D 05 00                 STA   SDEATH.REGS+A.SAVE ; SAVE REGISTERS
EE2D:8E 04 00                 STX   SDEATH.REGS+X.SAVE
EE30:8C 03 00                 STY   SDEATH.REGS+Y.SAVE
EE33:08                       PHP
EE34:68                       PLA
EE35:8D 06 00                 STA   SDEATH.REGS+P.SAVE
EE38:BA                       TSX
EE39:8E 09 00                 STX   SDEATH.REGS+S.SAVE
EE3C:AD DF FF                 LDA   E.REG
EE3F:8D 02 00                 STA   SDEATH.REGS+E.SAVE
EE42:AD D0 FF                 LDA   Z.REG
EE45:8D 01 00                 STA   SDEATH.REGS+Z.SAVE
EE48:AD EF FF                 LDA   B.REG
EE4B:8D 00 00                 STA   SDEATH.REGS+B.SAVE
EE4E:68                       PLA
EE4F:8D 07 00                 STA   SDEATH.REGS+PCL.SAVE
EE52:68                       PLA
EE53:8D 08 00                 STA   SDEATH.REGS+PCH.SAVE
EE56:              *
EE56:78                       SEI                    ; TURN OFF INTERRUPTS
EE57:D8                       CLD
EE58:              *
EE58:A2 00                    LDX   #0               ; SAVE SYSTEM STACK PAGE IN PAGE $17
EE5A:BD 00 01      SD005      LDA   $100,X
EE5D:9D 00 17                 STA   $1700,X
EE60:CA                       DEX
EE61:D0 F7   EE5A             BNE   SD005
EE63:              *
EE63:AD 59 C0                 LDA   $C059            ; ENSURE SILENTYPE PORT SHUT DOWN
EE66:AD DD C0                 LDA   $C0DD
EE69:AD DF C0                 LDA   $C0DF
EE6C:AD 5F C0                 LDA   $C05F
EE6F:AD 5A C0                 LDA   $C05A
EE72:              *
EE72:AD 40 C0                 LDA   $C040            ; SOUND BELL
EE75:              *
EE75:A9 74                    LDA   #$74             ; ENSURE RESET LOCK OFF & RAM SWITCHED IN.
EE77:8D DF FF                 STA   E.REG
EE7A:              *
EE7A:AD 50 C0                 LDA   TXT.CLR          ; SWITCH TO 40 COL B&W DISPLAY MODE
EE7D:AD 52 C0                 LDA   MIX.CLR
EE80:AD 56 C0                 LDA   HIRES.CLR
EE83:AD 54 C0                 LDA   PG2.CLR          ; & SELECT PAGE 1
EE86:              *
```

```
EE86:A9 02                    LDA   #$02
EE88:2C 00 00                 BIT   SCRNMODE
EE8B:70 0F   EE9C             BVS   SD015            ; IF GRAPHICS MODE THEN KEEP 40 COL MODE
EE8D:F0 0D   EE9C             BEQ   SD015            ; IF 40 COL MODE THEN KEEP
EE8F:AD 53 C0                 LDA   MIX.CLR+1        ; ELSE SWITCH TO 80 COL DISPLAY MODE
EE92:              *
EE92:A2 14                    LDX   #MSGLEN+1        ; ENSURE BKGRND SET TO INVERSE SPACES
EE94:A9 20                    LDA   #$20             ; SPACE CHAR W/INVERSE
EE96:9D E3 0B      SD010      STA   MSGBASE2-1,X
EE99:CA                       DEX
EE9A:10 FA   EE96             BPL   SD010
EE9C:              *
EE9C:A2 00         SD015      LDX   #0               ; MOVE MSG TO TEXT SCREEN
EE9E:BD 04 EE      SD020      LDA   MSG,X
EEA1:9D E3 07                 STA   MSGBASE-1,X
EEA4:E8                       INX
EEA5:E0 13                    CPX   #MSGLEN
EEA7:D0 F5   EE9E             BNE   SD020
EEA9:              *
EEA9:AD 05 00                 LDA   SDEATH.REGS+A.SAVE ; DISPLAY ERROR CODE (2 HEX DIGITS)
EEAC:18                       CLC
EEAD:4A                       LSR   A
EEAE:4A                       LSR   A
EEAF:4A                       LSR   A
EEB0:4A                       LSR   A
EEB1:20 CB EE                 JSR   PRINT            ; FIRST DIGIT
EEB4:E8                       INX
EEB5:AD 05 00                 LDA   SDEATH.REGS+A.SAVE
EEB8:29 0F                    AND   #$0F
EEBA:20 CB EE                 JSR   PRINT            ; SECOND DIGIT
EEBD:              *
EEBD:A9 CA                    LDA   #>SD100
EEBF:8D FA FF                 STA   NMI.VECTOR
EEC2:A9 EE                    LDA   #<SD100
EEC4:8D FB FF                 STA   NMI.VECTOR+1
EEC7:              *
EEC7:              *
EEC7:4C C7 EE                 JMP   *                ; HANG UNTIL REBOOT (CTRL/RESET)
EECA:              ************************************************************
EECA:40            SD100      RTI                    ; NMI VECTOR POINT HERE TO MASK THEM OUT
EECB:              *
EECB:              *
EECB:              * PRINT SUBROUTINE
EECB:              *
EECB:        EECB  PRINT      EQU   *
EECB:C9 0A                    CMP   #$A
EECD:B0 04   EED3             BCS   PRNT100
EECF:69 30                    ADC   #$30             ; "0"-"9"
EED1:90 02   EED5             BCC   PRNT110          ; ALWAYS TAKEN
EED3:69 36         PRNT100    ADC   #$36             ; "A"-"F"
EED5:9D E3 07      PRNT110    STA   MSGBASE-1,X
EED8:60                       RTS
EED9:              *

EED9:                         LST   ON
EED9:        EED9  ZZEND      EQU   *
```

```
EED9:        00D5  193 ZZLEN      EQU   ZZEND-ZZORG
EED9:        0000  194            IFNE  ZZLEN-LENSERR
 S                 195            FAIL  2,"SOSORG        FILE IS INCORRECT FOR SYSERR"
EED9:              196            FIN
```

```
   05 A.SAVE         FFEF B.REG           00 B.SAVE         3200 BLABFM
?2E00 BLABFMI        6B52 BLABUFMG      6955 BLACFM         5E99 BLADISK3
 64D9 BLADMGR        68F4 BLAFMGR      ?2CF8 BLAGLOB       ?2AF8 BLAINIT
 55C0 BLAIPL         2000 BLALODR      ?6E6E BLAMEMMG       5466 BLAOMSG
 5466 BLAPATCH       665E BLASCMGR      6404 BLASERR        5A8B BLAUMGR
 FFDF E.REG            02 E.SAVE        C056 HIRES.CLR      2266 LENBFM
?0400 LENBFMI        031C LENBUFMG      01FD LENCFM         056B LENDISK3
 0185 LENDMGR          61 LENFMGR      ?01B2 LENINIT        04CB LENIPL
 0AF8 LENLODR       ?0751 LENMEMMG      015A LENOMSG          00 LENPATCH
 0296 LENSCMGR         D5 LENSERR       040E LENUMGR        C052 MIX.CLR
 EE04 MSG            07E4 MSGBASE       0BE4 MSGBASE2         13 MSGLEN
 FFFA NMI.VECTOR     BC00 ORGBFM        B800 ORGBFMI        F552 ORGBUFMG
 F355 ORGCFM         E899 ORGDISK3      EED9 ORGDMGR        FFBF ORGEND
 F2F4 ORGFMGR       ?18FC ORGGLOB       28F8 ORGINIT        DFC0 ORGIPL
 1E00 ORGLODR        F86E ORGMEMMG      DE66 ORGOMSG        DE66 ORGPATCH
 F05E ORGSCMGR       EE04 ORGSERR       E48B ORGUMGR          06 P.SAVE
   08 PCH.SAVE         07 PCL.SAVE      C054 PG2.CLR        EECB PRINT
 EED3 PRNT100        EED5 PRNT110         09 S.SAVE        X0005 SCRNMODE
 EE5A SD005          EE96 SD010         EE9C SD015          EE9E SD020
 EECA SD100         X0004 SDEATH.REGS  X0003 SERR           EE29 SERR.EXIT
NEE2A SYSDEATH      NEE17 SYSERR        C050 TXT.CLR          04 X.SAVE
   03 Y.SAVE         FFD0 Z.REG           01 Z.SAVE         EED9 ZZEND
   D5 ZZLEN          EE04 ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   255
** FREE SPACE PAGE COUNT   84
```

```
SOURCE   FILE #01 =>DEVMGR.SRC
 INCLUDE FILE #02 =>SOSORG
```

```
0000:                             REL
0000:                             INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955      BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52      BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E      BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
EED9:        EED9                 ORG   ORGDMGR
EED9:        EED9      ZZORG      EQU   *
EED9:                             MSB   OFF
EED9:
****************************************************************************************************
EED9:                  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
EED9:                  *                    ALL RIGHTS RESERVED
EED9:
****************************************************************************************************
EED9:                  *
EED9:                  * DEVICE MANAGER (VERSION = 1.1O   )
EED9:                  *                (DATE    = 8/04/81)
EED9:                  *
EED9:                  * THIS MODULE IS RESPONSIBLE FOR CALLING THE CORRECT DEVICE
EED9:                  * DRIVER WHEN A D.READ...D.INIT SYSTEM CALL IS MADE.
EED9:                  * (NOTE:  D.OPEN,D.CLOSE AND D.INIT ARE ONLY CALLABLE FROM
EED9:                  * INSIDE THE OPERATING SYSTEM).  D.INFO AND GET.DNUM CALLS
EED9:                  * ARE HANDLED INSIDE THIS MODULE. REPEAT.IO BYPASSES THIS MODULE.
EED9:
****************************************************************************************************
EED9:                  *
EED9:        EF7D                 ENTRY DMGR
EED9:                  *
EED9:        EED9                 ENTRY MAX.DNUM
EED9:        0019                 ENTRY SDT.SIZE
EED9:        EEDA                 ENTRY SDT.DIBL
EED9:        EEF3                 ENTRY SDT.DIBH
EED9:        EF0C                 ENTRY SDT.ADRL
EED9:        EF25                 ENTRY SDT.ADRH
EED9:        EF3E                 ENTRY SDT.BANK
EED9:        EF57                 ENTRY SDT.UNIT
EED9:        000D                 ENTRY BLKD.SIZE
EED9:        EF70                 ENTRY BLKDLST
EED9:                  *
EED9:        0000                 EXTRN SYSERR
EED9:        0000                 EXTRN SERR
EED9:        0000                 EXTRN NODNAME
EED9:        0000                 EXTRN BADDNUM
EED9:        0000                 EXTRN SYSDEATH
EED9:        0000                 EXTRN BADSYSCALL
EED9:                  *
EED9:        0000                 EXTRN SXPAGE
EED9:                  *
EED9:        FFDF      E.REG      EQU   $FFDF            ; ENVIRONMENT REGISTER
EED9:        FFEF      B.REG      EQU   $FFEF            ; BANK REGISTER
```

```
EED9:
****************************************************************************************************
EED9:                  *
EED9:                  * SYSTEM DEVICE TABLE (SDT)
EED9:                  *
EED9:                  * CONTAINS THE ADDRESS OF EACH DRIVER'S DIB (SDT.DIB), THE
EED9:                  * ADDRESS OF EACH DRIVER'S ENTRY POINT (SDT.ADR), AND THE
EED9:                  * UNIT # OF EACH DRIVER (SDT.UNIT).  THE TABLE IS INDEXED
EED9:                  * BY DEVICE NUMBER.  ENTRY 0 IS RESERVED FOR FUTURE USE.
EED9:                  *
EED9:
****************************************************************************************************
EED9:                  *
EED9:        0019      SDT.SIZE   EQU   25
EED9:                  *
EED9:        0001      MAX.DNUM   DS    1                ;MAX DEV NUMBER IN SYSTEM+1
EEDA:        0019      SDT.DIBL   DS    SDT.SIZE         ;ADR OF DEVICE INFORMATION BLOCK
EEF3:        0019      SDT.DIBH   DS    SDT.SIZE
EF0C:                  *
EF0C:        0019      SDT.ADRL   DS    SDT.SIZE         ;ADR OF ENTRY POINT
EF25:        0019      SDT.ADRH   DS    SDT.SIZE
EF3E:                  *
EF3E:        0019      SDT.BANK   DS    SDT.SIZE         ;BANK # OF DEVICE
EF57:                  *
EF57:        0019      SDT.UNIT   DS    SDT.SIZE         ;UNIT  # OF DRIVER
EF70:                  *
EF70:
****************************************************************************************************
EF70:                  * BLOCK DEVICE LIST TABLE
EF70:                  *
EF70:        000D      BLKD.SIZE  EQU   13
EF70:00                BLKDLST    DFB   $00
EF71:        000C                 DS    BLKD.SIZE-1
```

```
EF7D:
****************************************************************************************************
EF7D:                   *
EF7D:                   * DATA DECLARATIONS
EF7D:                   *
EF7D:
****************************************************************************************************
EF7D:                   *
EF7D:        00C0       D.TPARMX   EQU   $C0
EF7D:        00C0       REQCODE    EQU   D.TPARMX
EF7D:                   *
EF7D:                   * D.READ/WRITE/CTRL/STATUS/OPEN/CLOSE/INIT/REPEAT PARMS
EF7D:                   *
EF7D:        00C1       DNUM       EQU   D.TPARMX+1
EF7D:                   *
EF7D:                   * D.INFO PARMS
EF7D:                   *
EF7D:        00C1       I.DNUM     EQU   D.TPARMX+1
EF7D:        00C2       I.DNAME    EQU   D.TPARMX+2
EF7D:        00C4       I.DLIST    EQU   D.TPARMX+4
EF7D:        00C6       I.LENGTH   EQU   D.TPARMX+6
EF7D:                   *
EF7D:                   * GET.DEV.NUM PARMS
EF7D:                   *
EF7D:        00C1       G.DNAME    EQU   D.TPARMX+1
EF7D:        00C3       G.DNUM     EQU   D.TPARMX+3
EF7D:                   *
EF7D:                   * SDT ENTRY (=DIB) FIELDS
EF7D:                   *
EF7D:        0011       DIB.SLOT   EQU   $11              ;DIB'S DEVICE SLOT FIELD
EF7D:        0013       DIB.DTYPE  EQU   $13              ;DIB'S DEVICE TYPE FIELD
EF7D:                   *
EF7D:        00D0       SDTP       EQU   D.TPARMX+$10     ; PTR TO CURRENT SDT ENTRY
```

```
EF7D:
****************************************************************************************************
EF7D:                  *
EF7D:                  * DEVICE MANAGER (MAIN ENTRY POINT)
EF7D:                  *
EF7D:
****************************************************************************************************
EF7D:        EF7D      DMGR       EQU   *
EF7D:                  *
EF7D:A5 C0                        LDA   REQCODE
EF7F:C9 04                        CMP   #4
EF81:90 12   EF95                 BCC   DRIVER           ; D.READ/WRITE/CTRL/STATUS CALL
EF83:D0 03   EF88                 BNE   DM000
EF85:4C 17 F0                     JMP   GET.DNUM         ; GET.DEV.NUM CALL
EF88:C9 05             DM000      CMP   #5
EF8A:F0 51   EFDD                 BEQ   D.INFO           ; D.INFO CALL
EF8C:C9 0A                        CMP   #$A
EF8E:90 05   EF95                 BCC   DRIVER           ; D.OPEN/CLOSE/INIT
EF90:A9 00                        LDA   #BADSYSCALL      ; ELSE FATAL ERROR
EF92:20 00 00                     JSR   SYSDEATH         ; EXIT
```

```
EF95:              
****************************************************************************************************
EF95:              * D.READ/WRITE/CTRL/STATUS/OPEN/CLOSE/INIT CALLS
EF95:              * "JSR" TO DEVICE DRIVER
EF95:              
****************************************************************************************************
EF95:        EF95  DRIVER     EQU   *
EF95:              *
EF95:A6 C1                    LDX   DNUM             ; GET DNUM SYSCALL PARM
EF97:F0 05   EF9E             BEQ   DM005            ; WITHIN BOUNDS?
EF99:EC D9 EE                 CPX   MAX.DNUM         ;      "
EF9C:90 05   EFA3             BCC   DM010
EF9E:              *
EF9E:              * DNUM TOO LARGE
EF9E:              *
EF9E:A9 00         DM005      LDA   #>BADDNUM        ; INVALID DEVICE NUMBER
EFA0:20 00 00                 JSR   SYSERR           ;   ERROR EXIT
EFA3:              *
EFA3:              * MAP DEV# TO UNIT#
EFA3:              *
EFA3:BD 57 EF      DM010      LDA   SDT.UNIT,X
EFA6:85 C1                    STA   DNUM
EFA8:              *
EFA8:              * "JSR" TO DEVICE DRIVER VIA JMP TABLE
EFA8:              *
EFA8:AD EF FF                 LDA   B.REG            ; STACK B.REG
EFAB:48                       PHA
EFAC:A9 EF                    LDA   #<DM.RTN-1       ; STACK RETURN ADDRESS
EFAE:48                       PHA
EFAF:A9 C8                    LDA   #>DM.RTN-1
EFB1:48                       PHA
EFB2:              *
EFB2:BD 3E EF                 LDA   SDT.BANK,X       ; SELECT RAM BANK
EFB5:8D EF FF                 STA   B.REG
EFB8:BD 25 EF                 LDA   SDT.ADRH,X       ; STACK DRIVER ENTRY POINT ADDRESS
EFBB:48                       PHA
EFBC:BD 0C EF                 LDA   SDT.ADRL,X
EFBF:48                       PHA
EFC0:              *
EFC0:AD DF FF                 LDA   E.REG            ; SWITCH IN I/O BANK
EFC3:09 40                    ORA   #$40
EFC5:8D DF FF                 STA   E.REG
EFC8:60                       RTS                    ; AND, "JSR" TO DEVICE DRIVER
EFC9:              *
EFC9:AD DF FF      DM.RTN     LDA   E.REG            ; SWITCH OUT I/O BANK
EFCC:29 BF                    AND   #$BF
EFCE:8D DF FF                 STA   E.REG
EFD1:68                       PLA                    ; RESTORE B.REG
EFD2:8D EF FF                 STA   B.REG
EFD5:38                       SEC
EFD6:AD 00 00                 LDA   SERR             ; RETRIEVE ERROR CODE
EFD9:D0 01   EFDC             BNE   DM017            ; ENSURE CARRY CLEARED IF NO ERROR
EFDB:18                       CLC
EFDC:60            DM017      RTS                    ; AND, EXIT TO CALLER
```

```
EFDD:
****************************************************************************************************
EFDD:              * D.INFO(IN.DNUM, OUT.DNAME, OUT.DEVLIST, IN.LENGTH) SYSTEM CALL
EFDD:
****************************************************************************************************
EFDD:        EFDD  D.INFO     EQU   *
EFDD:              *
EFDD:A6 C1                    LDX   I.DNUM           ; GET DNUM PARM
EFDF:F0 05   EFE6             BEQ   DM020            ; WITHIN BOUNDS?
EFE1:EC D9 EE                 CPX   MAX.DNUM         ;      "
EFE4:90 05   EFEB             BCC   DM030
EFE6:A9 00         DM020      LDA   #>BADDNUM        ; NO, DNUM TOO LARGE
EFE8:20 00 00                 JSR   SYSERR           ; ERROR EXIT
EFEB:              *
EFEB:              * MOVE PARMS FM SDT ENTRY (DEV INFO BLOCK) TO CALLER'S
EFEB:              * PARM LIST
EFEB:              *
EFEB:20 48 F0      DM030      JSR   SETUP.SDT        ; SET UP ZPAGE PTR TO SDT ENTRY
EFEE:              *
EFEE:              * OUPUT DNAME PARM
EFEE:              *
EFEE:B1 D0                    LDA   (SDTP),Y         ; LOAD PARM'S BYTE COUNT
EFF0:A8                       TAY
EFF1:B1 D0         DM040      LDA   (SDTP),Y
EFF3:91 C2                    STA   (I.DNAME),Y
EFF5:88                       DEY
EFF6:10 F9   EFF1             BPL   DM040
EFF8:              *
EFF8:              * OUTPUT DEVINFO PARM (SLOT,UNIT,DEVID,PRODCODE)
EFF8:              *
EFF8:A9 11                    LDA   #DIB.SLOT
EFFA:18                       CLC                    ; ADVANCE SDTP TO 2ND PARM IN SDT
EFFB:65 D0                    ADC   SDTP
EFFD:85 D0                    STA   SDTP
EFFF:90 02   F003             BCC   DM045
F001:E6 D1                    INC   SDTP+1
F003:A4 C6         DM045      LDY   I.LENGTH         ; LOAD BYTE COUNT
F005:F0 0E   F015             BEQ   DM.EXIT          ; IF 0 THEN DONE
F007:88                       DEY
F008:C0 0B                    CPY   #$B
F00A:90 02   F00E             BCC   DM050
F00C:A0 0A                    LDY   #$A
F00E:B1 D0         DM050      LDA   (SDTP),Y
F010:91 C4                    STA   (I.DLIST),Y
F012:88                       DEY
F013:10 F9   F00E             BPL   DM050
F015:              *
F015:18            DM.EXIT    CLC
F016:60                       RTS                    ; NORMAL EXIT
```

```
F017:
****************************************************************************************************
F017:              * GET.DEV.NUM(IN.DNAME; OUT.DNUM) SYSTEM CALL
F017:
****************************************************************************************************
F017:              *
F017:        F017  GET.DNUM   EQU   *
F017:              *
F017:A2 01                    LDX   #1               ; SETUP PTR TO 1ST SDT ENTRY
F019:              *
F019:20 48 F0      DM070      JSR   SETUP.SDT        ; SET UP ZPAGE PTR TO SDT ENTRY
F01C:              *
F01C:B1 D0                    LDA   (SDTP),Y         ; COMPARE DNAME LENGTHS
F01E:D1 C1                    CMP   (G.DNAME),Y
F020:D0 1B   F03D             BNE   NXTSDT
F022:              *
F022:A8                       TAY                    ; LENGTHS MATCH, NOW COMPARE CHARS
F023:B1 C1         DM080      LDA   (G.DNAME),Y
F025:C9 60                    CMP   #$60
F027:90 02   F02B             BCC   DM090
F029:29 DF                    AND   #$DF             ; UPSHIFT
F02B:D1 D0         DM090      CMP   (SDTP),Y
F02D:D0 0E   F03D             BNE   NXTSDT
F02F:88                       DEY
F030:D0 F1   F023             BNE   DM080
F032:              *
F032:8A                       TXA                    ; CHARS MATCH
F033:A0 00                    LDY   #0
F035:91 C3                    STA   (G.DNUM),Y       ; OUTPUT DEV NUM PARM
F037:A0 13                    LDY   #DIB.DTYPE       ; SET "N" FLAG IN STATUS REG.
F039:B1 D0                    LDA   (SDTP),Y         ; N=1(BLOCK DEVICE) N=0(CHAR DEVICE)
F03B:18                       CLC
F03C:60                       RTS                    ; NORMAL EXIT
F03D:              *
F03D:E8            NXTSDT     INX                    ; LAST SDT ENTRY?
F03E:EC D9 EE                 CPX   MAX.DNUM
F041:90 D6   F019             BCC   DM070
F043:              *
F043:A9 00                    LDA   #>NODNAME        ; ERROR, DNAME NOT FOUND IN SDT
F045:20 00 00                 JSR   SYSERR           ;  RETURN TO CALLER
```

```
F048:
****************************************************************************************************
F048:              * SETUP.SDT(IN.X=DNUM, OUT.SDTP, B.REG, Y=0)   X="UNCHANGED"
F048:
****************************************************************************************************
F048:        F048  SETUP.SDT  EQU   *
F048:BD DA EE                 LDA   SDT.DIBL,X       ; SET UP ZPAGE PTR TO SDT ENTRY
F04B:85 D0                    STA   SDTP             ; (POINTS TO DNAME FIELD)
F04D:BD F3 EE                 LDA   SDT.DIBH,X
F050:85 D1                    STA   SDTP+1
F052:BD 3E EF                 LDA   SDT.BANK,X
F055:8D EF FF                 STA   B.REG
F058:A0 00                    LDY   #0
F05A:8C D1 00                 STY   SXPAGE+SDTP+1
F05D:60                       RTS
F05E:              *

F05E:                         LST   ON
F05E:        F05E  ZZEND      EQU   *
F05E:        0185  ZZLEN      EQU   ZZEND-ZZORG
F05E:        0000             IFNE  ZZLEN-LENDMGR
 S                            FAIL  2,"SOSORG        FILE IS INCORRECT FOR DEVMGR"
F05E:                         FIN
```

```
 FFEF B.REG         X000F BADDNUM       X0011 BADSYSCALL    ?2E00 BLABFMI
 3200 BLABFM         6B52 BLABUFMG       6955 BLACFM         5E99 BLADISK3
 64D9 BLADMGR        68F4 BLAFMGR       ?2CF8 BLAGLOB       ?2AF8 BLAINIT
 55C0 BLAIPL         2000 BLALODR       ?6E6E BLAMEMMG       5466 BLAOMSG
 5466 BLAPATCH       665E BLASCMGR       6404 BLASERR        5A8B BLAUMGR
N000D BLKD.SIZE     NEF70 BLKDLST        EFDD D.INFO           C0 D.TPARMX
   13 DIB.DTYPE        11 DIB.SLOT       F015 DM.EXIT        EFC9 DM.RTN
 EF88 DM000          EF9E DM005          EFA3 DM010          EFDC DM017
 EFE6 DM020          EFEB DM030          EFF1 DM040          F003 DM045
 F00E DM050          F019 DM070          F023 DM080          F02B DM090
NEF7D DMGR             C1 DNUM           EF95 DRIVER         FFDF E.REG
   C1 G.DNAME          C3 G.DNUM         F017 GET.DNUM         C4 I.DLIST
   C2 I.DNAME          C1 I.DNUM           C6 I.LENGTH      ?0400 LENBFMI
 2266 LENBFM         031C LENBUFMG       01FD LENCFM         056B LENDISK3
 0185 LENDMGR          61 LENFMGR       ?01B2 LENINIT        04CB LENIPL
 0AF8 LENLODR       ?0751 LENMEMMG       015A LENOMSG          00 LENPATCH
 0296 LENSCMGR         D5 LENSERR        040E LENUMGR       NEED9 MAX.DNUM
X000E NODNAME        F03D NXTSDT         B800 ORGBFMI        BC00 ORGBFM
 F552 ORGBUFMG       F355 ORGCFM         E899 ORGDISK3       EED9 ORGDMGR
 FFBF ORGEND         F2F4 ORGFMGR       ?18FC ORGGLOB        28F8 ORGINIT
 DFC0 ORGIPL         1E00 ORGLODR        F86E ORGMEMMG       DE66 ORGOMSG
 DE66 ORGPATCH       F05E ORGSCMGR       EE04 ORGSERR        E48B ORGUMGR
   C0 REQCODE       NEF25 SDT.ADRH      NEF0C SDT.ADRL      NEF3E SDT.BANK
NEEF3 SDT.DIBH      NEEDA SDT.DIBL      N0019 SDT.SIZE      NEF57 SDT.UNIT
   D0 SDTP          X000D SERR           F048 SETUP.SDT     X0012 SXPAGE
X0010 SYSDEATH      X000C SYSERR         F05E ZZEND          0185 ZZLEN
 EED9 ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   347
** FREE SPACE PAGE COUNT   84
```

```
SOURCE   FILE #01 =>SCMGR.SRC
 INCLUDE FILE #02 =>SOSORG
```

```
0000:                              REL
0000:                              INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                    *  SOS KERNEL MODULE ORIGINS
0000:        1E00        ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8        ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC        ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800        ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00        ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66        ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66        ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0        ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B        ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899        ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04        ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9        ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E        ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4        ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355        ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552        ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E        ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF        ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                    *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8        LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2        LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400        LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266        LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000        LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A        LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB        LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E        LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B        LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5        LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185        LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296        LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061        LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD        LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C        LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751        LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                    *      SOS BLOAD ADDRESSES
0000:        2000        BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8        BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8        BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00        BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200        BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466        BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466        BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0        BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B        BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99        BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404        BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9        BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E        BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4        BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955      BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52      BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E      BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
F05E:        F05E                 ORG   ORGSCMGR
F05E:        F05E      ZZORG      EQU   *
F05E:                             MSB   OFF
F05E:                  ************************************************************
F05E:                  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
F05E:                  *                    ALL RIGHTS RESERVED
F05E:                  ************************************************************
F05E:                  *
F05E:                  * SYSTEM CALL MANAGER (VERSION = 1.1O  )
F05E:                  *                     (DATE    = 8/04/81)
F05E:                  *
F05E:                  * THE SYSTEM CALL MANAGER:
F05E:                  * (1) RETRIEVE THE SYSCALL #,
F05E:                  * (2) DETERMINE THE LOCATION OF THE SYSTEM CALL PARMS AND
F05E:                  *     MOVE THEM TO THE SOS ZPAGE,
F05E:                  * (3) TRANSFER CONTROL TO THE APPROPRIATE INTERFACE MANAGER,
F05E:                  *     (FILE,DEVICE,UTILITY,MEMORY)
F05E:                  *
F05E:                  ************************************************************
F05E:                  *
F05E:        F0F6                 ENTRY SCMGR
F05E:                  *
F05E:        0000                 EXTRN FMGR
F05E:        0000                 EXTRN DMGR
F05E:        0000                 EXTRN UMGR
F05E:        0000                 EXTRN MMGR
F05E:        0000                 EXTRN DBUGBRK
F05E:                  *
F05E:        0000                 EXTRN SYSERR
F05E:        0000                 EXTRN SERR
F05E:        0000                 EXTRN BADSCNUM
F05E:        0000                 EXTRN BADCZPAGE
F05E:        0000                 EXTRN BADXBYTE
F05E:        0000                 EXTRN BADSCPCNT
F05E:        0000                 EXTRN BADSCBNDS
F05E:                  *
F05E:        0000                 EXTRN SZPAGE
F05E:        0000                 EXTRN SXPAGE
F05E:        0000                 EXTRN CZPAGE
F05E:        0000                 EXTRN CXPAGE
F05E:        0000                 EXTRN CSPAGE
```

```
F05E:                  ************************************************************
F05E:                  *
F05E:                  * SYSTEM CALL PARAMETER DEFINITION TABLES
F05E:                  *
F05E:                  * EACH ENTRY IS FOUR BYTES LONG.  THE FIRST BYTE CONTAINS THE
F05E:                  * NUMBER OF PARMS IN THE CALL.  THE REMAINING SIX NIBBLES, EACH
F05E:                  * DEFINE A PARAMETER IN THE CALL.  THE FIRST BIT OF THE
F05E:                  * NIBBLE DEFINES WHETHER THE PARM IS INPUT (0) OR OUTPUT (1).
F05E:                  * THE NEXT BIT DEFINES WHETHER THE PARM IS BY VALUE (0)
F05E:                  * OR BY REFERENCE (1).  THE FINAL TWO BITS SPECIFY THE
F05E:                  * PARM LENGTH IN BYTES (E.G. 0=LENGTH OF 1, 3=LENGTH OF 4 BYTES)
F05E:                  *
F05E:                  ************************************************************
F05E:                  *
F05E:                  *   FILE SYSTEM CALL DEFINITIONS
F05E:                  *
F05E:        0013      FSC.CNT    EQU   $13
F05E:        F05E      FSC.TBL    EQU   *
F05E:03 5D 00 00                  DFB   $3,$5D,$00,$00   ; SCNUM=$C0 - CREATE
F062:01 50 00 00                  DFB   $1,$50,$00,$00   ;   "  =$C1 - DESTROY
F066:02 55 00 00                  DFB   $2,$55,$00,$00   ;   "  =$C2 - RENAME
F06A:03 5D 00 00                  DFB   $3,$5D,$00,$00   ;   "  =$C3 - SET.FILE.INFO
F06E:03 5D 00 00                  DFB   $3,$5D,$00,$00   ;   "  =$C4 - GET.FILE.INFO
F072:04 55 99 00                  DFB   $4,$55,$99,$00   ;   "  =$C5 - VOLUME
F076:01 50 00 00                  DFB   $1,$50,$00,$00   ;   "  =$C6 - SET.PREFIX
F07A:02 50 00 00                  DFB   $2,$50,$00,$00   ;   "  =$C7 - GET.PREFIX
F07E:04 58 D0 00                  DFB   $4,$58,$D0,$00   ;   "  =$C8 - OPEN
F082:03 00 00 00                  DFB   $3,$00,$00,$00   ;   "  =$C9 - NEW.LINE
F086:04 05 19 00                  DFB   $4,$05,$19,$00   ;   "  =$CA - READ
F08A:03 05 10 00                  DFB   $3,$05,$10,$00   ;   "  =$CB - WRITE
F08E:01 00 00 00                  DFB   $1,$00,$00,$00   ;   "  =$CC - CLOSE
F092:01 00 00 00                  DFB   $1,$00,$00,$00   ;   "  =$CD - FLUSH
F096:03 00 30 00                  DFB   $3,$00,$30,$00   ;   "  =$CE - SET.MARK
F09A:02 0B 00 00                  DFB   $2,$0B,$00,$00   ;   "  =$CF - GET.MARK
F09E:03 00 30 00                  DFB   $3,$00,$30,$00   ;   "  =$D0 - SET.EOF
F0A2:02 0B 00 00                  DFB   $2,$0B,$00,$00   ;   "  =$D1 - GET.EOF
F0A6:01 00 00 00                  DFB   $1,$00,$00,$00   ;   "  =$D2 - SET.LEVEL
F0AA:01 80 00 00                  DFB   $1,$80,$00,$00   ;   "  =$D3 - GET.LEVEL
```

```
F0AE:                 *
F0AE:                 *   DEVICE SYSTEM CALL DEFINITIONS
F0AE:                 *
F0AE:        0005     DSC.CNT    EQU   5
F0AE:        F0AE     DSC.TBL    EQU   *
F0AE:05 05 11 90                 DFB   $5,$05,$11,$90   ; SCNUM=$80 - D.READ
F0B2:04 05 11 00                 DFB   $4,$05,$11,$00   ;   "  =$81 - D.WRITE
F0B6:03 00 50 00                 DFB   $3,$00,$50,$00   ;   "  =$82 - D.STATUS
F0BA:03 00 50 00                 DFB   $3,$00,$50,$00   ;   "  =$83 - D.CONTROL
F0BE:02 58 00 00                 DFB   $2,$58,$00,$00   ;   "  =$84 - GET.DEV.NUM
F0C2:04 05 D0 00                 DFB   $4,$05,$D0,$00   ;   "  =$85 - D.INFO
F0C6:                 *
F0C6:                 *   UTILITY SYSTEM CALL DEFINITIONS
F0C6:                 *
F0C6:        0005     USC.CNT    EQU   5
F0C6:        F0C6     USC.TBL    EQU   *
F0C6:01 00 00 00                 DFB   $1,$00,$00,$00   ; SCNUM=$60 - SET.FENCE
F0CA:01 80 00 00                 DFB   $1,$80,$00,$00   ;   "  =$61 - GET.FENCE
F0CE:01 50 00 00                 DFB   $1,$50,$00,$00   ;   "  =$62 - SET.TIME
F0D2:01 50 00 00                 DFB   $1,$50,$00,$00   ;   "  =$63 - GET.TIME
F0D6:02 0B 00 00                 DFB   $2,$0B,$00,$00   ;   "  =$64 - JOYSTICK
F0DA:00 00 00 00                 DFB   $0,$00,$00,$00   ;   "  =$65 - COLD.START
F0DE:                 *
F0DE:                 *   MEMORY SYSTEM CALL DEFINITIONS
F0DE:                 *
F0DE:        0005     MSC.CNT    EQU   5
F0DE:        F0DE     MSC.TBL    EQU   *
F0DE:04 11 08 00                 DFB   $4,$11,$08,$00   ; SCNUM=$40 - REQUEST.SEG
F0E2:06 00 99 98                 DFB   $6,$00,$99,$98   ;   "  =$41 - FIND.SEG
F0E6:03 00 90 00                 DFB   $3,$00,$90,$00   ;   "  =$42 - CHANGE.SEG
F0EA:05 09 99 80                 DFB   $5,$09,$99,$80   ;   "  =$43 - GET.SEG.INFO
F0EE:02 18 00 00                 DFB   $2,$18,$00,$00   ;   "  =$44 - GET.SEG.NUM
F0F2:01 00 00 00                 DFB   $1,$00,$00,$00   ;   "  =$45 - RELEASE.SEG
F0F6:                 *
F0F6:                 *   DEBUG SYSTEM CALL DEFINITION
F0F6:                 *
F0F6:        00FE     DBUG       EQU   $FE
```

```
F0F6:              ************************************************************
F0F6:              *
F0F6:              * DATA DECLARATIONS
F0F6:              *
F0F6:              ************************************************************
F0F6:        FFD0  Z.REG      EQU   $FFD0
F0F6:        01FF  SP.SAVE    EQU   $01FF
F0F6:        01FD  Z.SAVE     EQU   $01FD
F0F6:        01FC  B.SAVE     EQU   $01FC
F0F6:              *
F0F6:        2000  ADR.LOW    EQU   $2000            ; LOW    ADDRESS   (BOUNDS CHECKING)
F0F6:        B800  ADR.HIGH   EQU   $B800            ; HIGH   ADDRESS
F0F6:        A000  ADR.MID    EQU   $A000            ; MIDDLE ADDRESS
F0F6:              *
F0F6:              * SCMGR'S VARIABLES
F0F6:              *
F0F6:        00E0  SCM.VARS   EQU   $E0
F0F6:        00E0  SCNUM      EQU   SCM.VARS+0       ; SYSTEM CALL NUMBER
F0F6:        00E0  SCRNUM     EQU   SCM.VARS+0       ; SYSTEM CALL REQUEST NUMBER
F0F6:        00E1  SCPTR      EQU   SCM.VARS+1       ;&2  SYSTEM CALL POINTER
F0F6:        00E3  MOVE.VARS  EQU   SCPTR+2          ; !! (LOOKOUT) !!
F0F6:              *
F0F6:              *
F0F6:        00A0  F.TPARMX   EQU   $A0              ; FILE SYS CALL PARM START LOC
F0F6:        00C0  D.TPARMX   EQU   $C0              ; DEVICE SYS CALL PARM START LOC
F0F6:        00C0  U.TPARMX   EQU   $C0              ; UTILITY SYS CALL PARM START LOC
F0F6:        0060  M.TPARMX   EQU   $60              ; MEMORY SYS CALL PARM START LOC
F0F6:              *
F0F6:              * MOVE.PARM'S VARIABLES
F0F6:              *
F0F6:        00E3  TPARMX     EQU   MOVE.VARS+0      ; TARGET ADR OF SYS CALL PARMS
F0F6:        00E4  DFN.PTR    EQU   MOVE.VARS+1      ;&2
F0F6:        00E6  DFN.PTRX   EQU   MOVE.VARS+3
F0F6:        00E7  SCPTRX     EQU   MOVE.VARS+4
F0F6:        00E8  RGHT.NIB   EQU   MOVE.VARS+5
F0F6:        00E9  SCT.DFN    EQU   MOVE.VARS+6
F0F6:        00EA  SCT.DCNT   EQU   MOVE.VARS+7
F0F6:        00EB  PARM.CNT   EQU   MOVE.VARS+8
```

```
F0F6:              ************************************************************
F0F6:              *
F0F6:              * SYSTEM CALL MANAGER
F0F6:              *
F0F6:              ************************************************************
F0F6:              *
F0F6:        F0F6  SCMGR      EQU   *
F0F6:A9 00                    LDA   #<SZPAGE         ; SET Z REG TO SOS ZPAGE
F0F8:8D D0 FF                 STA   Z.REG
F0FB:              *
F0FB:              * SET SYSTEM X BYTES TO ABSOLUTE ADDRESS MODE.
F0FB:              *
F0FB:A9 00                    LDA   #0
F0FD:8D E2 00                 STA   SXPAGE+SCPTR+1
F100:8D 00 00                 STA   SERR             ; AND INIT SYSTEM ERR CODE
F103:              *
F103:              * CALLER'S Z REG MUST BE $1A !!
F103:              * (B REG NOT CHECKED)
F103:              *
F103:AD FD 01                 LDA   Z.SAVE
F106:C9 00                    CMP   #<CZPAGE
F108:F0 05   F10F             BEQ   SCM005
F10A:A9 00                    LDA   #>BADCZPAGE
F10C:20 00 00                 JSR   SYSERR           ; EXIT TO DISPATCHER
F10F:              *
F10F:              * RETRIEVE CALLER'S PC ON HIS STACK
F10F:              *
F10F:AE FF 01      SCM005     LDX   SP.SAVE
F112:BD 06 00                 LDA   CSPAGE+6,X
F115:85 E2                    STA   SCPTR+1
F117:BD 05 00                 LDA   CSPAGE+5,X
F11A:85 E1                    STA   SCPTR
F11C:D0 02   F120             BNE   SCM010           ; AND POINT IT TO SYS CALL NUM
F11E:C6 E2                    DEC   SCPTR+1
F120:C6 E1         SCM010     DEC   SCPTR
F122:              *
F122:              * ADVANCE CALLER'S PC ON HIS STACK.
F122:              *
F122:18                       CLC
F123:BD 05 00                 LDA   CSPAGE+5,X
F126:69 02                    ADC   #2
F128:9D 05 00                 STA   CSPAGE+5,X
F12B:90 03   F130             BCC   SCM020
F12D:FE 06 00                 INC   CSPAGE+6,X
F130:              *
F130:              * RETRIEVE SYSTEM CALL NUMBER
F130:              *
F130:A0 00         SCM020     LDY   #0
F132:B1 E1                    LDA   (SCPTR),Y
F134:C9 FE                    CMP   #DBUG
F136:D0 03   F13B             BNE   SCM025
F138:20 00 00                 JSR   DBUGBRK          ; DEBUG SYSTEM CALL
F13B:85 E0         SCM025     STA   SCNUM
F13D:              *
F13D:              * RETRIEVE SYSTEM CALL PARAMETER ADDRESS
F13D:              *
```

```
F13D:C8                       INY
F13E:A2 E1                    LDX   #>SCPTR
F140:20 64 F2                 JSR   POINTER
F143:90 01   F146             BCC   SCM030
F145:60                       RTS                    ; ERROR EXIT
F146:              *
F146:              * CASE INTERFACE CODE OF SYSTEM CALL NUMBER
F146:              *  (INTERFACE CODE STRIPPED, LEAVING REQUEST CODE)
F146:              *
F146:A9 20         SCM030     LDA   #$20
F148:24 E0                    BIT   SCNUM
F14A:10 2C   F178             BPL   SCM050
F14C:A5 E0                    LDA   SCNUM
F14E:29 3F                    AND   #$3F
F150:85 E0                    STA   SCRNUM
F152:50 12   F166             BVC   SCM040
F154:              *
F154:A9 A0                    LDA   #F.TPARMX        ; "11XXXXXX" - JMP TO FILE MANAGER.
F156:85 E3                    STA   TPARMX
F158:A2 5E                    LDX   #>FSC.TBL
F15A:A0 F0                    LDY   #<FSC.TBL
F15C:A9 13                    LDA   #FSC.CNT
F15E:20 AD F1                 JSR   MOVE.PARMS
F161:B0 47   F1AA             BCS   SCM.ERR1         ; ERR EXIT
F163:4C 00 00                 JMP   FMGR
F166:              *
F166:A9 C0         SCM040     LDA   #D.TPARMX        ; "10XXXXXX" - JMP TO DEVICE MANAGER.
F168:85 E3                    STA   TPARMX
F16A:A2 AE                    LDX   #>DSC.TBL
F16C:A0 F0                    LDY   #<DSC.TBL
F16E:A9 05                    LDA   #DSC.CNT
F170:20 AD F1                 JSR   MOVE.PARMS
F173:B0 35   F1AA             BCS   SCM.ERR1         ; ERR EXIT
F175:4C 00 00                 JMP   DMGR
F178:              *
F178:50 2E   F1A8  SCM050     BVC   SCM.ERR
F17A:08                       PHP
F17B:A5 E0                    LDA   SCNUM
F17D:29 1F                    AND   #$1F
F17F:85 E0                    STA   SCRNUM
F181:28                       PLP
F182:F0 12   F196             BEQ   SCM060
F184:              *
F184:A9 C0                    LDA   #U.TPARMX        ; "011XXXXX" - JMP TO UTILITY MANAGER.
F186:85 E3                    STA   TPARMX
F188:A2 C6                    LDX   #>USC.TBL
F18A:A0 F0                    LDY   #<USC.TBL
F18C:A9 05                    LDA   #USC.CNT
F18E:20 AD F1                 JSR   MOVE.PARMS
F191:B0 17   F1AA             BCS   SCM.ERR1         ; ERR EXIT
F193:4C 00 00                 JMP   UMGR
F196:              *
F196:A9 60         SCM060     LDA   #M.TPARMX        ; "010XXXXX" - JMP TO MEMORY MANAGER.
F198:85 E3                    STA   TPARMX
F19A:A2 DE                    LDX   #>MSC.TBL
F19C:A0 F0                    LDY   #<MSC.TBL
```

```
F19E:A9 05                     LDA   #MSC.CNT
F1A0:20 AD F1                  JSR   MOVE.PARMS
F1A3:B0 05   F1AA              BCS   SCM.ERR1         ; ERR EXIT
F1A5:4C 00 00                  JMP   MMGR
F1A8:                *
F1A8:A9 00           SCM.ERR    LDA   #>BADSCNUM       ; ERROR, INVALID SYSTEM CALL NUMBER.
F1AA:20 00 00        SCM.ERR1   JSR   SYSERR           ;   EXIT TO DISPATCHER ON ERROR
```

```
F1AD:              ************************************************************
F1AD:              *
F1AD:              * MOVE.PARMS
F1AD:              *
F1AD:              * MOVES THE CALLER'S PARAMETERS TO THE OPERATING SYSTEM'S
F1AD:              * ZERO PAGE, ACCORDING TO THE SPECIFICATIONS CONTAINED
F1AD:              * IN THE SPECIFIED SYS CALL DFN TABLE.
F1AD:              *
F1AD:              * INPUT: (A) = MAX # ENTRIES IN PARM DFN TABLE
F1AD:              *        (X) = PARM DFN TBL ADR (LO)
F1AD:              *        (Y) =         "        (HI)
F1AD:              *      SCPTR = ADR OF CALLER'S SYS CALL PARMS
F1AD:              * ERROR: CARRY SET (SYSERR)
F1AD:              *
F1AD:              ************************************************************
F1AD:              *
F1AD:        F1AD  MOVE.PARMS EQU   *
F1AD:86 E4                    STX   DFN.PTR          ; SAVE ADR OF DEFINITION TABLE
F1AF:84 E5                    STY   DFN.PTR+1
F1B1:              *
F1B1:              * IF REQ NUM > MAX REQ NUM (A REG)
F1B1:              *
F1B1:C5 E0                    CMP   SCRNUM
F1B3:B0 04   F1B9             BCS   MOVE010
F1B5:              *
F1B5:              *   THEN ERR(BAD SYS CALL NUM)
F1B5:              *
F1B5:A9 00                    LDA   #>BADSCNUM
F1B7:90 18   F1D1             BCC   SYSERR1          ;BRANCH ALWAYS TAKEN
F1B9:              *
F1B9:              * CALCULATE DEFINITION TABLE INDEX
F1B9:              *  AND INIT SYS CALL PARM INDEX
F1B9:              *
F1B9:A5 E0         MOVE010    LDA   SCRNUM
F1BB:0A                       ASL   A
F1BC:0A                       ASL   A
F1BD:85 E6                    STA   DFN.PTRX
F1BF:A9 00                    LDA   #0
F1C1:8D E5 00                 STA   SXPAGE+DFN.PTR+1 ; AND X BYTE
F1C4:85 E7                    STA   SCPTRX
F1C6:              *
F1C6:              * IF SCPTR(SCPTRX)<>DFN.PTR(DFN.PTRX) THEN ERR
F1C6:              *
F1C6:A8                       TAY
F1C7:B1 E1                    LDA   (SCPTR),Y
F1C9:A4 E6                    LDY   DFN.PTRX
F1CB:D1 E4                    CMP   (DFN.PTR),Y
F1CD:F0 05   F1D4             BEQ   INITLOOPCT
F1CF:              *
F1CF:A9 00                    LDA   #>BADSCPCNT      ; ERR, CALLER'S PARM COUNT INVALID
F1D1:20 00 00      SYSERR1    JSR   SYSERR           ; EXIT
F1D4:              *
F1D4:              * INIT LOOP CTR(PARM.CNT) TO # OF PARMS IN SYS CALL
F1D4:              *
F1D4:85 EB         INITLOOPCT STA   PARM.CNT
F1D6:              *
```

```
F1D6:              * ADVANCE PTRS
F1D6:              *
F1D6:              *
F1D6:E6 E7                    INC   SCPTRX
F1D8:E6 E6                    INC   DFN.PTRX
F1DA:              *
F1DA:              * MOVE REQ CODE TO SYS ZPAGE PARM LIST
F1DA:              *  AND ADVANCE SYS ZPAGE PTR (X=TPARMX)
F1DA:              *
F1DA:A5 E0                    LDA   SCRNUM
F1DC:A6 E3                    LDX   TPARMX
F1DE:95 00                    STA   0,X
F1E0:E8                       INX
F1E1:              *
F1E1:              * INIT NIBBLE FLAG TO "RIGHT" NIBBLE
F1E1:              *  ZERO STATE="LEFT" NIBBLE
F1E1:              *
F1E1:A9 FF                    LDA   #$FF
F1E3:85 E8                    STA   RGHT.NIB
F1E5:              ************************************************************
F1E5:              *
F1E5:              *  BEGIN PARAMETER PROCESSING LOOP
F1E5:              *
F1E5:A5 E8         PARMLOOP   LDA   RGHT.NIB
F1E7:49 FF                    EOR   #$FF             ; COMPLEMENT NIBBLE FLAG
F1E9:85 E8                    STA   RGHT.NIB
F1EB:              *
F1EB:              * IF "LEFT" NIBBLE
F1EB:              *
F1EB:D0 10   F1FD             BNE   ELSE.RNIB
F1ED:              *
F1ED:              * THEN FETCH SYS CALL PARM DFN
F1ED:              *  AND # OF BYTES IN PARM WITHIN IT
F1ED:              *
F1ED:A4 E6                    LDY   DFN.PTRX
F1EF:B1 E4                    LDA   (DFN.PTR),Y
F1F1:85 E9                    STA   SCT.DFN
F1F3:29 30                    AND   #$30
F1F5:4A                       LSR   A
F1F6:4A                       LSR   A
F1F7:4A                       LSR   A
F1F8:4A                       LSR   A
F1F9:85 EA                    STA   SCT.DCNT
F1FB:10 10   F20D             BPL   VALUE            ;BRANCH ALWAYS
F1FD:              *
F1FD:              * ELSE FETCH SYS CALL PARM DFN
F1FD:              *  AND # OF BYTES IN PARM WITHIN IT
F1FD:              *  FROM "RIGHT" NIBBLE OF DFN BYTE
F1FD:              *
F1FD:A5 E9         ELSE.RNIB  LDA   SCT.DFN
F1FF:A8                       TAY
F200:29 03                    AND   #$03
F202:85 EA                    STA   SCT.DCNT
F204:98                       TYA
F205:0A                       ASL   A
F206:0A                       ASL   A
```

```
F207:0A                       ASL   A
F208:0A                       ASL   A
F209:85 E9                    STA   SCT.DFN
F20B:E6 E6                    INC   DFN.PTRX         ; ADVANCE SYS CALL DFN PTR
F20D:              ************************************************************
F20D:              *
F20D:              *  PARAMETER PASSED BY VALUE
F20D:              *
F20D:              ************************************************************
F20D:24 E9         VALUE      BIT   SCT.DFN
F20F:70 31   F242             BVS   REFERENCE
F211:30 11   F224             BMI   VAL.OUT
F213:              *
F213:              *  INPUT BY VALUE
F213:              *
F213:A4 E7                    LDY   SCPTRX           ; MOVE BYTES TO ZPAGE
F215:B1 E1         VAL.IN     LDA   (SCPTR),Y
F217:95 00                    STA   0,X
F219:C8                       INY
F21A:E8                       INX
F21B:C6 EA                    DEC   SCT.DCNT
F21D:10 F6   F215             BPL   VAL.IN
F21F:84 E7                    STY   SCPTRX
F221:4C 59 F2                 JMP   ENDLOOP1
F224:              *
F224:              *  OUTPUT BY VALUE
F224:              *
F224:18            VAL.OUT    CLC                    ; BUILD PTR TO PARM ON ZPAGE
F225:A5 E1                    LDA   SCPTR
F227:65 E7                    ADC   SCPTRX
F229:95 00                    STA   0,X
F22B:E8                       INX
F22C:A5 E2                    LDA   SCPTR+1
F22E:69 00                    ADC   #0
F230:95 00                    STA   0,X
F232:              *
F232:18                       CLC                    ; ADVANCE INDEX TO NEXT PARM
F233:A5 E7                    LDA   SCPTRX
F235:65 EA                    ADC   SCT.DCNT
F237:85 E7                    STA   SCPTRX
F239:              *
F239:AD E2 00                 LDA   SXPAGE+SCPTR+1   ; INCLUDE X BYTE
F23C:9D 00 00                 STA   SXPAGE,X
F23F:4C 56 F2                 JMP   ENDLOOP2
F242:              ************************************************************
F242:              *
F242:              *  PARAMETER PASSED BY REFERENCE
F242:              *
F242:              ************************************************************
F242:10 08   F24C  REFERENCE  BPL   REF1
F244:              *
F244:              * "LIST" PTR FOUND, CHK IF "LENGTH" PARM = 0
F244:              *
F244:A4 E7                    LDY   SCPTRX
F246:C8                       INY
F247:C8                       INY
```

```
F248:B1 E1                    LDA   (SCPTR),Y
F24A:F0 07   F253             BEQ   ENDLOOP0         ; "LENGTH" PARM=0, SKIP "LIST" PARM
F24C:              *
F24C:A4 E7         REF1       LDY   SCPTRX           ; MOVE PTR TO ZPAGE
F24E:20 64 F2                 JSR   POINTER
F251:B0 10   F263             BCS   PARM.ERR         ; ERROR EXIT
F253:              *
F253:              * ADVANCE SYSTEM ZPAGE POINTER (X), CALLER'S PARM PTR.
F253:              * DECREMENT PARM CTR AND CHECK IF LAST PARM PROCESSED.
F253:              *
F253:E8            ENDLOOP0   INX
F254:E6 E7                    INC   SCPTRX
F256:E8            ENDLOOP2   INX
F257:E6 E7                    INC   SCPTRX
F259:C6 EB         ENDLOOP1   DEC   PARM.CNT
F25B:F0 05   F262             BEQ   PARM.EXIT
F25D:30 03   F262             BMI   PARM.EXIT        ;SPECIAL FOR 'COLD START'
F25F:4C E5 F1                 JMP   PARMLOOP
F262:              *
F262:              *  END OF PARAMETER PROCESSING LOOP
F262:              *
F262:              ************************************************************
F262:              *
F262:18            PARM.EXIT  CLC                    ; NO ERRORS
F263:60            PARM.ERR   RTS                    ; RETURN TO SYS CALL MANAGER
```

```
F264:              ************************************************************
F264:              *
F264:              * POINTER
F264:              *
F264:              * INPUT:   SRC ADR   (SCPTR),Y & (SCPTR),Y+1
F264:              *          DEST ADR  (X)
F264:              *
F264:              * OUTPUT:  SCPTR     UNCHANGED
F264:              *          X REG         "
F264:              *          A,Y REGS  FLATTENED
F264:              *
F264:              * ERROR:   CARRY SET (SYSERR)
F264:              *
F264:              * POINTER.  RETRIEVES THE CALLER'S POINTER PARAMETER IN
F264:              * (SCPTR),Y, PERFORMS ADDRESS COMPENSATION, IF NECESSARY
F264:              * AND PLACES THE RESULTING POINTER AT X, X+1 AND SXPAGE+1,X.
F264:              *
F264:              ************************************************************
F264:              *
F264:        F264  POINTER    EQU   *
F264:B1 E1                    LDA   (SCPTR),Y
F266:48                       PHA
F267:C8                       INY
F268:B1 E1                    LDA   (SCPTR),Y
F26A:F0 09   F275             BEQ   INDIRECT
F26C:              *
F26C:95 01                    STA   1,X              ; DIRECT POINTER
F26E:68                       PLA
F26F:95 00                    STA   0,X
F271:A0 00                    LDY   #0
F273:F0 10   F285             BEQ   PTR010
F275:              *
F275:68            INDIRECT   PLA                    ; INDIRECT POINTER
F276:A8                       TAY
F277:B9 00 00                 LDA   CZPAGE,Y
F27A:95 00                    STA   0,X
F27C:B9 01 00                 LDA   CZPAGE+1,Y
F27F:95 01                    STA   1,X
F281:B9 01 00                 LDA   CXPAGE+1,Y
F284:A8                       TAY
F285:              *
F285:B5 01         PTR010     LDA   1,X
F287:              *
F287:              * CHECK BOUNDS OF CALLER'S POINTER PARAMETER
F287:              *
F287:C0 8F                    CPY   #$8F
F289:90 0E   F299             BCC   PTR.X808E
F28B:F0 02   F28F             BEQ   PTR.X8F
F28D:B0 60   F2EF             BCS   PTR.ERR1         ; ERROR, INVALID X BYTE
F28F:C9 20         PTR.X8F    CMP   #<ADR.LOW
F291:90 57   F2EA             BCC   PTR.ERR
F293:C9 B8                    CMP   #<ADR.HIGH
F295:B0 53   F2EA             BCS   PTR.ERR
F297:90 49   F2E2             BCC   PTR.EXIT
F299:              *
F299:              * X BYTE = 80..8E
```

```
F299:              *
F299:C0 80         PTR.X808E  CPY   #$80
F29B:90 0D   F2AA             BCC   PTR.X0
F29D:C9 00                    CMP   #0
F29F:F0 49   F2EA             BEQ   PTR.ERR
F2A1:C9 FF                    CMP   #$FF
F2A3:D0 2E   F2D3             BNE   PATCH
F2A5:C8                       INY                    ; $8N:FFXX --> $8N+1:7FXX
F2A6:A9 7F                    LDA   #$7F
F2A8:D0 38   F2E2             BNE   PTR.EXIT
F2AA:              *
F2AA:              * X BYTE = 0
F2AA:              *
F2AA:C0 00         PTR.X0     CPY   #0
F2AC:D0 41   F2EF             BNE   PTR.ERR1
F2AE:C9 20                    CMP   #<ADR.LOW
F2B0:90 38   F2EA             BCC   PTR.ERR
F2B2:C9 B8                    CMP   #<ADR.HIGH
F2B4:B0 34   F2EA             BCS   PTR.ERR
F2B6:C9 A0                    CMP   #<ADR.MID
F2B8:B0 28   F2E2             BCS   PTR.EXIT
F2BA:              *
F2BA:48                       PHA
F2BB:AD FC 01                 LDA   B.SAVE
F2BE:29 0F                    AND   #$0F
F2C0:D0 05   F2C7             BNE   PTR030
F2C2:68                       PLA                    ; $B=0:2000..9FFF --> $8F:2000.9FFF
F2C3:A0 8F                    LDY   #$8F
F2C5:D0 1B   F2E2             BNE   PTR.EXIT
F2C7:              *
F2C7:09 80         PTR030     ORA   #$80             ; $B<>0:2000..9FFF --> $8B:0000..7FFF
F2C9:A8                       TAY
F2CA:68                       PLA
F2CB:38                       SEC
F2CC:E9 20                    SBC   #$20
F2CE:D0 03   F2D3             BNE   PATCH
F2D0:88                       DEY                    ; $8B:00XX --> $8B-1:80XX
F2D1:A9 80                    LDA   #$80
F2D3:              *
F2D3:C0 80         PATCH      CPY   #$80             ; KLUDGE FOR BFM:  $8N:01XX --> $8N-1:81XX
F2D5:90 0B   F2E2             BCC   PTR.EXIT
F2D7:C9 01                    CMP   #1
F2D9:D0 07   F2E2             BNE   PTR.EXIT
F2DB:C0 80                    CPY   #$80
F2DD:F0 0B   F2EA             BEQ   PTR.ERR          ; ERROR, $80:01XX NOT ALLOWED
F2DF:88                       DEY
F2E0:A9 81                    LDA   #$81
F2E2:              *
F2E2:95 01         PTR.EXIT   STA   1,X
F2E4:98                       TYA
F2E5:9D 01 00                 STA   SXPAGE+1,X
F2E8:18                       CLC
F2E9:60                       RTS
F2EA:              *
F2EA:              *
F2EA:A9 00         PTR.ERR    LDA   #>BADSCBNDS
```

```
F2EC:20 00 00      588            JSR   SYSERR
F2EF:A9 00         589 PTR.ERR1   LDA   #>BADXBYTE
F2F1:20 00 00      590            JSR   SYSERR
F2F4:              591 *

F2F4:              592            LST   ON
F2F4:        F2F4  593 ZZEND      EQU   *
F2F4:        0296  594 ZZLEN      EQU   ZZEND-ZZORG
F2F4:        0000  595            IFNE  ZZLEN-LENSCMGR
 S                 596            FAIL  2,"SOSORG        FILE IS INCORRECT FOR SCMGR"
F2F4:              597            FIN
```

```
 B800 ADR.HIGH       2000 ADR.LOW        A000 ADR.MID        01FC B.SAVE
X000A BADCZPAGE     X000D BADSCBNDS     X0009 BADSCNUM      X000C BADSCPCNT
X000B BADXBYTE       3200 BLABFM        ?2E00 BLABFMI        6B52 BLABUFMG
 6955 BLACFM         5E99 BLADISK3       64D9 BLADMGR        68F4 BLAFMGR
?2CF8 BLAGLOB       ?2AF8 BLAINIT        55C0 BLAIPL         2000 BLALODR
?6E6E BLAMEMMG       5466 BLAOMSG        5466 BLAPATCH       665E BLASCMGR
 6404 BLASERR        5A8B BLAUMGR       X0012 CSPAGE        X0011 CXPAGE
X0010 CZPAGE           C0 D.TPARMX         FE DBUG          X0006 DBUGBRK
   E4 DFN.PTR          E6 DFN.PTRX      X0003 DMGR            05 DSC.CNT
 F0AE DSC.TBL        F1FD ELSE.RNIB      F253 ENDLOOP0       F259 ENDLOOP1
 F256 ENDLOOP2         A0 F.TPARMX      X0002 FMGR            13 FSC.CNT
 F05E FSC.TBL        F275 INDIRECT       F1D4 INITLOOPCT     2266 LENBFM
?0400 LENBFMI        031C LENBUFMG       01FD LENCFM         056B LENDISK3
 0185 LENDMGR          61 LENFMGR       ?01B2 LENINIT        04CB LENIPL
 0AF8 LENLODR       ?0751 LENMEMMG       015A LENOMSG          00 LENPATCH
 0296 LENSCMGR         D5 LENSERR        040E LENUMGR          60 M.TPARMX
X0005 MMGR           F1AD MOVE.PARMS       E3 MOVE.VARS      F1B9 MOVE010
   05 MSC.CNT        F0DE MSC.TBL        BC00 ORGBFM         B800 ORGBFMI
 F552 ORGBUFMG       F355 ORGCFM         E899 ORGDISK3       EED9 ORGDMGR
 FFBF ORGEND         F2F4 ORGFMGR       ?18FC ORGGLOB        28F8 ORGINIT
 DFC0 ORGIPL         1E00 ORGLODR        F86E ORGMEMMG       DE66 ORGOMSG
 DE66 ORGPATCH       F05E ORGSCMGR       EE04 ORGSERR        E48B ORGUMGR
   EB PARM.CNT       F263 PARM.ERR       F262 PARM.EXIT      F1E5 PARMLOOP
 F2D3 PATCH          F264 POINTER        F2EA PTR.ERR        F2EF PTR.ERR1
 F2E2 PTR.EXIT       F2AA PTR.X0         F299 PTR.X808E      F28F PTR.X8F
 F285 PTR010         F2C7 PTR030         F24C REF1           F242 REFERENCE
   E8 RGHT.NIB       F1A8 SCM.ERR        F1AA SCM.ERR1         E0 SCM.VARS
 F10F SCM005         F120 SCM010         F130 SCM020         F13B SCM025
 F146 SCM030         F166 SCM040         F178 SCM050         F196 SCM060
NF0F6 SCMGR            E0 SCNUM            E7 SCPTRX           E1 SCPTR
   E0 SCRNUM           EA SCT.DCNT         E9 SCT.DFN       X0008 SERR
 01FF SP.SAVE       X000F SXPAGE        X0007 SYSERR         F1D1 SYSERR1
X000E SZPAGE           E3 TPARMX           C0 U.TPARMX      X0004 UMGR
   05 USC.CNT        F0C6 USC.TBL        F215 VAL.IN         F224 VAL.OUT
 F20D VALUE          FFD0 Z.REG          01FD Z.SAVE         F2F4 ZZEND
 0296 ZZLEN          F05E ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   656
** FREE SPACE PAGE COUNT   82
```

```
SOURCE   FILE #01 =>FMGR.SRC
 INCLUDE FILE #02 =>SOSORG
```

```
0000:                             REL
0000:                             INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955   BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52   BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E   BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
F2F4:        F2F4                ORG   ORGFMGR
F2F4:        F2F4   ZZORG      EQU   *
F2F4:                            MSB   OFF
F2F4:               ************************************************************
F2F4:               *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
F2F4:               *                    ALL RIGHTS RESERVED
F2F4:               ************************************************************
F2F4:               *
F2F4:               * FILE MANAGER (VERSION = 1.1O   )
F2F4:               *              (DATE    = 8/04/81)
F2F4:               *
F2F4:               * THIS MODULE IS ENTERED FROM THE SYSTEM CALL MANAGER, AND
F2F4:               * IS RESPONSIBLE FOR SWITCHING TO EITHER THE BLOCK FILE
F2F4:               * MANAGER, OR THE CHARACTER FILE MANAGER.
F2F4:               *
F2F4:               ************************************************************
F2F4:               *
F2F4:        F2F5                ENTRY FMGR
F2F4:        F2F4                ENTRY LEVEL
F2F4:               *
F2F4:        0000                EXTRN BFMGR
F2F4:        0000                EXTRN CFMGR
F2F4:        0000                EXTRN SYSERR
F2F4:        0000                EXTRN SERR
F2F4:        0000                EXTRN BADPATH
F2F4:        0000                EXTRN FNFERR
F2F4:        0000                EXTRN LVLERR
F2F4:               *
F2F4:        00A0   F.TPARMX   EQU   $A0              ; LOC OF FILE SYSTEM CALL PARMS
F2F4:        0008   OPEN       EQU   $8
F2F4:        000C   CLOSE      EQU   $C
F2F4:        0012   SETLEVEL   EQU   $12
F2F4:        0013   GETLEVEL   EQU   $13
F2F4:        00A0   F.REQCODE  EQU   F.TPARMX
F2F4:        00A1   F.LEVEL    EQU   F.TPARMX+$1
F2F4:        00A1   PATHNAME   EQU   F.TPARMX+$1
F2F4:        00A1   REFNUM     EQU   F.TPARMX+$1
F2F4:        002E   PERIOD     EQU   $2E
F2F4:01             LEVEL      DFB   $1
```

```
F2F5:               ************************************************************
F2F5:               *
F2F5:               * FILE MANAGER
F2F5:               *
F2F5:               ************************************************************
F2F5:        F2F5   FMGR       EQU   *
F2F5:               *
F2F5:A5 A0                     LDA   F.REQCODE
F2F7:C9 08                     CMP   #OPEN
F2F9:90 10   F30B              BCC   FMGR010
F2FB:F0 11   F30E              BEQ   FMGR020
F2FD:C9 0C                     CMP   #CLOSE
F2FF:90 2A   F32B              BCC   FMGR030
F301:F0 2F   F332              BEQ   FMGR040
F303:C9 12                     CMP   #SETLEVEL
F305:F0 35   F33C              BEQ   SLEVEL
F307:C9 13                     CMP   #GETLEVEL
F309:F0 42   F34D              BEQ   GLEVEL
F30B:               *
F30B:4C 00 00       FMGR010    JMP   BFMGR            ; EXIT
F30E:               *
F30E:A0 01          FMGR020    LDY   #1
F310:B1 A1                     LDA   (PATHNAME),Y
F312:C9 2E                     CMP   #PERIOD
F314:D0 F5   F30B              BNE   FMGR010
F316:20 00 00                  JSR   CFMGR
F319:90 07   F322              BCC   FMGR024
F31B:AD 00 00                  LDA   SERR
F31E:C9 00                     CMP   #FNFERR
F320:F0 01   F323              BEQ   FMGR026
F322:60             FMGR024    RTS                    ; EXIT
F323:               *
F323:A9 00          FMGR026    LDA   #0
F325:8D 00 00                  STA   SERR
F328:4C 00 00                  JMP   BFMGR            ; EXIT
F32B:               *
F32B:A5 A1          FMGR030    LDA   REFNUM
F32D:10 DC   F30B   FMGR031    BPL   FMGR010
F32F:4C 00 00                  JMP   CFMGR            ; EXIT
F332:               *
F332:A5 A1          FMGR040    LDA   REFNUM
F334:D0 F7   F32D              BNE   FMGR031
F336:20 00 00                  JSR   BFMGR            ; CLOSE (0)
F339:4C 00 00                  JMP   CFMGR            ; EXIT
F33C:               *
F33C:A5 A1          SLEVEL     LDA   F.LEVEL
F33E:F0 08   F348              BEQ   LVL.ERR
F340:C9 04                     CMP   #4
F342:B0 04   F348              BCS   LVL.ERR
F344:8D F4 F2                  STA   LEVEL
F347:60                        RTS
F348:A9 00          LVL.ERR    LDA   #LVLERR
F34A:20 00 00                  JSR   SYSERR
F34D:               *
F34D:A0 00          GLEVEL     LDY   #0
F34F:AD F4 F2                  LDA   LEVEL
```

```
F352:91 A1                    STA   (F.LEVEL),Y
F354:60                       RTS
F355:              *

F355:                         LST   ON
F355:        F355  ZZEND      EQU   *
F355:        0061  ZZLEN      EQU   ZZEND-ZZORG
F355:        0000             IFNE  ZZLEN-LENFMGR
 S                            FAIL  2,"SOSORG        FILE IS INCORRECT FOR FMGR"
F355:                         FIN
```

```
?0007 BADPATH       X0003 BFMGR          3200 BLABFM        ?2E00 BLABFMI
 6B52 BLABUFMG       6955 BLACFM         5E99 BLADISK3       64D9 BLADMGR
 68F4 BLAFMGR       ?2CF8 BLAGLOB       ?2AF8 BLAINIT        55C0 BLAIPL
 2000 BLALODR       ?6E6E BLAMEMMG       5466 BLAOMSG        5466 BLAPATCH
 665E BLASCMGR       6404 BLASERR        5A8B BLAUMGR       X0004 CFMGR
   0C CLOSE            A1 F.LEVEL          A0 F.REQCODE        A0 F.TPARMX
NF2F5 FMGR           F30B FMGR010        F30E FMGR020        F322 FMGR024
 F323 FMGR026        F32B FMGR030        F32D FMGR031        F332 FMGR040
X0008 FNFERR           13 GETLEVEL       F34D GLEVEL        ?0400 LENBFMI
 2266 LENBFM         031C LENBUFMG       01FD LENCFM         056B LENDISK3
 0185 LENDMGR          61 LENFMGR       ?01B2 LENINIT        04CB LENIPL
 0AF8 LENLODR       ?0751 LENMEMMG       015A LENOMSG          00 LENPATCH
 0296 LENSCMGR         D5 LENSERR        040E LENUMGR       NF2F4 LEVEL
 F348 LVL.ERR       X0009 LVLERR           08 OPEN           BC00 ORGBFM
 B800 ORGBFMI        F552 ORGBUFMG       F355 ORGCFM         E899 ORGDISK3
 EED9 ORGDMGR        FFBF ORGEND         F2F4 ORGFMGR       ?18FC ORGGLOB
 28F8 ORGINIT        DFC0 ORGIPL         1E00 ORGLODR        F86E ORGMEMMG
 DE66 ORGOMSG        DE66 ORGPATCH       F05E ORGSCMGR       EE04 ORGSERR
 E48B ORGUMGR          A1 PATHNAME         2E PERIOD           A1 REFNUM
X0006 SERR             12 SETLEVEL       F33C SLEVEL        X0005 SYSERR
 F355 ZZEND            61 ZZLEN          F2F4 ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   167
** FREE SPACE PAGE COUNT   85
```

```
SOURCE   FILE #01 =>CFMGR.SRC
 INCLUDE FILE #02 =>SOSORG
```

```
0000:                             REL
0000:                             INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955   55 BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52   56 BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E   57 BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:               58
****************************************************************************************************
F355:        F355    4            ORG   ORGCFM
F355:        F355    5 ZZORG      EQU   *
F355:                6            MSB   OFF
F355:                7 ************************************************************
F355:                8 *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
F355:                9 *                    ALL RIGHTS RESERVED
F355:               10 ************************************************************
F355:               11 *
F355:               12 * CHARACTER FILE MANAGER (VERSION = 1.1O   )
F355:               13 *                        (DATE    = 8/04/81)
F355:               14 *
F355:               15 * THIS MODULE TRANSFORMS CHARACTER FILE SYSTEM CALLS INTO
F355:               16 * DEVICE CALLS TO THE APPROPRIATE DEVICE HANDLER.  ONLY
F355:               17 * OPEN, NEWLINE, READ, WRITE AND CLOSE CALLS ARE PERMITTED
F355:               18 * ON CHARACTER FILES.
F355:               19 *
F355:               20 ************************************************************
F355:               21 *
F355:        F37A   22            ENTRY CFMGR
F355:               23 *
F355:        0011   24            ENTRY CFCB.MAX
F355:        F358   25            ENTRY CFCB.DEV
F355:               26 *
F355:        0000   27            EXTRN DMGR
F355:        0000   28            EXTRN LEVEL
F355:        0000   29            EXTRN MAX.DNUM
F355:        0000   30            EXTRN SXPAGE
F355:               31 *
F355:        0000   32            EXTRN SYSERR
F355:        0000   33            EXTRN SERR
F355:        0000   34            EXTRN BADSCNUM
F355:        0000   35            EXTRN CFCBFULL
F355:        0000   36            EXTRN BADREFNUM
F355:        0000   37            EXTRN FNFERR
```

```
F355:                 ************************************************************
F355:                 *
F355:                 * DATA DECLARATIONS
F355:                 *
F355:                 ************************************************************
F355:                 *
F355:                 * FILE CALL PARM LOCATIONS ON SOS ZPAGE
F355:                 *
F355:         00A0    F.TPARMX   EQU   $A0
F355:         00A0    REQCODE    EQU   F.TPARMX
F355:         00A1    O.PATH     EQU   F.TPARMX+1       ; OPEN'S PATHNAME LOC
F355:         00A3    O.REFNUM   EQU   F.TPARMX+3       ; OPEN'S REFNUM LOC
F355:         00A1    REFNUM     EQU   F.TPARMX+1       ; REFNUM'S LOC IN OTHER CALLS
F355:         00A2    NL.ISNL    EQU   F.TPARMX+2       ; NEWLINE'S ISNEWLINE LOC
F355:         00A3    NL.NLCHR   EQU   F.TPARMX+3       ; NEWLINE'S NEWLINECHAR LOC
F355:         00A2    RW.BUF     EQU   F.TPARMX+2       ; READ/WRITE'S BUF LOC
F355:         00A4    RW.BYTES   EQU   F.TPARMX+4       ; READ/WRITE'S BYTES LOC
F355:         00A6    RD.BYTESRD EQU   F.TPARMX+6       ; READ'S BYTESREAD LOC
F355:                 *
F355:                 * FILE REQUEST CODE VALUES
F355:                 *
F355:         0008    OPEN       EQU   8
F355:         0009    NEWLINE    EQU   9
F355:         000A    READ       EQU   $A
F355:         000B    WRITE      EQU   $B
F355:         000C    CLOSE      EQU   $C
```

```
F355:               * DEVICE CALL PARM LOCATIONS ON SOS ZPAGE
F355:               *
F355:        00C0   D.TPARMX   EQU   $C0
F355:        00C0   D.SCNUM    EQU   D.TPARMX         ; DEVICE SYS CALL # LOC
F355:        00C1   GDN.DNAME  EQU   D.TPARMX+1       ; GETDEVNUM DNAME LOC
F355:        00C3   GDN.DNUM   EQU   D.TPARMX+3       ; GETDEVNUM DNUM LOC
F355:        00C1   D.DNUM     EQU   D.TPARMX+1       ; OPN/CLOSE/RD/WR/CTRL'S DNUM LOC
F355:        00C2   DRW.BUF    EQU   D.TPARMX+2       ; RD/WR'S BUF LOC
F355:        00C4   DRW.BYTES  EQU   D.TPARMX+4       ; RD/WR'S BYTES LOC
F355:        00C8   DRD.BYTESRD EQU  D.TPARMX+8       ; RD/WR'S BYTESREAD LOC
F355:        00C2   DC.CCODE   EQU   D.TPARMX+2       ; DCTRL'S CTRLCODE LOC
F355:        00C3   DC.CLIST   EQU   D.TPARMX+3       ; DCTRL'S CTRLLIST LOC
F355:               *
F355:               * DEVICE REQUEST CODE VALUES
F355:               *
F355:        0000   DREAD      EQU   $0
F355:        0001   DWRITE     EQU   $1
F355:        0003   DCTRL      EQU   $3
F355:        0004   GETDEVNUM  EQU   $4
F355:        0006   DOPEN      EQU   $6
F355:        0007   DCLOSE     EQU   $7
F355:               *
F355:        0002   CTRL.LIST  DS    2                ; CONTAINER FOR NEWLINE DCTRL CALL
F357:        0002   NEWLINECC  EQU   2                ; NEWLINE CTRL CODE
F357:               *
F357:               * GETDNUM VARS
F357:               *
F357:        0001   DNUM.TEMP  DS    1
F358:               *
F358:               * CLOSEALL VARS
F358:               *
F358:        00AF   DCLOSE.ERR EQU   F.TPARMX+$F
F358:        0200   DCLOSE.TBL EQU   $200
F358:        0080   TRUE       EQU   $80
F358:        0000   FALSE      EQU   $0
F358:               *
F358:               *
F358:               ************************************************************
F358:               *
F358:               * CHARACTER FILE CONTROL BLOCK TABLE
F358:               * (ENTRY 0 IS NOT USED)
F358:               *
F358:               ************************************************************
F358:        0011   CFCB.MAX   EQU   17
F358:        0011   CFCB.DEV   DS    CFCB.MAX
F369:        0011   CFCB.LVL   DS    CFCB.MAX
```

```
F37A:              ************************************************************
F37A:              *
F37A:              * CHARACTER FILE MANAGER - MAIN ENTRY POINT
F37A:              *
F37A:              ************************************************************
F37A:        F37A  CFMGR      EQU   *
F37A:              *
F37A:              * SWITCH, BASED ON REQUEST CODE
F37A:              *
F37A:A5 A0                    LDA   REQCODE
F37C:C9 08                    CMP   #OPEN
F37E:F0 1B   F39B             BEQ   CFOPEN           ; "OPEN"
F380:C9 09                    CMP   #NEWLINE
F382:F0 42   F3C6             BEQ   CFNEWLINE        ; "NEWLINE"
F384:C9 0A                    CMP   #READ
F386:F0 6B   F3F3             BEQ   CFREAD           ; "READ"
F388:C9 0B                    CMP   #WRITE
F38A:D0 03   F38F             BNE   CFM010
F38C:4C 28 F4                 JMP   CFWRITE          ; "WRITE"
F38F:C9 0C         CFM010     CMP   #CLOSE
F391:D0 03   F396             BNE   CFM020
F393:4C 4F F4                 JMP   CFCLOSE          ; "CLOSE"
F396:A9 00         CFM020     LDA   #BADSCNUM
F398:20 00 00                 JSR   SYSERR           ; ERR EXIT
```

```
F39B:              ************************************************************
F39B:              * OPEN(IN.PATHNAME; OUT.REFNUM; IN.OPENLIST,LENGTH) SYSTEM CALL
F39B:              ************************************************************
F39B:        F39B  CFOPEN     EQU   *                ; BUILD "D.OPEN" CALL
F39B:20 C3 F4                 JSR   GETDNUM          ; MAP PATH TO DEV#
F39E:B0 25   F3C5             BCS   CFOP.ERR1        ; ERR - FILE NOT FOUND
F3A0:85 C1                    STA   D.DNUM
F3A2:              *
F3A2:20 F2 F4                 JSR   REQ.CFCB         ; BUILD NEW CFCB ENTRY
F3A5:B0 1E   F3C5             BCS   CFOP.ERR1        ; ERR - CFCB FULL
F3A7:A2 00                    LDX   #0
F3A9:81 A3                    STA   (O.REFNUM,X)     ; RETURN REFNUM TO CALLER
F3AB:C0 01                    CPY   #1
F3AD:D0 09   F3B8             BNE   CFOP.EXIT        ; DEVICE ALREADY OPEN
F3AF:              *
F3AF:A9 06                    LDA   #DOPEN
F3B1:85 C0                    STA   D.SCNUM
F3B3:20 00 00                 JSR   DMGR             ; DOPEN CALL
F3B6:B0 01   F3B9             BCS   CFOP.ERR
F3B8:60            CFOP.EXIT  RTS                    ; NORMAL EXIT
F3B9:              *
F3B9:AD 00 00      CFOP.ERR   LDA   SERR             ;KLUDGE - 1.0 DRIVERS DON'T SUPPORT CARRY ERR PROTOCOL
F3BC:F0 FA   F3B8             BEQ   CFOP.EXIT        ;NO ERROR
F3BE:A2 00                    LDX   #0               ; RELEASE CFCB ENTRY
F3C0:A1 A3                    LDA   (O.REFNUM,X)
F3C2:20 17 F5                 JSR   REL.CFCB
F3C5:60            CFOP.ERR1  RTS                    ; ERR EXIT
```

```
F3C6:                ************************************************************
F3C6:                * NEWLINE(IN.REFNUM,IS .NEWLINE,NEWLINE.CHAR) SYSTEM CALL
F3C6:                ************************************************************
F3C6:        F3C6    CFNEWLINE  EQU   *                 ; BUILD "D.CONTROL" CALL
F3C6:A9 03                      LDA   #DCTRL
F3C8:85 C0                      STA   D.SCNUM
F3CA:A5 A1                      LDA   REFNUM
F3CC:20 3F F5                   JSR   GET.CFCB          ; MAP REFNUM TO DEV #
F3CF:B0 21   F3F2               BCS   CFNL.ERR          ; ERR - BAD REFNUM
F3D1:                *
F3D1:85 C1                      STA   D.DNUM
F3D3:A9 02                      LDA   #NEWLINECC
F3D5:85 C2                      STA   DC.CCODE
F3D7:                *
F3D7:A9 55                      LDA   #>CTRL.LIST
F3D9:85 C3                      STA   DC.CLIST
F3DB:A9 F3                      LDA   #<CTRL.LIST
F3DD:85 C4                      STA   DC.CLIST+1
F3DF:A9 00                      LDA   #0
F3E1:8D C4 00                   STA   SXPAGE+DC.CLIST+1
F3E4:                *
F3E4:A5 A2                      LDA   NL.ISNL
F3E6:8D 55 F3                   STA   CTRL.LIST
F3E9:A5 A3                      LDA   NL.NLCHR
F3EB:8D 56 F3                   STA   CTRL.LIST+1
F3EE:                *
F3EE:20 00 00                   JSR   DMGR              ; DCONTROL CALL
F3F1:60                         RTS                     ; NORMAL EXIT
F3F2:                *
F3F2:60              CFNL.ERR   RTS                     ; ERR EXIT
```

```
F3F3:              ************************************************************
F3F3:              * READ(IN.REFNUM,BUF,BYTES,BYTESREAD) SYSTEM CALL
F3F3:              ************************************************************
F3F3:        F3F3  CFREAD     EQU   *                ; BUILD "D.READ" CALL
F3F3:A9 00                    LDA   #DREAD
F3F5:85 C0                    STA   D.SCNUM
F3F7:A5 A1                    LDA   REFNUM
F3F9:20 3F F5                 JSR   GET.CFCB         ; MAP REFNUM TO DEV #
F3FC:B0 29   F427             BCS   CFRD.ERR         ; ERR - BAD REFNUM
F3FE:              *
F3FE:85 C1                    STA   D.DNUM
F400:A2 03                    LDX   #3
F402:B5 A2         CFRD010    LDA   RW.BUF,X
F404:95 C2                    STA   DRW.BUF,X
F406:CA                       DEX
F407:10 F9   F402             BPL   CFRD010
F409:              *
F409:A5 A6                    LDA   RD.BYTESRD
F40B:85 C8                    STA   DRD.BYTESRD
F40D:A5 A7                    LDA   RD.BYTESRD+1
F40F:85 C9                    STA   DRD.BYTESRD+1
F411:              *
F411:AD A3 00                 LDA   SXPAGE+RW.BUF+1
F414:8D C3 00                 STA   SXPAGE+DRW.BUF+1
F417:AD A5 00                 LDA   SXPAGE+RW.BYTES+1
F41A:8D C5 00                 STA   SXPAGE+DRW.BYTES+1
F41D:AD A7 00                 LDA   SXPAGE+RD.BYTESRD+1
F420:8D C9 00                 STA   SXPAGE+DRD.BYTESRD+1
F423:              *
F423:20 00 00                 JSR   DMGR             ; DREAD CALL
F426:60                       RTS                    ; NORMAL EXIT
F427:              *
F427:60            CFRD.ERR   RTS                    ; ERR EXIT
```

```
F428:              ************************************************************
F428:              * WRITE(IN.REFNUM,BUF,BYTES) SYSTEM CALL
F428:              ************************************************************
F428:        F428  CFWRITE    EQU   *                 ; BUILD "D.WRITE" CALL
F428:A9 01                    LDA   #DWRITE
F42A:85 C0                    STA   D.SCNUM
F42C:A5 A1                    LDA   REFNUM
F42E:20 3F F5                 JSR   GET.CFCB          ; MAP REFNUM TO DEV #
F431:B0 1B   F44E             BCS   CFWR.ERR          ; ERR - BAD REFNUM
F433:85 C1                    STA   D.DNUM
F435:A2 03                    LDX   #3
F437:B5 A2         CFWR010    LDA   RW.BUF,X
F439:95 C2                    STA   DRW.BUF,X
F43B:CA                       DEX
F43C:10 F9   F437             BPL   CFWR010
F43E:AD A3 00                 LDA   SXPAGE+RW.BUF+1
F441:8D C3 00                 STA   SXPAGE+DRW.BUF+1
F444:AD A5 00                 LDA   SXPAGE+RW.BYTES+1
F447:8D C5 00                 STA   SXPAGE+DRW.BYTES+1
F44A:              *
F44A:20 00 00                 JSR   DMGR              ; DWRITE CALL
F44D:60                       RTS                     ; NORMAL EXIT
F44E:              *
F44E:60            CFWR.ERR   RTS                     ; ERR EXIT
```

```
F44F:              ************************************************************
F44F:              * CLOSE(IN.REFNUM) SYSTEM CALL
F44F:              ************************************************************
F44F:        F44F  CFCLOSE    EQU   *                ; BUILD "D.CLOSE" CALL
F44F:A9 07                    LDA   #DCLOSE
F451:85 C0                    STA   D.SCNUM
F453:A5 A1                    LDA   REFNUM
F455:F0 0E   F465             BEQ   CLOSEALL
F457:              *
F457:20 17 F5                 JSR   REL.CFCB         ; RELEASE CFCB ENTRY
F45A:B0 08   F464             BCS   CFCL010
F45C:85 C1                    STA   D.DNUM
F45E:98                       TYA
F45F:D0 03   F464             BNE   CFCL010
F461:20 00 00                 JSR   DMGR             ; DCLOSE CALL
F464:60            CFCL010    RTS                    ; NORMAL EXIT
F465:              *
```

```
F465:              ************************************************************
F465:              *
F465:              * CLOSE ALL CHARACTER FILES W/LEVELS >= TO CURRENT SYSTEM FILE LEVEL.
F465:              *
F465:              ************************************************************
F465:              *
F465:        F465  CLOSEALL   EQU   *
F465:A9 00                    LDA   #FALSE           ; SET ENTRIES IN DEV CLOSE TBL TO FALSE
F467:AE 00 00                 LDX   MAX.DNUM
F46A:9D 00 02      CFCL020    STA   DCLOSE.TBL,X
F46D:CA                       DEX
F46E:10 FA   F46A             BPL   CFCL020
F470:              *
F470:A2 10                    LDX   #CFCB.MAX-1      ; CLOSE ALL DEVICES >= TO CURRENT LEVEL
F472:BD 58 F3      CFCL030    LDA   CFCB.DEV,X       ; AND MARK TRUE IN DEV CLOSE TBL
F475:A8                       TAY
F476:30 11   F489             BMI   CFCL050
F478:BD 69 F3                 LDA   CFCB.LVL,X
F47B:CD 00 00                 CMP   LEVEL
F47E:90 09   F489             BCC   CFCL050
F480:A9 80                    LDA   #TRUE
F482:99 00 02                 STA   DCLOSE.TBL,Y
F485:38                       SEC
F486:7E 58 F3                 ROR   CFCB.DEV,X
F489:CA            CFCL050    DEX
F48A:D0 E6   F472             BNE   CFCL030
F48C:              *
F48C:A2 10                    LDX   #CFCB.MAX-1      ; DON'T CLOSE DEVICES < CURRENT LEVEL
F48E:BD 58 F3      CFCL060    LDA   CFCB.DEV,X
F491:A8                       TAY
F492:30 05   F499             BMI   CFCL070
F494:A9 00                    LDA   #FALSE
F496:99 00 02                 STA   DCLOSE.TBL,Y
F499:CA            CFCL070    DEX
F49A:D0 F2   F48E             BNE   CFCL060
F49C:              *
F49C:A9 00                    LDA   #0
F49E:85 AF                    STA   DCLOSE.ERR
F4A0:AE 00 00                 LDX   MAX.DNUM         ; ISSUE D'CLOSE CALLS TO ALL DEVICES MARKED AS TRUE
F4A3:BD 00 02      CFCL080    LDA   DCLOSE.TBL,X     ; IN DEV CLOSE TABLE
F4A6:10 10   F4B8             BPL   CFCL090
F4A8:8A                       TXA
F4A9:48                       PHA
F4AA:86 C1                    STX   D.DNUM
F4AC:20 00 00                 JSR   DMGR
F4AF:68                       PLA
F4B0:AA                       TAX
F4B1:AD 00 00                 LDA   SERR
F4B4:F0 02   F4B8             BEQ   CFCL090          ; IF ERROR,
F4B6:85 AF                    STA   DCLOSE.ERR       ; THEN SAVE IT
F4B8:CA            CFCL090    DEX
F4B9:D0 E8   F4A3             BNE   CFCL080
F4BB:              *
F4BB:A5 AF                    LDA   DCLOSE.ERR       ; IF $0 THEN NO ERRORS FROM D.CLOSE CALLS
F4BD:D0 01   F4C0             BNE   CFCL.ERR
F4BF:60                       RTS                    ; NORMAL EXIT
```

```
F4C0:20 00 00      330 CFCL.ERR   JSR   SYSERR           ; RETURN LAST D.CLOSE ERROR REPORTED
```

```
F4C3:              ************************************************************
F4C3:              *
F4C3:              * GET DEVICE NUMBER
F4C3:              *
F4C3:              * INPUT:  CPATH
F4C3:              * OUTPUT: DEVICE NUMBER (A)
F4C3:              * ERROR:  CARRY SET ("FILE NOT FOUND")
F4C3:              *
F4C3:              * GETDNUM FIRST CALLS THE DMGR (GETDEVNUM) MAP THE PATHNAME
F4C3:              * TO A DEVICE #.  GETDNUM THEN ENSURES THAT THE PATHNAME
F4C3:              * IS NOT A BLOCK DEVICE BY CHECKING THE DBLKLST TABLE.
F4C3:              *
F4C3:              ************************************************************
F4C3:              *
F4C3:        F4C3  GETDNUM    EQU   *
F4C3:A9 04                    LDA   #GETDEVNUM
F4C5:85 C0                    STA   D.SCNUM
F4C7:              *
F4C7:A5 A1                    LDA   O.PATH
F4C9:85 C1                    STA   GDN.DNAME
F4CB:A5 A2                    LDA   O.PATH+1
F4CD:85 C2                    STA   GDN.DNAME+1
F4CF:              *
F4CF:A9 57                    LDA   #>DNUM.TEMP
F4D1:85 C3                    STA   GDN.DNUM
F4D3:A9 F3                    LDA   #<DNUM.TEMP
F4D5:85 C4                    STA   GDN.DNUM+1
F4D7:              *
F4D7:AD A2 00                 LDA   SXPAGE+O.PATH+1
F4DA:8D C2 00                 STA   SXPAGE+GDN.DNAME+1
F4DD:A9 00                    LDA   #0
F4DF:8D C4 00                 STA   SXPAGE+GDN.DNUM+1
F4E2:              *
F4E2:20 00 00                 JSR   DMGR
F4E5:B0 06   F4ED             BCS   GETD.ERR         ; D.NAME NOT FOUND
F4E7:30 04   F4ED             BMI   GETD.ERR         ; BLOCK DEVICE FOUND
F4E9:AD 57 F3                 LDA   DNUM.TEMP
F4EC:60                       RTS
F4ED:              *
F4ED:A9 00         GETD.ERR   LDA   #FNFERR
F4EF:20 00 00                 JSR   SYSERR
```

```
F4F2:              ************************************************************
F4F2:              * REQUEST FCB ENTRY
F4F2:              *
F4F2:              * INPUT: DNUM (A)
F4F2:              * OUTPUT: REFNUM (A), OPENCT (Y)
F4F2:              * ERROR:  CARRY SET ("CFCB FULL")
F4F2:              *
F4F2:              * REQ.CFCB FIRST SEARCHES THE CFCB TABLE USING THE DEV#
F4F2:              * AS A KEY.  IF FOUND THE OPENCT IS INCREMENTED, OTHERWISE,
F4F2:              * REQ.CFCB FINDS A FREE ENTRY AND STORES THE DEV# AND LEVEL #.
F4F2:              *
F4F2:              ************************************************************
F4F2:              *
F4F2:        F4F2  REQ.CFCB   EQU   *
F4F2:A2 10                    LDX   #CFCB.MAX-1
F4F4:A8                       TAY
F4F5:BD 58 F3      REQ010     LDA   CFCB.DEV,X
F4F8:30 08   F502             BMI   REQ020
F4FA:CA                       DEX
F4FB:D0 F8   F4F5             BNE   REQ010
F4FD:A9 00                    LDA   #CFCBFULL
F4FF:20 00 00                 JSR   SYSERR
F502:98            REQ020     TYA
F503:9D 58 F3                 STA   CFCB.DEV,X
F506:AD 00 00                 LDA   LEVEL
F509:9D 69 F3                 STA   CFCB.LVL,X
F50C:8A                       TXA
F50D:48                       PHA
F50E:98                       TYA
F50F:20 31 F5                 JSR   OPENCOUNT
F512:68                       PLA
F513:09 80                    ORA   #$80
F515:18                       CLC
F516:60                       RTS                   ; NORMAL EXIT
```

```
F517:              ************************************************************
F517:              *
F517:              * RELEASE FCB ENTRY
F517:              *
F517:              * INPUT:  REFNUM (A)
F517:              * OUTPUT:  DNUM (A), OPENCT (Y)
F517:              * ERROR:   CARRY SET ("INVALID REFNUM")
F517:              *
F517:              * USES REFNUM AS AN CFCB TABLE INDEX TO RELEASE A CFCB ENTRY.
F517:              *
F517:              ************************************************************
F517:        F517  REL.CFCB   EQU   *
F517:29 7F                    AND   #$7F
F519:C9 11                    CMP   #CFCB.MAX
F51B:B0 0F   F52C             BCS   REL.ERR
F51D:AA                       TAX
F51E:BD 58 F3                 LDA   CFCB.DEV,X
F521:30 09   F52C             BMI   REL.ERR
F523:38                       SEC                    ; MARK ENTRY FREE
F524:7E 58 F3                 ROR   CFCB.DEV,X
F527:20 31 F5                 JSR   OPENCOUNT
F52A:18                       CLC
F52B:60                       RTS                    ; NORMAL EXIT
F52C:              *
F52C:A9 00         REL.ERR    LDA   #BADREFNUM
F52E:20 00 00                 JSR   SYSERR
F531:              ************************************************************
F531:              *
F531:              * OPENCOUNT SUBROUTINE
F531:              *
F531:              * INPUT:   DEVNUM (A)
F531:              * OUTPUT:  DEVNUM (A), OPENCTR (Y)
F531:              *
F531:              * OPENCTR:=COUNT OF ALL CFCB ENTRIES W/CFCB.DEV=DEVNUM
F531:              *
F531:              ************************************************************
F531:        F531  OPENCOUNT  EQU   *
F531:A0 00                    LDY   #0
F533:A2 10                    LDX   #CFCB.MAX-1
F535:DD 58 F3      OPNCT010   CMP   CFCB.DEV,X
F538:D0 01   F53B             BNE   OPNCT020
F53A:C8                       INY
F53B:CA            OPNCT020   DEX
F53C:D0 F7   F535             BNE   OPNCT010
F53E:60                       RTS
```

```
F53F:              ************************************************************
F53F:              *
F53F:              * GET FCB ENTRY
F53F:              *
F53F:              * INPUT:   REFNUM (A)
F53F:              * OUTPUT:  DNUM (A)
F53F:              * ERROR:   CARRY SET ("INVALID REFNUM")
F53F:              *
F53F:              * USES REFNUM AS AN INDEX TO RETURN THE CORRESPONDING DEVICE #.
F53F:              * IF THE ENTRY INDICATED BY REFNUM IS A FREE ENTRY, THEN AN
F53F:              * ERROR, "INVALID REF NUM" IS RETURNED.
F53F:              *
F53F:              ************************************************************
F53F:        F53F  GET.CFCB   EQU   *
F53F:29 7F                    AND   #$7F
F541:C9 11                    CMP   #CFCB.MAX
F543:B0 08   F54D             BCS   GET.ERR
F545:AA                       TAX
F546:BD 58 F3                 LDA   CFCB.DEV,X
F549:30 02   F54D             BMI   GET.ERR
F54B:18                       CLC
F54C:60                       RTS                    ; NORMAL EXIT
F54D:              *
F54D:A9 00         GET.ERR    LDA   #BADREFNUM
F54F:20 00 00                 JSR   SYSERR           ; ERR EXIT
F552:              *

F552:                         LST   ON
F552:        F552  ZZEND      EQU   *
F552:        01FD  ZZLEN      EQU   ZZEND-ZZORG
F552:        0000             IFNE  ZZLEN-LENCFM
 S                            FAIL  2,"SOSORG        FILE IS INCORRECT FOR CFMGR"
F552:                         FIN
```

```
X000C BADREFNUM     X000A BADSCNUM      ?2E00 BLABFMI        3200 BLABFM
 6B52 BLABUFMG       6955 BLACFM         5E99 BLADISK3       64D9 BLADMGR
 68F4 BLAFMGR       ?2CF8 BLAGLOB       ?2AF8 BLAINIT        55C0 BLAIPL
 2000 BLALODR       ?6E6E BLAMEMMG       5466 BLAOMSG        5466 BLAPATCH
 665E BLASCMGR       6404 BLASERR        5A8B BLAUMGR       NF358 CFCB.DEV
 F369 CFCB.LVL      N0011 CFCB.MAX      X000B CFCBFULL       F4C0 CFCL.ERR
 F464 CFCL010        F46A CFCL020        F472 CFCL030        F489 CFCL050
 F48E CFCL060        F499 CFCL070        F4A3 CFCL080        F4B8 CFCL090
 F44F CFCLOSE        F38F CFM010         F396 CFM020        NF37A CFMGR
 F3C6 CFNEWLINE      F3F2 CFNL.ERR       F3C5 CFOP.ERR1      F3B9 CFOP.ERR
 F3B8 CFOP.EXIT      F39B CFOPEN         F427 CFRD.ERR       F402 CFRD010
 F3F3 CFREAD         F44E CFWR.ERR       F437 CFWR010        F428 CFWRITE
 F465 CLOSEALL         0C CLOSE          F355 CTRL.LIST        C1 D.DNUM
   C0 D.SCNUM          C0 D.TPARMX         C2 DC.CCODE         C3 DC.CLIST
   AF DCLOSE.ERR       07 DCLOSE         0200 DCLOSE.TBL       03 DCTRL
X0004 DMGR           F357 DNUM.TEMP        06 DOPEN            C8 DRD.BYTESRD
   00 DREAD            C2 DRW.BUF          C4 DRW.BYTES        01 DWRITE
   A0 F.TPARMX         00 FALSE         X000D FNFERR           C1 GDN.DNAME
   C3 GDN.DNUM       F53F GET.CFCB       F54D GET.ERR        F4ED GETD.ERR
   04 GETDEVNUM      F4C3 GETDNUM       ?0400 LENBFMI        2266 LENBFM
 031C LENBUFMG       01FD LENCFM         056B LENDISK3       0185 LENDMGR
   61 LENFMGR       ?01B2 LENINIT        04CB LENIPL         0AF8 LENLODR
?0751 LENMEMMG       015A LENOMSG          00 LENPATCH       0296 LENSCMGR
   D5 LENSERR        040E LENUMGR       X0005 LEVEL         X0006 MAX.DNUM
   09 NEWLINE          02 NEWLINECC        A2 NL.ISNL          A3 NL.NLCHR
   A1 O.PATH           A3 O.REFNUM       F531 OPENCOUNT        08 OPEN
 F535 OPNCT010       F53B OPNCT020       B800 ORGBFMI        BC00 ORGBFM
 F552 ORGBUFMG       F355 ORGCFM         E899 ORGDISK3       EED9 ORGDMGR
 FFBF ORGEND         F2F4 ORGFMGR       ?18FC ORGGLOB        28F8 ORGINIT
 DFC0 ORGIPL         1E00 ORGLODR        F86E ORGMEMMG       DE66 ORGOMSG
 DE66 ORGPATCH       F05E ORGSCMGR       EE04 ORGSERR        E48B ORGUMGR
   A6 RD.BYTESRD       0A READ             A1 REFNUM         F517 REL.CFCB
 F52C REL.ERR        F4F2 REQ.CFCB       F4F5 REQ010         F502 REQ020
   A0 REQCODE          A2 RW.BUF           A4 RW.BYTES      X0009 SERR
X0007 SXPAGE        X0008 SYSERR           80 TRUE             0B WRITE
 F552 ZZEND          01FD ZZLEN          F355 ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   545
** FREE SPACE PAGE COUNT   81
```

```
SOURCE   FILE #01 =>BUFMGR.SRC
 INCLUDE FILE #02 =>SOSORG
```

```
0000:                             REL
0000:                             INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955      BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52      BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E      BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
0000:                  *ORGBUFMG EQU $F552
0000:                  *LENBUFMG EQU $31C
F552:        F552                 ORG   ORGBUFMG
F552:        F552      ZZORG      EQU   *
F552:                             MSB   OFF
F552:                  ************************************************************
F552:                  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
F552:                  *                    ALL RIGHTS RESERVED
F552:                  ************************************************************
F552:                  *
F552:                  * BUFFER MANAGER (VERSION = 1.1O   )
F552:                  *                (DATE    = 8/04/81)
F552:                  *
F552:                  * THIS MODULE IS RESPONSIBLE FOR CREATING AND RELEASING BUFFERS
F552:                  * FOR BOTH THE BLOCK FILE MANAGER AND, LATER, DEVICE HANDLERS
F552:                  * THE BUFFER MANAGER CREATES BUFFERS BY REQUESTING MEMORY
F552:                  * SEGMENTS FROM THE MEMORY MANAGER, AND RELEASES THEM VIA SAME.
F552:                  * THE PRIMARY DATA STRUCTURE IN THIS MODULE IS THE BUFFER TABLE.
F552:                  *
F552:                  ************************************************************
F552:                  *
F552:        F5C5                 ENTRY REQBUF
F552:        F622                 ENTRY REQFXBUF
F552:        F686                 ENTRY GETBUFADR
F552:        F6EC                 ENTRY CHKBUF
F552:        F710                 ENTRY RELBUF
F552:                  *
F552:        0000                 EXTRN MMGR
F552:        0000                 EXTRN SXPAGE
F552:        0000                 EXTRN CZPAGE
F552:        0000                 EXTRN CXPAGE
F552:                  *
F552:        0000                 EXTRN SYSERR
F552:        0000                 EXTRN SERR
F552:        0000                 EXTRN OUTOFMEM
F552:        0000                 EXTRN BUFTBLFULL
F552:        0000                 EXTRN BADSYSBUF
F552:                  *
F552:        0000                 EXTRN SYSDEATH
F552:        0000                 EXTRN BADBUFNUM
F552:        0000                 EXTRN BADBUFSIZ
F552:                  *
F552:        0011                 ENTRY BUF.CNT
F552:        F55E                 ENTRY PGCT.T
F552:        F56F                 ENTRY XBYTE.T
F552:        F5B3                 ENTRY BUFREF
```

```
F552:               ************************************************************
F552:               *
F552:               * DATA DECLARATIONS
F552:               *
F552:               ************************************************************
F552:               *
F552:        FFD0   Z.REG      EQU   $FFD0
F552:               *
F552:               * MEMORY MGMT CALL PARM LOCATIONS ON SOS ZPAGE
F552:               *
F552:        0060   M.TPARMX   EQU   $60              ; FIRST ADR OF MEM SYS CALL PARMS ON SOS ZPAGE
F552:        0060   REQCODE    EQU   M.TPARMX+$0
F552:               *
F552:        0001   FINDSEG    EQU   $1
F552:        0061   SRCHMODE   EQU   M.TPARMX+$1
F552:        0062   F.ID       EQU   M.TPARMX+$2
F552:        0063   F.PGCT     EQU   M.TPARMX+$3
F552:        0002   F.PGCTX    DS    2                ; TEMP LOC FOR F.PGCT PARM
F554:        0065   F.BASE     EQU   M.TPARMX+$5
F554:        0002   F.BASEX    DS    2                ; TEMP LOC FOR F.BASE PARM
F556:        0067   F.LIM      EQU   M.TPARMX+$7
F556:        0002   F.LIMX     DS    2                ; TEMP LOC FOR F.LIM PARM
F558:        0069   F.NUM      EQU   M.TPARMX+$9
F558:        0001   F.NUMX     DS    1                ; TEMP LOC FOR F.NUM PARM
F559:               *
F559:        0005   RELSEG     EQU   $5
F559:        0061   RLS.NUM    EQU   M.TPARMX+$1
F559:               *
F559:               * REQBUF DATA DECLARATIONS
F559:               *
F559:        0001   RQB.PGCT   DS    1                ; REQUESTED PAGE COUNT
F55A:        0001   RQB.BNUM   DS    1                ; BUFFER NUMBER (FM GETFREE CALL)
F55B:               *
F55B:               * REQFXBUF DATA DECLARATIONS
F55B:               *
F55B:        0001   RQFB.PGCT  DS    1                ; REQUESTED PAGE COUNT
F55C:        0001   RQFB.BNUM  DS    1                ; BUFFER NUMBER (FM GETFREE CALL)
F55D:        0040   MAXPGCT    EQU   64               ; MAX BUFSIZE=16K
F55D:        00A0   F.TPARMX   EQU   $A0              ; FIRST ADR OF FILE SYS CALL PARMS ON SOS ZPAGE
F55D:        00A5   OPEN.LIST  EQU   F.TPARMX+$5      ; LOC OF OPEN.LIST PARM (OPEN SYS CALL)
F55D:               *
F55D:               * BUFCOMPACT DATA DECLARATIONS (SOURCE ALSO USED BY CHKBUF)
F55D:               *
F55D:        0001   BUFC.BNUM  DS    1                ; BUF# OF LOWEST BUFFER IN BUF.TBL
F55E:        0070   SOURCE     EQU   M.TPARMX+$10     ; & $11
F55E:        0072   DEST       EQU   M.TPARMX+$12     ; & $13
```

```
F55E:              ************************************************************
F55E:              *
F55E:              * BUFFER TABLE
F55E:              *
F55E:              * THE BUFFER TABLE CONSISTS OF "CNT"-1 ENTRIES (1 TO "CNT"-1).
F55E:              * EACH ENTRY IS "SIZ" BYTES IN LENGTH.  THE "PGCT" FIELD
F55E:              * CONTAINS 3 SUBFIELDS.  BIT 7 IS THE "FREE" FLAG (0=ACTIVE,1=FREE)
F55E:              * BIT 6 IS THE "FIXED" FLAG (0=FLOATING BUFFER,1=FIXED BUFFER)
F55E:              * BITS 5 THRU 0 CONTAIN THE PAGE COUNT OF AN "ACTIVE" ENTRY
F55E:              * (0=>1 PAGE,63=>64 PAGES DECIMAL).  THE "XBYTE" FIELD CONTAINS
F55E:              * THE PROPER XBYTE OF AN "ACTIVE" ENTRY.  THE "ADRH" FIELD
F55E:              * CONTAINS THE HIGH BYTE OF THE BUFFER ADDRESS.  IF THE
F55E:              * BUFFER ENTRY IS "FLOATING", THEN THE "SEG" FIELD CONTAINS THE
F55E:              * SEGMENT NUMBER AND THE LOW BYTE OF THE BUFFER ADDRESS IS
F55E:              * ASSUMMED TO BE ZERO.
F55E:              *
F55E:              * THUS, THE FOLLOWING RESTRICTIONS APPLY TO BUFFERS:
F55E:              *
F55E:              * (1) MAXIMUM BUFFER LENGTH IS 64 PAGES (16K)
F55E:              * (2) "FLOATING" BUFFERS ALWAYS BEGIN ON A PAGE BOUNDARY
F55E:              *     "FIXED" BUFFERS DO NOT.
F55E:              * (3) BUFFERS ARE ALWAYS AN INTEGRAL NUMBER OF PAGES IN LENGTH
F55E:              * (4) BUFFERS ALWAYS RESIDE IN THE 32K BANK MEMORY REGION,
F55E:              *     A LIMITATION OF FIND.SEG (MEMORY MANAGER)
F55E:              * (5) MAXIMUM NUMBER OF BUFFERS = 16; ENTRY 0 IS NOT USED.
F55E:              *
F55E:              ************************************************************
F55E:              *
F55E:              * BUFFER TABLE
F55E:              *
F55E:        0005  BUF.SIZ    EQU   5
F55E:        0011  BUF.CNT    EQU   17
F55E:        0055  BUF.TBL    DS    BUF.SIZ*BUF.CNT
F5B3:        F55E  PGCT.T     EQU   BUF.TBL
F5B3:        F56F  XBYTE.T    EQU   PGCT.T+BUF.CNT
F5B3:        F580  ADRH.T     EQU   XBYTE.T+BUF.CNT
F5B3:        F591  SEG.T      EQU   ADRH.T+BUF.CNT
F5B3:        F591  ADRL.T     EQU   SEG.T
F5B3:        F5A2  CHK.T      EQU   ADRL.T+BUF.CNT
F5B3:        0040  ISFIXED    EQU   $40
F5B3:        0080  ISFREE     EQU   $80
F5B3:              *
F5B3:              * BUFFER REFERENCE TABLE
F5B3:              *
F5B3:              * FIRST BYTE IS COUNT, FOLLOWED BY "COUNT" BUFFER #S.
F5B3:              * THIS TABLE IS A LIST OF ALL BUFFERS REFERENCED DURING ONE
F5B3:              * SOS SYSTEM CALL.  BUFFER #S ARE ADDED TO THIS LIST BY
F5B3:              * GETBUFADR AND REMOVED BY CHKSUM.
F5B3:              *
F5B3:        0011  BUFREF.CNT EQU   17
F5B3:        0011  BUFREF     DS    BUFREF.CNT
F5C4:        0001  ZPAGEX     DS    1
```

```
F5C5:              ************************************************************
F5C5:              *
F5C5:              * REQBUF
F5C5:              *
F5C5:              * INPUT:  PAGE.CNT (A)
F5C5:              * OUTPUT: BUFNUM   (A)
F5C5:              * ERROR:  "BUFFER TABLE FULL" - SYSERR
F5C5:              *         "OUT OF MEMORY"     - SYSERR
F5C5:              *         "BAD BUFFER SIZE"   - SYSDEATH
F5C5:              *
F5C5:              * THIS ROUTINE FINDS A FREE ENTRY IN THE BUFFER TABLE
F5C5:              * AND THEN CALLS FIND.SEG (MMGR) TO OBTAIN MEMORY FOR IT.
F5C5:              * IF MEMORY IS FOUND THEN THE BUFFER ENTRY IS MARKED "ACTIVE"
F5C5:              * AND THE BUFFER INFO IS INSERTED INTO THE ENTRY
F5C5:              *
F5C5:              ************************************************************
F5C5:              *
F5C5:        F5C5  REQBUF     EQU   *
F5C5:              *
F5C5:              * IF REQUESTED PGCT OUT OF BOUNDS THEN FATAL ERR
F5C5:              *
F5C5:A8                       TAY
F5C6:F0 55   F61D             BEQ   RQB.ERR2         ; FATAL ERR, INVALID BUFFER SIZE
F5C8:C0 41                    CPY   #MAXPGCT+1
F5CA:B0 51   F61D             BCS   RQB.ERR2         ; FATAL ERR, INVALID BUFFER SIZE
F5CC:8C 59 F5                 STY   RQB.PGCT         ; SAVE PAGE COUNT
F5CF:              *
F5CF:              * FIND FREE ENTRY IN BUF.TBL
F5CF:              *
F5CF:20 40 F8                 JSR   GETFREE
F5D2:B0 3F   F613             BCS   RQB.ERR          ; ERR, BUFFER TABLE FULL
F5D4:8E 5A F5                 STX   RQB.BNUM
F5D7:              *
F5D7:              * FIND PGCT*256 BYTES OF FREE MEMORY
F5D7:              *
F5D7:AD 59 F5                 LDA   RQB.PGCT
F5DA:20 F9 F7                 JSR   FSEG
F5DD:B0 39   F618             BCS   RQB.ERR1         ; ERR, OUT OF MEMORY
F5DF:              *
F5DF:              * INSERT PGCT, XBYTE, ADRH, SEG#, CHK BYTE IN BUF.TBL(BUF#)
F5DF:              *
F5DF:AE 5A F5                 LDX   RQB.BNUM
F5E2:CE 59 F5                 DEC   RQB.PGCT         ; PAGE COUNT FIELD
F5E5:AD 59 F5                 LDA   RQB.PGCT
F5E8:9D 5E F5                 STA   PGCT.T,X
F5EB:              *
F5EB:AE 54 F5                 LDX   F.BASEX          ; XBYTE & ADRH FIELDS
F5EE:AC 55 F5                 LDY   F.BASEX+1
F5F1:20 51 F8                 JSR   CNVRT.ADR
F5F4:E0 8F                    CPX   #$8F
F5F6:D0 02   F5FA             BNE   RQB010
F5F8:A2 7F                    LDX   #$7F             ; IF XBYTE=$8F THEN XBYTE:=$7F
F5FA:8A            RQB010     TXA
F5FB:AE 5A F5                 LDX   RQB.BNUM
F5FE:9D 6F F5                 STA   XBYTE.T,X
F601:98                       TYA
```

```
F602:9D 80 F5                     STA   ADRH.T,X
F605:                  *
F605:AD 58 F5                     LDA   F.NUMX           ; SEG# FIELD
F608:9D 91 F5                     STA   SEG.T,X
F60B:                  *
F60B:A9 00                        LDA   #0               ; INIT CHECK BYTE TO NULL
F60D:9D A2 F5                     STA   CHK.T,X
F610:                  *
F610:8A                           TXA                    ; RETURN BUF#
F611:18                           CLC
F612:60                           RTS                    ; NORMAL EXIT
F613:                  *
F613:                  *
F613:A9 00             RQB.ERR    LDA   #BUFTBLFULL
F615:20 00 00                     JSR   SYSERR
F618:                  *
F618:A9 00             RQB.ERR1   LDA   #OUTOFMEM
F61A:20 00 00                     JSR   SYSERR
F61D:                  *
F61D:A9 00             RQB.ERR2   LDA   #BADBUFSIZ
F61F:20 00 00                     JSR   SYSDEATH
```

```
F622:              ************************************************************
F622:              *
F622:              * REQFXBUF
F622:              *
F622:              * INPUT:  PAGE.CNT (A)
F622:              * OUTPUT: BUFNUM   (A)
F622:              * ERROR:  "BUFFER TABLE FULL"            - SYSERR
F622:              *         "BAD SYSTEM.BUF PARM ADDRESS"  - SYSERR
F622:              *         "BAD BUFFER SIZE"              - SYSDEATH
F622:              *
F622:              * THIS ROUTINE COMPUTES THE ACTUAL BUFFER ADDRESS IN THE OPEN
F622:              * CALL (PARM "OPEN.LIST"), AND ALLOCATES A BUFFER ENTRY FOR IT.
F622:              * NOTE:  THE SYSBUF PARAMETER MUST BE AN EXTENDED INDIRECT PTR!!
F622:              *
F622:              ************************************************************
F622:              *
F622:        F622  REQFXBUF   EQU   *
F622:              *
F622:              * IF REQUESTED PGCT OUT OF BOUNDS THEN FATAL ERR
F622:              *
F622:A8                       TAY
F623:F0 5C   F681             BEQ   RQFB.ERR2        ; FATAL ERR, BAD BUFFER SIZE
F625:C0 41                    CPY   #MAXPGCT+1
F627:B0 58   F681             BCS   RQFB.ERR2        ; FATAL ERR, BAD BUFFER SIZE
F629:              *
F629:8C 5B F5                 STY   RQFB.PGCT        ; SAVE PAGE COUNT
F62C:              *
F62C:              * GET A FREE BUFFER ENTRY
F62C:              *
F62C:20 40 F8                 JSR   GETFREE
F62F:B0 46   F677             BCS   RQFB.ERR         ; ERR, BUFFER TABLE FULL
F631:8E 5C F5                 STX   RQFB.BNUM        ; SAVE BUF#
F634:              *
F634:              * FETCH SYSTEM.BUF PARAMETER IN OPEN SYSTEM CALL
F634:              *
F634:A0 03                    LDY   #3
F636:B1 A5                    LDA   (OPEN.LIST),Y
F638:D0 42   F67C             BNE   RQFB.ERR1        ; ERR, SYSBUF ADR
F63A:88                       DEY
F63B:B1 A5                    LDA   (OPEN.LIST),Y
F63D:A8                       TAY
F63E:B9 01 00                 LDA   CXPAGE+1,Y
F641:10 39   F67C             BPL   RQFB.ERR1        ; ERR, SYSBUF ADR
F643:C9 8F                    CMP   #$8F
F645:B0 35   F67C             BCS   RQFB.ERR1        ; ERR, SYSBUF ADR
F647:              *
F647:              * INSERT XBYTE, ADRH, ADRL, PGCT, CHK BYTE INTO BUF.TBL(BUF#)
F647:              *
F647:AE 5C F5                 LDX   RQFB.BNUM
F64A:9D 6F F5                 STA   XBYTE.T,X
F64D:              *
F64D:B9 01 00                 LDA   CZPAGE+1,Y
F650:F0 2A   F67C             BEQ   RQFB.ERR1        ; ERR SYSBUF ADR
F652:C9 81                    CMP   #$81             ; CHECK FOR ADDRESS COMPENSATION
F654:90 05   F65B             BCC   RQFB010
F656:FE 6F F5                 INC   XBYTE.T,X
```

```
F659:29 7F                    AND   #$7F
F65B:9D 80 F5      RQFB010    STA   ADRH.T,X
F65E:              *
F65E:B9 00 00                 LDA   CZPAGE,Y
F661:9D 91 F5                 STA   ADRL.T,X
F664:              *
F664:CE 5B F5                 DEC   RQFB.PGCT
F667:AD 5B F5                 LDA   RQFB.PGCT
F66A:09 40                    ORA   #ISFIXED
F66C:9D 5E F5                 STA   PGCT.T,X         ; BUFFER ENTRY NOW "ACTIVE"
F66F:              *
F66F:A9 00                    LDA   #0               ; INIT CHECK BYTE TO NULL
F671:9D A2 F5                 STA   CHK.T,X
F674:              *
F674:8A                       TXA                    ; RETURN BUF#
F675:18                       CLC
F676:60                       RTS                    ; NORMAL EXIT
F677:              *
F677:A9 00         RQFB.ERR   LDA   #BUFTBLFULL
F679:20 00 00                 JSR   SYSERR
F67C:              *
F67C:A9 00         RQFB.ERR1  LDA   #BADSYSBUF
F67E:20 00 00                 JSR   SYSERR
F681:              *
F681:A9 00         RQFB.ERR2  LDA   #BADBUFSIZ
F683:20 00 00                 JSR   SYSDEATH
```

```
F686:             ************************************************************
F686:             *
F686:             * GETBUFADR
F686:             *
F686:             * INPUT:  BUFNUM   (A)
F686:             *         ZPAGELOC (X)
F686:             * OUTPUT: BUF ADR AT: X,X+1 & SXPAGE+1,X
F686:             *         PAGE.CNT (A)
F686:             *         BUFNUM   (Y)
F686:             *
F686:             * ERROR:  "BADBUFNUM" SYSDEATH
F686:             *
F686:             ************************************************************
F686:             *
F686:        F686 GETBUFADR  EQU   *
F686:             *
F686:             * IF BUF# OUT OF RANGE OR BUF.TBL(BUF#)=FREE
F686:             * THEN FATAL ERR
F686:             *
F686:A8                      TAY
F687:F0 43   F6CC            BEQ   GTBF.ERR         ; BUF#=0, FATAL ERR
F689:C0 11                   CPY   #BUF.CNT
F68B:B0 3F   F6CC            BCS   GTBF.ERR         ; BUF# > MAX BUF TABLE ENTRY, FATAL ERR
F68D:B9 5E F5                LDA   PGCT.T,Y
F690:30 3A   F6CC            BMI   GTBF.ERR         ; BUF ENTRY MARKED "FREE", FATAL ERR
F692:             *
F692:             * OTHERWISE, CONSTRUCT BUFFER PTR ON SOS ZPAGE
F692:             *
F692:20 D1 F6                JSR   GETBUFADR1
F695:             *
F695:             * IF BUFFER NOT PREVIOUSLY REFERENCED ON THIS SOS CALL AND CHECK BYTE <> 0
F695:             *    THEN COMPARE FIRST BYTE OF BUFFER WITH CHECK BYTE IN BUFFER TABLE.
F695:             *         IF NO MATCH THEN KILL SYSTEM.
F695:             *
F695:8E C4 F5                STX   ZPAGEX
F698:98                      TYA
F699:AE B3 F5                LDX   BUFREF
F69C:F0 08   F6A6            BEQ   GTBF020          ; BUFREF EMPTY
F69E:             *
F69E:DD B3 F5     GTBF010    CMP   BUFREF,X         ; SEARCH FOR PREVIOUS REFERENCE
F6A1:F0 1F   F6C2            BEQ   GTBF030          ; MATCH FOUND
F6A3:CA                      DEX
F6A4:D0 F8   F69E            BNE   GTBF010
F6A6:             *
F6A6:EE B3 F5     GTBF020    INC   BUFREF           ; LOG BUF # IN BUFREF TABLE
F6A9:AE B3 F5                LDX   BUFREF
F6AC:E0 11                   CPX   #BUFREF.CNT
F6AE:B0 1C   F6CC            BCS   GTBF.ERR         ; BUFREF TABLE OVFLOW, KILL SYSTEM
F6B0:9D B3 F5                STA   BUFREF,X
F6B3:             *
F6B3:B9 A2 F5                LDA   CHK.T,Y
F6B6:F0 0A   F6C2            BEQ   GTBF030          ; NO CHECK BYTE, SKIP CHECK
F6B8:AE C4 F5                LDX   ZPAGEX
F6BB:A1 00                   LDA   ($0,X)           ; COMPARE FIRST BYTE OF BUFFER
F6BD:D9 A2 F5                CMP   CHK.T,Y          ; WITH CHECK BYTE IN BUF TABLE
F6C0:D0 0A   F6CC            BNE   GTBF.ERR         ; NO MATCH, PULL THE PLUG
```

```
F6C2:              *
F6C2:              * RETURN PAGE.CNT TO CALLER
F6C2:              *
F6C2:B9 5E F5      GTBF030    LDA   PGCT.T,Y
F6C5:29 3F                    AND   #$3F             ; STRIP OFF FREE,FIXED FLAGS
F6C7:18                       CLC
F6C8:69 01                    ADC   #1
F6CA:              *
F6CA:18                       CLC
F6CB:60                       RTS
F6CC:              *
F6CC:              *
F6CC:A9 00         GTBF.ERR   LDA   #BADBUFNUM
F6CE:20 00 00                 JSR   SYSDEATH
F6D1:              *
F6D1:              *
F6D1:              ************************************************************
F6D1:              *
F6D1:              * GETBUFADR1
F6D1:              *
F6D1:              * INPUT: PGCT.T(BUF#)  (A)
F6D1:              *        ZPAGELOC      (X)
F6D1:              *        BUF#          (Y)
F6D1:              * ERROR: NONE.
F6D1:              *
F6D1:              * EXTRACTS THE BUFFER POINTER FROM THE BUFFER TABLE AND
F6D1:              * PLACES IT ON ZERO PAGE AT X, X+1 & SXPAGE+1,X
F6D1:              *
F6D1:              ************************************************************
F6D1:              *
F6D1:        F6D1  GETBUFADR1 EQU   *
F6D1:29 40                    AND   #$40
F6D3:D0 04   F6D9             BNE   GTB1010
F6D5:A9 00                    LDA   #0               ; "FIXED" BUFFER
F6D7:F0 03   F6DC             BEQ   GTB1020          ; ALWAYS TAKEN
F6D9:B9 91 F5      GTB1010    LDA   ADRL.T,Y         ; "FLOATING" BUFFER
F6DC:95 00         GTB1020    STA   0,X
F6DE:B9 80 F5                 LDA   ADRH.T,Y
F6E1:95 01                    STA   1,X
F6E3:B9 6F F5                 LDA   XBYTE.T,Y
F6E6:09 80                    ORA   #$80             ; ENSURE $7F->$8F
F6E8:9D 01 00                 STA   SXPAGE+1,X
F6EB:60                       RTS
```

```
F6EC:              ************************************************************
F6EC:              *
F6EC:              * CHKBUF
F6EC:              *
F6EC:              * CHECK BUFFER.  FETCHES THE FIRST BYTE OF EACH BUFFER
F6EC:              * REFERENCED DURING THE CURRENT SYSTEM CALL AND PLACES IT
F6EC:              * IN CHK.T(BUF#).
F6EC:              *
F6EC:              * INPUT:  BUFREF TABLE
F6EC:              *         BUFFER TABLE
F6EC:              * OUTPUT: EMPTY BUFREF TABLE
F6EC:              *         BUFFER TABLE'S CHECK BYTES UPDATED
F6EC:              *         Z REG:=$18
F6EC:              * ERROR:  NONE.
F6EC:              *
F6EC:              ************************************************************
F6EC:              *
F6EC:        F6EC  CHKBUF     EQU   *
F6EC:AC B3 F5                 LDY   BUFREF           ; PICK UP COUNT
F6EF:F0 1E   F70F             BEQ   CHKB.EXIT        ; EXIT IF BUFREF EMPTY
F6F1:              *
F6F1:A9 18                    LDA   #$18             ; ENSURE SOS ZPAGE SWITCHED IN
F6F3:8D D0 FF                 STA   Z.REG
F6F6:              *
F6F6:              * UPDATE THE CHECK BYTE OF EACH BUF# IN THE BUFREF TABLE
F6F6:              *
F6F6:A2 70         CHKB010    LDX   #>SOURCE
F6F8:B9 B3 F5                 LDA   BUFREF,Y
F6FB:A8                       TAY
F6FC:B9 5E F5                 LDA   PGCT.T,Y
F6FF:20 D1 F6                 JSR   GETBUFADR1       ; PUT BUF#S ADR ON ZPAGE
F702:A1 00                    LDA   ($0,X)
F704:99 A2 F5                 STA   CHK.T,Y
F707:CE B3 F5                 DEC   BUFREF
F70A:AC B3 F5                 LDY   BUFREF
F70D:D0 E7   F6F6             BNE   CHKB010          ; IF COUNT<>0 THEN PROCESS NEXT BUF# IN BUFREF TABLE
F70F:              *
F70F:60            CHKB.EXIT  RTS                    ; BUFREF TABLE IS EMPTY (COUNT=0)
```

```
F710:              ************************************************************
F710:              *
F710:              * RELBUF
F710:              *
F710:              * INPUT:  BUFNUM   (A)
F710:              * OUTPUT: NONE.
F710:              * ERROR:  "BADBUFNUM" SYSDEATH
F710:              *
F710:              * THIS ROUTINE RELEASES THE BUFFER ENTRY, CALLS FIND.SEG TO
F710:              * RELEASE THE CORRESPONDING MEMORY SEGMENT, AND CALLS
F710:              * BUFCOMPACT TO PERFORM BUFFER COMPACTION.
F710:              *
F710:              ************************************************************
F710:              *
F710:        F710  RELBUF     EQU   *
F710:              *
F710:              * IF BUF# OUT OF RANGE OR BUF.TBL(BUF#)=FREE
F710:              * THEN FATAL ERR
F710:              *
F710:A8                       TAY
F711:F0 25   F738             BEQ   RLBF.ERR
F713:C0 11                    CPY   #BUF.CNT
F715:B0 21   F738             BCS   RLBF.ERR
F717:B9 5E F5                 LDA   PGCT.T,Y
F71A:30 1C   F738             BMI   RLBF.ERR
F71C:              *
F71C:              * MARK BUF.TBL(BUF#)=FREE
F71C:              *
F71C:09 80                    ORA   #ISFREE
F71E:99 5E F5                 STA   PGCT.T,Y
F721:              *
F721:              * IF BUF.TBL(BUF#)=FIXED THEN EXIT
F721:              *
F721:29 40                    AND   #ISFIXED
F723:D0 11   F736             BNE   RLBF.EXIT
F725:              *
F725:              * OTHERWISE CALL MEMORY MGR TO RELEASE BUFFER'S MEMORY SEG
F725:              *
F725:A9 05                    LDA   #RELSEG
F727:85 60                    STA   REQCODE
F729:              *
F729:B9 91 F5                 LDA   SEG.T,Y
F72C:85 61                    STA   RLS.NUM
F72E:              *
F72E:20 00 00                 JSR   MMGR
F731:B0 05   F738             BCS   RLBF.ERR         ; ANY ERR IS FATAL
F733:              *
F733:              * AND COMPACT BUFFERS
F733:              *
F733:20 3D F7                 JSR   BUFCOMPACT
F736:              *
F736:18            RLBF.EXIT  CLC
F737:60                       RTS
F738:              *
F738:A9 00         RLBF.ERR   LDA   #BADBUFNUM
F73A:20 00 00                 JSR   SYSDEATH
```

```
F73D:              ************************************************************
F73D:              *
F73D:              * BUFCOMPACT
F73D:              *
F73D:              * THIS ROUTINE IS RESPONSIBLE FOR PACKING ALL SOS BUFFERS UP
F73D:              * AGAINST THE HIGHEST AVAILABLE FREE MEMORY.  COULD IMPROVE THE
F73D:              * EFFICIENCY OF THIS COMPACTION CYCLE BY NOT RELEASING THE "RELEASED" BUFFER
F73D:              * UNTIL IT IS KNOWN THAT ANOTHER BUFFER WILL NOT BE MOVED INTO ITS LOC.
F73D:              *
F73D:              ************************************************************
F73D:              *
F73D:        F73D  BUFCOMPACT EQU   *
F73D:              *
F73D:              * FIND THE FLOATING BUFFER IN BUF.TBL WITH THE LOWEST ADDRESS.
F73D:              *
F73D:A0 00         BUFC010    LDY   #0
F73F:A2 10                    LDX   #BUF.CNT-1
F741:              *
F741:BD 5E F5      BUFC020    LDA   PGCT.T,X
F744:29 C0                    AND   #$C0             ; STRIP OUT PAGE COUNT BITS
F746:D0 10   F758             BNE   BUFC030
F748:              *
F748:BD 80 F5                 LDA   ADRH.T,X
F74B:D9 80 F5                 CMP   ADRH.T,Y
F74E:BD 6F F5                 LDA   XBYTE.T,X
F751:F9 6F F5                 SBC   XBYTE.T,Y
F754:B0 02   F758             BCS   BUFC030
F756:              *
F756:8A                       TXA                    ; SMALLER BUFFER FOUND, SAVE IN Y
F757:A8                       TAY
F758:              *
F758:CA            BUFC030    DEX
F759:D0 E6   F741             BNE   BUFC020
F75B:              *
F75B:              * IF NO BUFFER FOUND THEN DONE
F75B:              *
F75B:98                       TYA
F75C:D0 03   F761             BNE   BUFC040
F75E:4C ED F7                 JMP   BUFC.EXIT
F761:8C 5D F5      BUFC040    STY   BUFC.BNUM        ; OTHERWISE SAVE BUF# IN Y REG.
F764:              *
F764:              * CALL FIND.SEG:  FINDS HIGHEST AVAILABLE FREE MEMORY
F764:              *
F764:B9 5E F5                 LDA   PGCT.T,Y
F767:29 3F                    AND   #$3F             ; STRIP OUT "FREE","FIXED" FLAGS
F769:18                       CLC
F76A:69 01                    ADC   #1
F76C:20 F9 F7                 JSR   FSEG
F76F:B0 7C   F7ED             BCS   BUFC.EXIT        ; DONE IF NO FREE SEG FOUND
F771:              *
F771:              * CONVERT BASE.BKPG TO BUFFER ADR
F771:              *
F771:AE 54 F5                 LDX   F.BASEX          ; BASE BANK
F774:AC 55 F5                 LDY   F.BASEX+1        ; BASE PAGE
F777:20 51 F8                 JSR   CNVRT.ADR
F77A:8E 54 F5                 STX   F.BASEX          ; XBYTE
```

```
F77D:8C 55 F5                 STY   F.BASEX+1        ; ADRH
F780:              *
F780:              * IF NEW SEG'S BASE < CURRENT BUFFER'S BASE ADR THEN DONE
F780:              *
F780:AC 5D F5                 LDY   BUFC.BNUM
F783:B9 80 F5                 LDA   ADRH.T,Y
F786:85 71                    STA   SOURCE+1
F788:CD 55 F5                 CMP   F.BASEX+1
F78B:B9 6F F5                 LDA   XBYTE.T,Y
F78E:8D 71 00                 STA   SXPAGE+SOURCE+1
F791:ED 54 F5                 SBC   F.BASEX
F794:B0 49   F7DF             BCS   BUFC.EXIT1
F796:              *
F796:              * MOVE DATA FROM CURRENT BUFFER TO NEW BUFFER
F796:              *
F796:AE 54 F5                 LDX   F.BASEX
F799:8E 73 00                 STX   SXPAGE+DEST+1
F79C:AC 55 F5                 LDY   F.BASEX+1
F79F:84 73                    STY   DEST+1
F7A1:A9 00                    LDA   #0
F7A3:85 70                    STA   SOURCE
F7A5:85 72                    STA   DEST
F7A7:              *
F7A7:A8                       TAY
F7A8:AE 52 F5                 LDX   F.PGCTX
F7AB:B1 70         BUFC200    LDA   (SOURCE),Y       ; MOVE LOOP
F7AD:91 72                    STA   (DEST),Y
F7AF:88                       DEY
F7B0:D0 F9   F7AB             BNE   BUFC200
F7B2:E6 71                    INC   SOURCE+1
F7B4:E6 73                    INC   DEST+1
F7B6:CA                       DEX
F7B7:D0 F2   F7AB             BNE   BUFC200
F7B9:              *
F7B9:              * UPDATE BUF.TBL(BUF#)
F7B9:              *
F7B9:AC 5D F5                 LDY   BUFC.BNUM
F7BC:AD 54 F5                 LDA   F.BASEX
F7BF:99 6F F5                 STA   XBYTE.T,Y
F7C2:AD 55 F5                 LDA   F.BASEX+1
F7C5:99 80 F5                 STA   ADRH.T,Y
F7C8:              *
F7C8:BE 91 F5                 LDX   SEG.T,Y
F7CB:AD 58 F5                 LDA   F.NUMX
F7CE:99 91 F5                 STA   SEG.T,Y
F7D1:              *
F7D1:              * AND RELEASE OLD MEMORY SEGMENT
F7D1:              *
F7D1:86 61                    STX   RLS.NUM
F7D3:A9 05                    LDA   #RELSEG
F7D5:85 60                    STA   REQCODE
F7D7:20 00 00                 JSR   MMGR
F7DA:B0 18   F7F4             BCS   BUFC.ERR
F7DC:              *
F7DC:4C 3D F7                 JMP   BUFC010          ; REPEAT COMPACTION CYCLE
F7DF:              *
```

```
F7DF:                *
F7DF:AE 58 F5        BUFC.EXIT1 LDX   F.NUMX           ; DONE,
F7E2:86 61                      STX   RLS.NUM          ; RELEASE SEG BEFORE EXIT
F7E4:A9 05                      LDA   #RELSEG
F7E6:85 60                      STA   REQCODE
F7E8:20 00 00                   JSR   MMGR
F7EB:B0 07   F7F4               BCS   BUFC.ERR
F7ED:                *
F7ED:A9 00           BUFC.EXIT  LDA   #0
F7EF:8D 00 00                   STA   SERR             ; MASK OUT ANY ERROR FROM MEMORY MGR
F7F2:18                         CLC
F7F3:60                         RTS                    ; NORMAL EXIT
F7F4:                *
F7F4:                *
F7F4:A9 00           BUFC.ERR   LDA   #BADBUFNUM
F7F6:20 00 00                   JSR   SYSDEATH
```

```
F7F9:              ************************************************************
F7F9:              *
F7F9:              * FSEG
F7F9:              *
F7F9:              * INPUT:  PAGE.CNT (A)
F7F9:              * OUTPUT: PAGE.CNT (A) UNCHANGED IF FIND.SEG SUCCESSFUL
F7F9:              * ERROR:  CARRY SET "UNABLE TO FIND MEMORY SEG OF PAGE.CNT*256 BYTES"
F7F9:              *
F7F9:              * THIS ROUTINE BUILDS THE PARAMETERS FOR A FIND.SEG SYSTEM CALL
F7F9:              * AND THEN CALLS THE MEMORY MANAGER.
F7F9:              *
F7F9:              ************************************************************
F7F9:              *
F7F9:        F7F9  FSEG       EQU   *
F7F9:              *
F7F9:              * SETUP INPUT PARAMETERS FOR FIND.SEG CALL
F7F9:              *
F7F9:8D 52 F5                 STA   F.PGCTX
F7FC:A9 01                    LDA   #FINDSEG
F7FE:85 60                    STA   REQCODE
F800:A9 02                    LDA   #2
F802:85 61                    STA   SRCHMODE
F804:A9 04                    LDA   #4
F806:85 62                    STA   F.ID
F808:              *
F808:              * SETUP OUTPUT PARAMETER ADRESSES
F808:              *
F808:A9 52                    LDA   #>F.PGCTX
F80A:85 63                    STA   F.PGCT
F80C:A9 F5                    LDA   #<F.PGCTX
F80E:85 64                    STA   F.PGCT+1
F810:A9 54                    LDA   #>F.BASEX
F812:85 65                    STA   F.BASE
F814:A9 F5                    LDA   #<F.BASEX
F816:85 66                    STA   F.BASE+1
F818:A9 56                    LDA   #>F.LIMX
F81A:85 67                    STA   F.LIM
F81C:A9 F5                    LDA   #<F.LIMX
F81E:85 68                    STA   F.LIM+1
F820:A9 58                    LDA   #>F.NUMX
F822:85 69                    STA   F.NUM
F824:A9 F5                    LDA   #<F.NUMX
F826:85 6A                    STA   F.NUM+1
F828:              *
F828:A9 00                    LDA   #0
F82A:8D 53 F5                 STA   F.PGCTX+1
F82D:8D 64 00                 STA   SXPAGE+F.PGCT+1
F830:8D 66 00                 STA   SXPAGE+F.BASE+1
F833:8D 68 00                 STA   SXPAGE+F.LIM+1
F836:8D 6A 00                 STA   SXPAGE+F.NUM+1
F839:              *
F839:20 00 00                 JSR   MMGR
F83C:AD 52 F5                 LDA   F.PGCTX
F83F:              *
F83F:60                       RTS                    ; EXIT.  CARRY SET->ERR
```

```
F840:              ************************************************************
F840:              *
F840:              * GETFREE
F840:              *
F840:              * INPUT:  NONE
F840:              * OUTPUT: BUF# (X)
F840:              * ERROR:  "BUFTBLFULL" SYSERR
F840:              *
F840:              * THIS ROUTINE SEARCHES THE BUFFER TABLE, LOOKING FOR A FREE
F840:              * ENTRY.  IF FOUND, IT RETURNS THE BUFFER NUMBER, ELSE ERROR.
F840:              *
F840:              ************************************************************
F840:              *
F840:        F840  GETFREE    EQU   *
F840:A2 10                    LDX   #BUF.CNT-1
F842:BD 5E F5      GFR010     LDA   PGCT.T,X
F845:30 08   F84F             BMI   GFR.EXIT         ; FREE ENTRY FOUND
F847:CA                       DEX
F848:D0 F8   F842             BNE   GFR010
F84A:              *
F84A:A9 00                    LDA   #BUFTBLFULL
F84C:20 00 00                 JSR   SYSERR           ; ERR EXIT
F84F:              *
F84F:18            GFR.EXIT   CLC
F850:60                       RTS                    ; NORMAL EXIT
```

```
F851:              ************************************************************
F851:              *
F851:              * CNVRT.ADR
F851:              *
F851:              * INPUT:  BANK VALUE (X)
F851:              *         PAGE VALUE (Y)
F851:              * OUTPUT: XBYTE (X)
F851:              *         ADRH  (Y)
F851:              * ERROR:  NONE.
F851:              *
F851:              * THIS ROUTINE CONVERTS A BASE.BKPG PARM (MMGR) INTO A
F851:              * VIRTUAL POINTER
F851:              *
F851:              ************************************************************
F851:              *
F851:        F851  CNVRT.ADR  EQU   *
F851:              *
F851:              * IF PAGE <> $20 THEN GOTO L2
F851:              *
F851:C0 20                    CPY   #$20
F853:D0 0F   F864             BNE   CNVA020
F855:              *
F855:              * IF BANK <> 0 THEN GOTO L1
F855:              *
F855:8A                       TXA
F856:D0 04   F85C             BNE   CNVA010
F858:              *
F858:              * XBYTE=$8F
F858:              * ADRH:=PAGE
F858:              *
F858:A2 8F                    LDX   #$8F
F85A:30 11   F86D             BMI   CNVA.EXIT
F85C:              *
F85C:              * L1: XBYTE:=(BANK-1) ORA #$80
F85C:              *     ADRH:=#$80
F85C:              *
F85C:09 80         CNVA010    ORA   #$80
F85E:AA                       TAX
F85F:CA                       DEX
F860:A0 80                    LDY   #$80
F862:30 09   F86D             BMI   CNVA.EXIT
F864:              *
F864:              * L2: XBYTE:=BANK ORA #$80
F864:              *     ADRH:=ADRH-#$20
F864:              *
F864:8A            CNVA020    TXA
F865:09 80                    ORA   #$80
F867:AA                       TAX
F868:38                       SEC
F869:98                       TYA
F86A:E9 20                    SBC   #$20
F86C:A8                       TAY
F86D:              *
F86D:60            CNVA.EXIT  RTS
F86E:              *
```

```
F86E:              774            LST   ON
F86E:        F86E  775 ZZEND      EQU   *
F86E:        031C  776 ZZLEN      EQU   ZZEND-ZZORG
F86E:        0000  777            IFNE  ZZLEN-LENBUFMG
 S                 778            FAIL  2,"SOSORG        FILE IS INCORRECT FOR BUFMGR"
F86E:              779            FIN
```

```
 F580 ADRH.T         F591 ADRL.T        X0010 BADBUFNUM     X0011 BADBUFSIZ
X000E BADSYSBUF     ?2E00 BLABFMI        3200 BLABFM         6B52 BLABUFMG
 6955 BLACFM         5E99 BLADISK3       64D9 BLADMGR        68F4 BLAFMGR
?2CF8 BLAGLOB       ?2AF8 BLAINIT        55C0 BLAIPL         2000 BLALODR
?6E6E BLAMEMMG       5466 BLAOMSG        5466 BLAPATCH       665E BLASCMGR
 6404 BLASERR        5A8B BLAUMGR       N0011 BUF.CNT          05 BUF.SIZ
 F55E BUF.TBL        F55D BUFC.BNUM      F7F4 BUFC.ERR       F7ED BUFC.EXIT
 F7DF BUFC.EXIT1     F73D BUFC010        F741 BUFC020        F758 BUFC030
 F761 BUFC040        F7AB BUFC200        F73D BUFCOMPACT    NF5B3 BUFREF
   11 BUFREF.CNT    X000D BUFTBLFULL     F5A2 CHK.T          F70F CHKB.EXIT
 F6F6 CHKB010       NF6EC CHKBUF         F86D CNVA.EXIT      F85C CNVA010
 F864 CNVA020        F851 CNVRT.ADR     X0009 CXPAGE        X0008 CZPAGE
   72 DEST           F554 F.BASEX          65 F.BASE           62 F.ID
 F556 F.LIMX           67 F.LIM          F558 F.NUMX           69 F.NUM
   63 F.PGCT         F552 F.PGCTX          A0 F.TPARMX         01 FINDSEG
 F7F9 FSEG          NF686 GETBUFADR      F6D1 GETBUFADR1     F840 GETFREE
 F84F GFR.EXIT       F842 GFR010         F6D9 GTB1010        F6DC GTB1020
 F6CC GTBF.ERR       F69E GTBF010        F6A6 GTBF020        F6C2 GTBF030
   40 ISFIXED          80 ISFREE         2266 LENBFM        ?0400 LENBFMI
 031C LENBUFMG       01FD LENCFM         056B LENDISK3       0185 LENDMGR
   61 LENFMGR       ?01B2 LENINIT        04CB LENIPL         0AF8 LENLODR
?0751 LENMEMMG       015A LENOMSG          00 LENPATCH       0296 LENSCMGR
   D5 LENSERR        040E LENUMGR          60 M.TPARMX         40 MAXPGCT
X0006 MMGR             A5 OPEN.LIST      BC00 ORGBFM         B800 ORGBFMI
 F552 ORGBUFMG       F355 ORGCFM         E899 ORGDISK3       EED9 ORGDMGR
 FFBF ORGEND         F2F4 ORGFMGR       ?18FC ORGGLOB        28F8 ORGINIT
 DFC0 ORGIPL         1E00 ORGLODR        F86E ORGMEMMG       DE66 ORGOMSG
 DE66 ORGPATCH       F05E ORGSCMGR       EE04 ORGSERR        E48B ORGUMGR
X000C OUTOFMEM      NF55E PGCT.T        NF710 RELBUF           05 RELSEG
NF5C5 REQBUF           60 REQCODE       NF622 REQFXBUF       F738 RLBF.ERR
 F736 RLBF.EXIT        61 RLS.NUM        F55A RQB.BNUM       F618 RQB.ERR1
 F61D RQB.ERR2       F613 RQB.ERR        F559 RQB.PGCT       F5FA RQB010
 F55C RQFB.BNUM      F677 RQFB.ERR       F67C RQFB.ERR1      F681 RQFB.ERR2
 F55B RQFB.PGCT      F65B RQFB010        F591 SEG.T         X000B SERR
   70 SOURCE           61 SRCHMODE      X0007 SXPAGE        X000F SYSDEATH
X000A SYSERR        NF56F XBYTE.T        FFD0 Z.REG          F5C4 ZPAGEX
 F86E ZZEND          031C ZZLEN          F552 ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED   838
** FREE SPACE PAGE COUNT   80
```

```
SOURCE   FILE #01 =>MEMMGR.A.SRC
 INCLUDE FILE #02 =>SOSORG
SOURCE   FILE #03 =>MEMMGR.B.SRC
SOURCE   FILE #04 =>MEMMGR.C.SRC
```

```
0000:                             REL
0000:                             INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
```

```
0000:        6955      BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52      BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E      BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
F86E:        F86E                 ORG   ORGMEMMG
F86E:        F86E      ZZORG      EQU   *
F86E:                             MSB   OFF
F86E:                  ************************************************************
F86E:                  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
F86E:                  *                    ALL RIGHTS RESERVED
F86E:                  ************************************************************
F86E:                  *
F86E:                  * MEMORY MANAGER (VERSION = 1.10   )
F86E:                  *                (DATE    = 8/04/81)
F86E:                  *
F86E:                  * THIS MODULE CONTAINS ALL OF THE MEMORY MANAGEMENT SYSTEM
F86E:                  * CALLS SUPPORTED BY THE SARA OPERATING SYSTEM.  IT IS
F86E:                  * ALSO CALLED BY THE BUFFER MANAGER.
F86E:                  *
F86E:                  ************************************************************
F86E:                  *
F86E:        F952                 ENTRY MMGR
F86E:                  *
F86E:        0020                 ENTRY ST.CNT
F86E:        F86F                 ENTRY ST.ENTRY
F86E:        F86E                 ENTRY ST.FREE
F86E:        F890                 ENTRY ST.FLINK
F86E:        0040                 ENTRY VRT.LIM
F86E:                  *
F86E:        0000                 EXTRN SYSERR
F86E:        0000                 EXTRN BADSCNUM
F86E:        0000                 EXTRN BADBKPG
F86E:        0000                 EXTRN SEGRQDN
F86E:        0000                 EXTRN SEGTBLFULL
F86E:        0000                 EXTRN BADSEGNUM
F86E:        0000                 EXTRN SEGNOTFND
F86E:        0000                 EXTRN BADSRCHMODE
F86E:        0000                 EXTRN BADCHGMODE
F86E:        0000                 EXTRN BADPGCNT
```

```
F86E:                ************************************************************
F86E:                *
F86E:                * SEGMENT TABLE
F86E:                * (NOTE: ENTRY 0 IS NOT USED)
F86E:                *
F86E:                ************************************************************
F86E:                *
F86E:        0001   ST.FREE    DS    1                ; PTR TO FIRST FREE SEG TABLE ENTRY
F86F:        0001   ST.ENTRY   DS    1                ; PTR TO HIGHEST ALLOC SEG TABLE ENTRY
F870:        0007   ST.SIZ     EQU   7
F870:        0020   ST.CNT     EQU   32
F870:        00E0   ST.TBL     DS    ST.SIZ*ST.CNT
F950:        F870   ST.BLINK   EQU   ST.TBL           ; BACK LINK TO PREV ALLOC SEG ENTRY
F950:        F890   ST.FLINK   EQU   ST.BLINK+ST.CNT  ; FORWARD LINK     "
F950:        F8B0   ST.BASEL   EQU   ST.FLINK+ST.CNT  ; BASE BANK/PAGE
F950:        F8D0   ST.BASEH   EQU   ST.BASEL+ST.CNT
F950:        F8F0   ST.LIML    EQU   ST.BASEH+ST.CNT  ; LIMIT BANK/PAGE
F950:        F910   ST.LIMH    EQU   ST.LIML+ST.CNT
F950:        F930   ST.ID      EQU   ST.LIMH+ST.CNT   ; SEG ID
```

```
F950:             ************************************************************
F950:             *
F950:             * DATA DECLARATIONS
F950:             *
F950:             ************************************************************
F950:             *
F950:        0040 ZPAGE      EQU   $40              ; BEGINNING OF ZPAGE TEMP SPACE FOR MEMORY MANAGER
F950:        0000 VRT.BASE   EQU   $0               ; INTERNAL BK/PG PTR TO LOWEST VIRT PAGE
F950:        0040 VRT.LIM    EQU   ZPAGE+$0         ; &$1, INTERNAL BK/PG PTR TO HIGHEST VIRT PAGE
F950:        0780 PHY1BASE   EQU   $0780            ; BANK "F",PAGE "0"
F950:        079F PHY1LIM    EQU   $079F            ; BANK "F",PAGE "1F"
F950:        0820 PHY2BASE   EQU   $0820            ; BANK "10",PAGE "A0"
F950:        087F PHY2LIM    EQU   $087F            ; BANK "10",PAGE "FF"
F950:             *
F950:             * REQUEST.SEG DATA DECLARATIONS
F950:             *
F950:        0060 M.TPARMX   EQU   $60              ; BEGINNING ADDRESS OF MMGR SOS CALL PARMS
F950:        0060 M.RQCODE   EQU   M.TPARMX
F950:        0061 RQ.BASE    EQU   M.TPARMX+1       ; BASE.BANK/PAGE
F950:        0063 RQ.LIM     EQU   M.TPARMX+3       ; LIMIT.BANK/PAGE
F950:        0065 RQ.ID      EQU   M.TPARMX+5
F950:        0066 RQ.NUM     EQU   M.TPARMX+6
F950:             *
F950:        0042 RQ.REGION  EQU   ZPAGE+$2         ;VRT(0),PHY0(1),PHY1(2)
F950:             *
F950:             * FIND.SEG DATA DECLARATIONS
F950:             *
F950:        0061 SRCHMODE   EQU   M.TPARMX+1       ; SEARCH MODE (0,1,2)
F950:        0062 F.ID       EQU   M.TPARMX+2       ; SEG ID
F950:        0063 F.PGCT     EQU   M.TPARMX+3       ; PAGE COUNT (LO
F950:        0043 FX.PGCT    EQU   ZPAGE+$3         ; &$4, INTERNAL PAGE COUNT
F950:        0065 F.BASE     EQU   M.TPARMX+5       ; BASE.BANK/PAGE
F950:        0067 F.LIM      EQU   M.TPARMX+7       ; LIMIT.BANK/PAGE
F950:        0069 F.NUM      EQU   M.TPARMX+9       ; SEG NUM
F950:        0045 F.ERR      EQU   ZPAGE+$5         ; ERROR FLAG
F950:        0080 TRUE       EQU   $80
F950:        0000 FALSE      EQU   $0
F950:        0046 CFS.PGCT   EQU   ZPAGE+$6         ; &7, CURRENT FREE SEGMENT'S PAGE COUNT
F950:        0048 CFS.BASE   EQU   ZPAGE+$8         ; &9,        "             BASE.BANK/PAGE
F950:        004A CFS.LIM    EQU   ZPAGE+$A         ; &$B,       "             LIMIT.BANK/PAGE
F950:        004C CFS.BLINK  EQU   ZPAGE+$C         ;          "             BACK LINK
F950:        004D CFS.BASE0  EQU   ZPAGE+$D         ; &$E,     "             BASE (SMODE=0)
F950:        004F CFS.BASE1  EQU   ZPAGE+$F         ; &$10,     "             BASE (SMODE=1)
F950:        0051 CFS.NEXT   EQU   ZPAGE+$11        ;          "             NEXT ENTRY
F950:        0052 CFS.PREV   EQU   ZPAGE+$12        ;          "             PREV ENTRY
F950:        0053 CFS.PTR    EQU   ZPAGE+$13        ; &$14     "             POINTER TO NXT FREE PG
F950:        0055 BFS.PGCT   EQU   ZPAGE+$15        ; &$16, BIGGEST FREE SEGMENT'S PAGE COUNT
F950:        0057 BFS.BASE   EQU   ZPAGE+$17        ; &$18     "             BASE.BANK/PAGE
F950:        0059 BFS.LIM    EQU   ZPAGE+$19        ; &$1A     "             LIMIT.BANK/PAGE
F950:        005B BFS.BLINK  EQU   ZPAGE+$1B        ;          "             BACK LINK
F950:             *
F950:             * CHANGE.SEG DATA DECLARATIONS
F950:             *
F950:        0061 CHG.NUM    EQU   M.TPARMX+1       ; SEGNUM PARM
F950:        0062 CHG.MODE   EQU   M.TPARMX+2       ; CHANGE MODE PARM
F950:        0063 CHG.PGCT   EQU   M.TPARMX+3       ; PAGE COUNT PARM
```

```
F950:        005C  CHG.PGCTX  EQU   ZPAGE+$1C        ; &$1D, INTERNAL STORE FOR PGCT
F950:        005E  CHG.NEW    EQU   ZPAGE+$1E        ; &$1F, BANK/PAGE OF SEG'S NEW LIMIT OR BASE
F950:              *
F950:              * GET.SEG.INFO DATA DECLARATIONS
F950:              *
F950:        0061  GSI.NUM    EQU   M.TPARMX+1
F950:        0062  GSI.BASE   EQU   M.TPARMX+2
F950:        0064  GSI.LIM    EQU   M.TPARMX+4
F950:        0066  GSI.PGCT   EQU   M.TPARMX+6
F950:        0068  GSI.ID     EQU   M.TPARMX+8
F950:              *
F950:              * GET.SEG.NUM DATA DECLARATIONS
F950:              *
F950:        0061  GSN.BKPG   EQU   M.TPARMX+1
F950:        0063  GSN.NUM    EQU   M.TPARMX+3
F950:              *
F950:              * RELEASE.SEG DATA DECLARATIONS
F950:              *
F950:        0061  RLS.NUM    EQU   M.TPARMX+1       ; SEG NUM
F950:              *
F950:              * REGION - DATA DECLARATIONS
F950:              *
F950:        0002  RGN.BKPG   DS    2                ; TEMP CONTAINER FOR BANK/PAGE
```

```
F952:              ************************************************************
F952:              *
F952:              * MMGR
F952:              *
F952:              * THIS ROUTINE IS THE MAIN ENTRANCE TO THE MEMORY MANAGER
F952:              * MODULE.  IT FUNCTIONS AS A SWITCH, BASED UPON THE RECEIVED
F952:              * REQUEST CODE, TO TRANSFER CONTROL TO THE ROUTINE THAT
F952:              * HANDLES THE SPECIFIC SYSTEM CALL.
F952:              *
F952:              ************************************************************
F952:              *
F952:        F952  MMGR       EQU   *
F952:A5 60                    LDA   M.RQCODE
F954:F0 19   F96F             BEQ   MMGR010          ; "REQ.SEG"
F956:C9 01                    CMP   #1
F958:F0 18   F972             BEQ   MMGR020          ; "FIND.SEG"
F95A:C9 02                    CMP   #2
F95C:F0 17   F975             BEQ   MMGR030          ; "CHANGE.SEG"
F95E:C9 03                    CMP   #3
F960:F0 16   F978             BEQ   MMGR040          ; "GET.SEG.INFO"
F962:C9 04                    CMP   #4
F964:F0 15   F97B             BEQ   MMGR050          ; "GET.SEG.NUM"
F966:C9 05                    CMP   #5
F968:F0 14   F97E             BEQ   MMGR060          ; "RELEASE.SEG"
F96A:              *
F96A:A9 00                    LDA   #BADSCNUM
F96C:20 00 00                 JSR   SYSERR
F96F:              *
F96F:4C 81 F9      MMGR010    JMP   REQ.SEG
F972:4C 23 FA      MMGR020    JMP   FIND.SEG
F975:4C 2F FC      MMGR030    JMP   CHG.SEG
F978:4C C8 FD      MMGR040    JMP   GET.SEG.INFO
F97B:4C 27 FE      MMGR050    JMP   GET.SEG.NUM
F97E:4C 67 FE      MMGR060    JMP   RELEASE.SEG
```

```
F981:              ************************************************************
F981:              *
F981:              * REQUEST.SEG(IN.BASE.BANKPAGE,LIMIT.BANKPAGE,SEGID; OUT.SEGNUM)
F981:              *
F981:              ************************************************************
F981:              *
F981:        F981  REQ.SEG    EQU   *
F981:              *
F981:              * CONVERT CALLER'S BASE.BANK/PAGE TO INTERNAL FMT
F981:              *
F981:A6 61                    LDX   RQ.BASE
F983:A4 62                    LDY   RQ.BASE+1
F985:20 C2 FE                 JSR   CNVRT.IBP
F988:90 01   F98B             BCC   RQ005
F98A:              *
F98A:60            RQ.ERR     RTS                    ; ERR EXIT - INVALID BANK/PAGE
F98B:              *
F98B:86 61         RQ005      STX   RQ.BASE
F98D:84 62                    STY   RQ.BASE+1
F98F:85 42                    STA   RQ.REGION
F991:              *
F991:              * CONVERT CALLER'S LIMIT.BANK/PAGE TO INTERNAL FMT
F991:              *
F991:A6 63                    LDX   RQ.LIM
F993:A4 64                    LDY   RQ.LIM+1
F995:20 C2 FE                 JSR   CNVRT.IBP
F998:B0 F0   F98A             BCS   RQ.ERR           ; ERR - INVALID BANK/PAGE
F99A:86 63                    STX   RQ.LIM
F99C:84 64                    STY   RQ.LIM+1
F99E:              *
F99E:              * IF BASE AND LIMIT ARE IN DIFFERENT REGIONS THEN ERR
F99E:              *
F99E:C5 42                    CMP   RQ.REGION
F9A0:D0 72   FA14             BNE   RQ.ERR1          ; ERR - INVALID BANK/PAGE PAIR
F9A2:              * IF CALLER'S BASE > LIMIT THEN ERR
F9A2:              *
F9A2:A5 63                    LDA   RQ.LIM
F9A4:C5 61                    CMP   RQ.BASE
F9A6:A5 64                    LDA   RQ.LIM+1
F9A8:E5 62                    SBC   RQ.BASE+1
F9AA:90 68   FA14             BCC   RQ.ERR1          ; ERR - INVALID BANK/PAGE PAIR
F9AC:              *
F9AC:              * PREV SEGNUM:=NULL; NEXT SEGNUM:=FIRST ENTRY
F9AC:              *
F9AC:A2 00                    LDX   #0
F9AE:AC 6F F8                 LDY   ST.ENTRY         ; NOTE: PREV/NEXT CARRIED IN X & Y REGISTERS
F9B1:              *
F9B1:              * IF NO SEGS IN SEG TABLE THEN ALLOCATE REQUESTED SEG
F9B1:              *
F9B1:F0 3A   F9ED             BEQ   RQ030
F9B3:              *
F9B3:              * IF FIRST SEG IN SEG TABLE BELOW REQUESTED SEG
F9B3:              * THEN ALLOCATE SEG
F9B3:              *
F9B3:B9 F0 F8                 LDA   ST.LIML,Y
F9B6:C5 61                    CMP   RQ.BASE
```

```
F9B8:B9 10 F9                 LDA   ST.LIMH,Y
F9BB:E5 62                    SBC   RQ.BASE+1
F9BD:90 2E   F9ED             BCC   RQ030
F9BF:              *
F9BF:              * ADVANCE TO NEXT SEG ENTRY
F9BF:              *
F9BF:98            RQ010      TYA
F9C0:AA                       TAX
F9C1:B9 90 F8                 LDA   ST.FLINK,Y
F9C4:A8                       TAY
F9C5:              *
F9C5:              * IF THERE IS NO NEXT SEG ENTRY
F9C5:              *   IF REQUESTED SEG IS BELOW PREV SEG
F9C5:              *      THEN ALLOCATE REQ SEG
F9C5:              *      ELSE ERR
F9C5:              *
F9C5:D0 0E   F9D5             BNE   RQ020
F9C7:A5 63                    LDA   RQ.LIM
F9C9:DD B0 F8                 CMP   ST.BASEL,X
F9CC:A5 64                    LDA   RQ.LIM+1
F9CE:FD D0 F8                 SBC   ST.BASEH,X
F9D1:90 1A   F9ED             BCC   RQ030
F9D3:              *
F9D3:B0 44   FA19             BCS   RQ.ERR2          ; ERR - SEGMENT REQUEST DENIED
F9D5:              *
F9D5:              * IF REQUESTED LIMIT >= PREV SEG'S BASE THEN ERR
F9D5:              *
F9D5:A5 63         RQ020      LDA   RQ.LIM
F9D7:DD B0 F8                 CMP   ST.BASEL,X
F9DA:A5 64                    LDA   RQ.LIM+1
F9DC:FD D0 F8                 SBC   ST.BASEH,X
F9DF:B0 38   FA19             BCS   RQ.ERR2          ; ERR - SEGMENT REQUEST DENIED
F9E1:              *
F9E1:              * IF REQUESTED BASE > NEXT SEG'S LIMIT
F9E1:              *    THEN ALLOCATE REQUESTED SEGMENT
F9E1:              *
F9E1:B9 F0 F8                 LDA   ST.LIML,Y
F9E4:C5 61                    CMP   RQ.BASE
F9E6:B9 10 F9                 LDA   ST.LIMH,Y
F9E9:E5 62                    SBC   RQ.BASE+1
F9EB:B0 D2   F9BF             BCS   RQ010            ; NO, ADVANCE TO NEXT SEGMENT
F9ED:              *
F9ED:8A            RQ030      TXA                    ; ALLOCATE REQUESTED SEGMENT
F9EE:20 76 FF                 JSR   GET.FREE
F9F1:B0 2B   FA1E             BCS   RQ.ERR3          ; ERR - SEG TABLE FULL
F9F3:              *
F9F3:              * ENTER BASE,LIMIT AND ID IN NEW SEG ENTRY
F9F3:              *
F9F3:AA                       TAX
F9F4:A5 61                    LDA   RQ.BASE
F9F6:9D B0 F8                 STA   ST.BASEL,X
F9F9:A5 62                    LDA   RQ.BASE+1
F9FB:9D D0 F8                 STA   ST.BASEH,X
F9FE:              *
F9FE:A5 63                    LDA   RQ.LIM
FA00:9D F0 F8                 STA   ST.LIML,X
```

```
FA03:A5 64                   LDA   RQ.LIM+1
FA05:9D 10 F9                STA   ST.LIMH,X
FA08:              *
FA08:A5 65                   LDA   RQ.ID
FA0A:9D 30 F9                STA   ST.ID,X
FA0D:              *
FA0D:              * RETURN NEW SEG NUM TO CALLER AND RETURN
FA0D:              *
FA0D:A0 00                   LDY   #0
FA0F:8A                      TXA
FA10:91 66                   STA   (RQ.NUM),Y
FA12:              *
FA12:18                      CLC
FA13:60                      RTS                     ; NORMAL EXIT
FA14:              *
FA14:A9 00         RQ.ERR1   LDA   #BADBKPG
FA16:20 00 00                JSR   SYSERR            ; ERR EXIT
FA19:A9 00         RQ.ERR2   LDA   #SEGRQDN
FA1B:20 00 00                JSR   SYSERR            ; ERR EXIT
FA1E:              *
FA1E:A9 00         RQ.ERR3   LDA   #SEGTBLFULL
FA20:20 00 00                JSR   SYSERR            ; ERR EXIT
```

```
FA23:              ************************************************************
FA23:              *
FA23:              * FIND.SEG(IN.SRCHMODE,SEGID; INOUT.PAGECT;
FA23:              *          OUT.BASE.BKPG,LIMIT.BKPG,SEGNUM)
FA23:              *
FA23:              ************************************************************
FA23:              *
FA23:        FA23  FIND.SEG   EQU   *
FA23:              *
FA23:              * RETRIEVE PAGE COUNT PARAMETER AND CLEAR ERR FLAG
FA23:              *
FA23:A0 00                    LDY   #0
FA25:B1 63                    LDA   (F.PGCT),Y
FA27:85 43                    STA   FX.PGCT
FA29:C8                       INY
FA2A:B1 63                    LDA   (F.PGCT),Y
FA2C:85 44                    STA   FX.PGCT+1
FA2E:              *
FA2E:D0 09   FA39             BNE   FIND001
FA30:A5 43                    LDA   FX.PGCT
FA32:D0 05   FA39             BNE   FIND001
FA34:A9 00                    LDA   #BADPGCNT        ; ERR, PAGECT=0, EXIT
FA36:20 00 00                 JSR   SYSERR
FA39:              *
FA39:A9 00         FIND001    LDA   #FALSE
FA3B:85 45                    STA   F.ERR
FA3D:              *
FA3D:              * IF SEARCH MODE>2 THEN ERR
FA3D:              *
FA3D:A5 61                    LDA   SRCHMODE
FA3F:C9 03                    CMP   #3
FA41:90 05   FA48             BCC   FIND005
FA43:A9 00                    LDA   #BADSRCHMODE
FA45:20 00 00                 JSR   SYSERR           ; ERR EXIT
FA48:              *
FA48:              * INITIALIZE NEXT FREE SEGMENT SUBROUTINE,
FA48:              * AND BIGGEST FREE SEGMENT PAGE COUNT
FA48:              *
FA48:20 F1 FA      FIND005    JSR   NXTFRSEG.I
FA4B:A9 00                    LDA   #0
FA4D:85 55                    STA   BFS.PGCT
FA4F:85 56                    STA   BFS.PGCT+1
FA51:              *
FA51:              * GET NEXT FREE SEGMENT
FA51:              *
FA51:20 31 FB      FIND010    JSR   NXTFRSEG
FA54:90 09   FA5F             BCC   FIND015          ; PROCESS FREE SEGMENT
FA56:              *
FA56:              * NO MORE FREE SEGMENTS LEFT
FA56:              * RETURN BIGGEST FREE SEGMENT FOUND
FA56:              * ALONG WITH ERR
FA56:              *
FA56:A9 80                    LDA   #TRUE
FA58:85 45                    STA   F.ERR
FA5A:A2 00                    LDX   #0               ; SEG#:=0
FA5C:4C B8 FA                 JMP   FIND070
```

```
FA5F:              *
FA5F:              * FREE SEGMENT FOUND.
FA5F:              *   IF FREE SEGMENT > BIGGEST FREE SEGMENT THEN BFS:=CFS
FA5F:              *
FA5F:A5 55         FIND015    LDA   BFS.PGCT
FA61:C5 46                    CMP   CFS.PGCT
FA63:A5 56                    LDA   BFS.PGCT+1
FA65:E5 47                    SBC   CFS.PGCT+1
FA67:B0 09   FA72             BCS   FIND030
FA69:              *
FA69:A2 06                    LDX   #6
FA6B:B5 46         FIND020    LDA   CFS.PGCT,X
FA6D:95 55                    STA   BFS.PGCT,X
FA6F:CA                       DEX
FA70:10 F9   FA6B             BPL   FIND020
FA72:              *
FA72:              * IF BFS.PGCT<F.PGCT THEN GET NEXT FREE SEGMENT
FA72:              *
FA72:A5 55         FIND030    LDA   BFS.PGCT
FA74:C5 43                    CMP   FX.PGCT
FA76:A5 56                    LDA   BFS.PGCT+1
FA78:E5 44                    SBC   FX.PGCT+1
FA7A:90 D5   FA51             BCC   FIND010
FA7C:              *
FA7C:              * BFS.BASE:=BFS.LIM-FX.PGCT+1
FA7C:              * BFS.PGCT:=FX.PGCT
FA7C:              *
FA7C:A5 59                    LDA   BFS.LIM
FA7E:E5 43                    SBC   FX.PGCT
FA80:85 57                    STA   BFS.BASE
FA82:A5 5A                    LDA   BFS.LIM+1
FA84:E5 44                    SBC   FX.PGCT+1
FA86:85 58                    STA   BFS.BASE+1
FA88:E6 57                    INC   BFS.BASE
FA8A:D0 02   FA8E             BNE   FIND050
FA8C:E6 58                    INC   BFS.BASE+1
FA8E:              *
FA8E:A5 43         FIND050    LDA   FX.PGCT
FA90:85 55                    STA   BFS.PGCT
FA92:A5 44                    LDA   FX.PGCT+1
FA94:85 56                    STA   BFS.PGCT+1
FA96:              *
FA96:              * DELINK ENTRY FROM FREE LIST, AND LINK
FA96:              * IT INTO SEGMENT LIST
FA96:              *
FA96:A5 5B                    LDA   BFS.BLINK
FA98:20 76 FF                 JSR   GET.FREE
FA9B:90 01   FA9E             BCC   FIND060
FA9D:60                       RTS                    ; ERR - SEG TABLE FULL
FA9E:              *
FA9E:              * ST.ID(NEW):=F.ID
FA9E:              * ST.BASE(NEW):=BFS.BASE
FA9E:              * ST.LIM(NEW):=BFS.LIM
FA9E:              *
FA9E:AA            FIND060    TAX
FA9F:A5 62                    LDA   F.ID
```

```
FAA1:9D 30 F9                 STA   ST.ID,X
FAA4:              *
FAA4:A5 57                    LDA   BFS.BASE
FAA6:9D B0 F8                 STA   ST.BASEL,X
FAA9:A5 58                    LDA   BFS.BASE+1
FAAB:9D D0 F8                 STA   ST.BASEH,X
FAAE:              *
FAAE:A5 59                    LDA   BFS.LIM
FAB0:9D F0 F8                 STA   ST.LIML,X
FAB3:A5 5A                    LDA   BFS.LIM+1
FAB5:9D 10 F9                 STA   ST.LIMH,X
FAB8:              *
FAB8:              * RETURN SEGNUM, PAGE COUNT, BASE BANK/PAGE, AND LIMIT BANK/PAGE
FAB8:              * TO CALLER
FAB8:A0 00         FIND070    LDY   #0
FABA:8A                       TXA
FABB:91 69                    STA   (F.NUM),Y
FABD:              *
FABD:A5 55                    LDA   BFS.PGCT
FABF:91 63                    STA   (F.PGCT),Y
FAC1:C8                       INY
FAC2:A5 56                    LDA   BFS.PGCT+1
FAC4:91 63                    STA   (F.PGCT),Y
FAC6:              *
FAC6:A6 57                    LDX   BFS.BASE
FAC8:A4 58                    LDY   BFS.BASE+1
FACA:20 05 FF                 JSR   CNVRT.XBP
FACD:98                       TYA
FACE:A0 01                    LDY   #1
FAD0:91 65                    STA   (F.BASE),Y
FAD2:88                       DEY
FAD3:8A                       TXA
FAD4:91 65                    STA   (F.BASE),Y
FAD6:              *
FAD6:A6 59                    LDX   BFS.LIM
FAD8:A4 5A                    LDY   BFS.LIM+1
FADA:20 05 FF                 JSR   CNVRT.XBP
FADD:98                       TYA
FADE:A0 01                    LDY   #1
FAE0:91 67                    STA   (F.LIM),Y
FAE2:88                       DEY
FAE3:8A                       TXA
FAE4:91 67                    STA   (F.LIM),Y
FAE6:              *
FAE6:A5 45                    LDA   F.ERR            ; IF ERR FLAG TRUE THEN REPORT IT.
FAE8:D0 02   FAEC             BNE   FIND.ERR
FAEA:              *
FAEA:18                       CLC
FAEB:60                       RTS                    ; NORMAL EXIT
FAEC:              *
FAEC:A9 00         FIND.ERR   LDA   #SEGRQDN
FAEE:20 00 00                 JSR   SYSERR           ; ERR EXIT
FAF1:                         CHN   MEMMGR.B.SRC
```

```
FAF1:                  ************************************************************
FAF1:                  *
FAF1:                  * NEXT FREE SEGMENT - INITIALIZATION
FAF1:                  *
FAF1:                  * INPUT:  SEGMENT TABLE
FAF1:                  * OUTPUT: CFS.PTR "1ST FREE BANK/PAGE IN VIRTUAL MEMORY
FAF1:                  *         CFS.PREV "PREVIOUS SEGMENT EXAMINED"
FAF1:                  *         CFS.NEXT "SEGMENT FOLLOWING CFS.PREV"
FAF1:                  * ERROR:  NONE (IF NO FREE BK/PG FOUND, THEN CFS.PTR="FFFF")
FAF1:                  *
FAF1:                  ************************************************************
FAF1:                  *
FAF1:        FAF1      NXTFRSEG.I EQU   *
FAF1:                  *
FAF1:                  * CFS.PTR := VRT.LIM
FAF1:                  * CFS.PREV := 0
FAF1:                  * CFS.NEXT := ST.ENTRY
FAF1:                  *
FAF1:AD 40 00                     LDA   >VRT.LIM
FAF4:85 53                        STA   CFS.PTR
FAF6:AD 41 00                     LDA   >VRT.LIM+1
FAF9:85 54                        STA   CFS.PTR+1
FAFB:                  *
FAFB:A9 00                        LDA   #0
FAFD:85 52                        STA   CFS.PREV
FAFF:                  *
FAFF:AE 6F F8                     LDX   ST.ENTRY
FB02:86 51                        STX   CFS.NEXT
FB04:                  *
FB04:                  * L0:  IF CFS.NEXT=0 THEN DONE
FB04:                  *
FB04:F0 2A   FB30      FRSGI010   BEQ   FRSGI.EXIT
FB06:                  *
FB06:                  * IF ST.LIM(CFS.NEXT)<=VRT.LIM THEN GOTO L1
FB06:                  *
FB06:AD 40 00                     LDA   >VRT.LIM
FB09:DD F0 F8                     CMP   ST.LIML,X
FB0C:AD 41 00                     LDA   >VRT.LIM+1
FB0F:FD 10 F9                     SBC   ST.LIMH,X
FB12:B0 0B   FB1F                 BCS   FRSGI020
FB14:                  *
FB14:                  * CFS.PREV:=CFS.NEXT
FB14:                  * CFS.NEXT:=ST.FLINK(CFS.NEXT)
FB14:                  * GOTO L0
FB14:                  *
FB14:86 52                        STX   CFS.PREV
FB16:BD 90 F8                     LDA   ST.FLINK,X
FB19:AA                           TAX
FB1A:86 51                        STX   CFS.NEXT
FB1C:4C 04 FB                     JMP   FRSGI010
FB1F:                  *
FB1F:                  * L1:  IF ST.LIM(CFS.NEXT)<VRT.LIM THEN DONE
FB1F:                  *
FB1F:BD F0 F8          FRSGI020   LDA   ST.LIML,X
FB22:CD 40 00                     CMP   >VRT.LIM
FB25:BD 10 F9                     LDA   ST.LIMH,X
```

```
FB28:ED 41 00                  SBC   >VRT.LIM+1
FB2B:90 03   FB30              BCC   FRSGI.EXIT
FB2D:               *
FB2D:               *
FB2D:20 05 FC                  JSR   NXTFRPG
FB30:               *
FB30:60             FRSGI.EXIT RTS                     ; NORMAL EXIT
```

```
FB31:              ************************************************************
FB31:              *
FB31:              * NEXT FREE SEGMENT
FB31:              *
FB31:              * INPUT:  SEG TABLE
FB31:              * OUTPUT: CFS.BLINK
FB31:              *         CFS.BASE
FB31:              *         CFS.LIMIT
FB31:              *         CFS.PGCT
FB31:              * OWN:    CFS.PREV
FB31:              *         CFS.NEXT
FB31:              *         CFS.PTR
FB31:              *
FB31:              * BUILDS A CANDIDATE FREE SEGMENT, WHOSE LIMIT BANK/PAGE =
FB31:              * THE CURRENT FREE PAGE (CFS.PTR).
FB31:              *
FB31:              ************************************************************
FB31:              *
FB31:        FB31  NXTFRSEG   EQU   *
FB31:              *
FB31:              * IF CFS.PTR="FFFF" THEN EXIT
FB31:              *
FB31:A5 54                    LDA   CFS.PTR+1
FB33:10 02   FB37             BPL   FRSG010
FB35:              *
FB35:38                       SEC
FB36:60                       RTS                    ; EXIT - NO MORE FREE SEGMENTS LEFT
FB37:              *
FB37:              * CFS.BLINK:=CFS.PREV
FB37:              * CFS.LIM:=CFS.PTR
FB37:              *
FB37:A5 52         FRSG010    LDA   CFS.PREV
FB39:85 4C                    STA   CFS.BLINK
FB3B:              *
FB3B:A5 53                    LDA   CFS.PTR
FB3D:85 4A                    STA   CFS.LIM
FB3F:A5 54                    LDA   CFS.PTR+1
FB41:85 4B                    STA   CFS.LIM+1
FB43:              *
FB43:              * IF CFS.NEXT=0 THEN CFS.BASE:=0
FB43:              *    ELSE CFS.BASE:=ST.LIM(CFS.NEXT)+1
FB43:              *
FB43:A5 51                    LDA   CFS.NEXT
FB45:D0 08   FB4F             BNE   FRSG020
FB47:A9 00                    LDA   #0
FB49:85 48                    STA   CFS.BASE
FB4B:85 49                    STA   CFS.BASE+1
FB4D:F0 11   FB60             BEQ   FRSG030
FB4F:              *
FB4F:A6 51         FRSG020    LDX   CFS.NEXT
FB51:18                       CLC
FB52:BD F0 F8                 LDA   ST.LIML,X
FB55:69 01                    ADC   #1
FB57:85 48                    STA   CFS.BASE
FB59:BD 10 F9                 LDA   ST.LIMH,X
FB5C:69 00                    ADC   #0
```

```
FB5E:85 49                    STA   CFS.BASE+1
FB60:              *
FB60:              * CFS.BASE0:=CFS.LIM AND $FF80
FB60:              *
FB60:A4 4B         FRSG030    LDY   CFS.LIM+1
FB62:84 4E                    STY   CFS.BASE0+1
FB64:A5 4A                    LDA   CFS.LIM
FB66:29 80                    AND   #$80
FB68:85 4D                    STA   CFS.BASE0
FB6A:              *
FB6A:              * CFS.BASE1:=CFS.BASE0-32K
FB6A:              *
FB6A:38                       SEC
FB6B:E9 80                    SBC   #$80
FB6D:85 4F                    STA   CFS.BASE1
FB6F:98                       TYA
FB70:E9 00                    SBC   #0
FB72:85 50                    STA   CFS.BASE1+1
FB74:B0 06   FB7C             BCS   FRSG035
FB76:A9 00                    LDA   #0
FB78:85 4F                    STA   CFS.BASE1
FB7A:85 50                    STA   CFS.BASE1+1
FB7C:              *
FB7C:              * IF CFS.BASE>=CFS.BASE0 THEN GOTO L1
FB7C:              *
FB7C:A5 48         FRSG035    LDA   CFS.BASE
FB7E:C5 4D                    CMP   CFS.BASE0
FB80:A5 49                    LDA   CFS.BASE+1
FB82:E5 4E                    SBC   CFS.BASE0+1
FB84:B0 27   FBAD             BCS   FRSG050
FB86:              *
FB86:              * IF SEARCH MODE=0 THEN CFS.BASE:=CFS.BASE0
FB86:              * GOTO L1
FB86:              *
FB86:A5 61                    LDA   SRCHMODE
FB88:D0 0B   FB95             BNE   FRSG040
FB8A:A5 4D                    LDA   CFS.BASE0
FB8C:85 48                    STA   CFS.BASE
FB8E:A5 4E                    LDA   CFS.BASE0+1
FB90:85 49                    STA   CFS.BASE+1
FB92:4C AD FB                 JMP   FRSG050
FB95:              *
FB95:              * IF CFS.BASE<CFS.BASE1 AND SEARCH MODE=1
FB95:              *    THEN CFS.BASE:=CFS.BASE1
FB95:              *
FB95:A5 48         FRSG040    LDA   CFS.BASE
FB97:C5 4F                    CMP   CFS.BASE1
FB99:A5 49                    LDA   CFS.BASE+1
FB9B:E5 50                    SBC   CFS.BASE1+1
FB9D:B0 0E   FBAD             BCS   FRSG050
FB9F:              *
FB9F:A5 61                    LDA   SRCHMODE
FBA1:C9 01                    CMP   #1
FBA3:D0 08   FBAD             BNE   FRSG050
FBA5:              *
FBA5:A5 4F                    LDA   CFS.BASE1
```

```
FBA7:85 48                    STA   CFS.BASE
FBA9:A5 50                    LDA   CFS.BASE1+1
FBAB:85 49                    STA   CFS.BASE+1
FBAD:              *
FBAD:              * L1:  CFS.PGCT:=CFS.LIM-CFS.BASE+1
FBAD:              *
FBAD:38            FRSG050    SEC
FBAE:A5 4A                    LDA   CFS.LIM
FBB0:E5 48                    SBC   CFS.BASE
FBB2:85 46                    STA   CFS.PGCT
FBB4:A5 4B                    LDA   CFS.LIM+1
FBB6:E5 49                    SBC   CFS.BASE+1
FBB8:85 47                    STA   CFS.PGCT+1
FBBA:E6 46                    INC   CFS.PGCT
FBBC:D0 02   FBC0             BNE   FRSG052
FBBE:E6 47                    INC   CFS.PGCT+1
FBC0:              *
FBC0:              * ADVANCE FREE PAGE POINTER TO NEXT FREE PAGE
FBC0:              *
FBC0:              * IF SEARCH MODE<>1 THEN L2:
FBC0:              *
FBC0:A5 61         FRSG052    LDA   SRCHMODE
FBC2:C9 01                    CMP   #1
FBC4:D0 19   FBDF             BNE   FRSG060
FBC6:              *
FBC6:              * IF CFS.BASE < CFS.BASE0 THEN CFS.PTR:=CFS.BASE0-1
FBC6:              *
FBC6:A5 48                    LDA   CFS.BASE
FBC8:C5 4D                    CMP   CFS.BASE0
FBCA:A5 49                    LDA   CFS.BASE+1
FBCC:E5 4E                    SBC   CFS.BASE0+1
FBCE:B0 0F   FBDF             BCS   FRSG060
FBD0:              *
FBD0:A4 4E                    LDY   CFS.BASE0+1
FBD2:A6 4D                    LDX   CFS.BASE0
FBD4:D0 01   FBD7             BNE   FRSG055
FBD6:88                       DEY
FBD7:CA            FRSG055    DEX
FBD8:86 53                    STX   CFS.PTR
FBDA:84 54                    STY   CFS.PTR+1
FBDC:              *
FBDC:4C 03 FC                 JMP   FRSG070          ; AND EXIT
FBDF:              * L2: CFS.PTR:=CFS.BASE-1
FBDF:              *
FBDF:38            FRSG060    SEC
FBE0:A5 48                    LDA   CFS.BASE
FBE2:E9 01                    SBC   #1
FBE4:85 53                    STA   CFS.PTR
FBE6:A5 49                    LDA   CFS.BASE+1
FBE8:E9 00                    SBC   #0
FBEA:85 54                    STA   CFS.PTR+1
FBEC:              *
FBEC:              * IF CFS.PTR="FFFF" OR CFS.NEXT=0 THEN EXIT
FBEC:              *
FBEC:90 15   FC03             BCC   FRSG070
FBEE:A5 51                    LDA   CFS.NEXT
```

```
FBF0:F0 11   FC03             BEQ   FRSG070
FBF2:              *
FBF2:              * IF CFS.PTR > ST.LIM(CFS.NEXT) THEN EXIT
FBF2:              *
FBF2:A6 51                    LDX   CFS.NEXT
FBF4:BD F0 F8                 LDA   ST.LIML,X
FBF7:C5 53                    CMP   CFS.PTR
FBF9:BD 10 F9                 LDA   ST.LIMH,X
FBFC:E5 54                    SBC   CFS.PTR+1
FBFE:90 03   FC03             BCC   FRSG070
FC00:              *
FC00:              * OTHERWISE, ADVANCE CFS PTR TO NEXT FREE PAGE BELOW NEXT
FC00:              * SEGMENT IN SEGMENT LIST
FC00:              *
FC00:20 05 FC                 JSR   NXTFRPG
FC03:              *
FC03:18            FRSG070    CLC
FC04:60                       RTS                    ; EXIT - FREE SEGMENT FOUND
```

```
FC05:              ************************************************************
FC05:              *
FC05:              * NEXT FREE PAGE
FC05:              *
FC05:              * "WALKS" THE FREE PAGE PTR (CFS.PTR) TO THE NEXT FREE PAGE
FC05:              * IMMEDIATELY BELOW THE CURRENT FREE SEGMENT.
FC05:              *
FC05:              ************************************************************
FC05:              *
FC05:        FC05  NXTFRPG    EQU   *
FC05:              *
FC05:              * L0: CFS.PTR:=ST.BASE(CFS.NEXT)-1
FC05:              *     IF CFS.PTR="FFFF" THEN DONE
FC05:              *
FC05:A6 51                    LDX   CFS.NEXT
FC07:38                       SEC
FC08:BD B0 F8                 LDA   ST.BASEL,X
FC0B:E9 01                    SBC   #1
FC0D:85 53                    STA   CFS.PTR
FC0F:BD D0 F8                 LDA   ST.BASEH,X
FC12:E9 00                    SBC   #0
FC14:85 54                    STA   CFS.PTR+1
FC16:90 16   FC2E             BCC   NFRPG.EXIT
FC18:              *
FC18:              * CFS.PREV:=CFS.NEXT
FC18:              * CFS.NEXT:=ST.FLINK(CFS.NEXT)
FC18:              *
FC18:86 52                    STX   CFS.PREV
FC1A:BD 90 F8                 LDA   ST.FLINK,X
FC1D:AA                       TAX
FC1E:86 51                    STX   CFS.NEXT
FC20:              *
FC20:              * IF CFS.NEXT=0 OR ST.LIM(CFS.NEXT)<CFS.PTR
FC20:              *    THEN DONE
FC20:              *    ELSE GOTO L0
FC20:              *
FC20:F0 0C   FC2E             BEQ   NFRPG.EXIT
FC22:BD F0 F8                 LDA   ST.LIML,X
FC25:C5 53                    CMP   CFS.PTR
FC27:BD 10 F9                 LDA   ST.LIMH,X
FC2A:E5 54                    SBC   CFS.PTR+1
FC2C:B0 D7   FC05             BCS   NXTFRPG
FC2E:              *
FC2E:60            NFRPG.EXIT RTS                    ; NORMAL EXIT
```

```
FC2F:             ***********************************************************
FC2F:             *
FC2F:             * CHANGE.SEG(IN.SEGNUM,CHG.MODE; INOUT.PAGECT) SYSTEM CALL
FC2F:             *
FC2F:             ***********************************************************
FC2F:             *
FC2F:        FC2F CHG.SEG    EQU   *
FC2F:             *
FC2F:             * MOVE CALLER'S PAGE COUNT TO INTERNAL BUFFER
FC2F:             *
FC2F:A0 00                   LDY   #0
FC31:B1 63                   LDA   (CHG.PGCT),Y
FC33:85 5C                   STA   CHG.PGCTX
FC35:C8                      INY
FC36:B1 63                   LDA   (CHG.PGCT),Y
FC38:85 5D                   STA   CHG.PGCTX+1
FC3A:             *
FC3A:             * IF SEG# OUT OF RANGE OR ST.FLINK(SEG#)=FREE THEN ERR
FC3A:             *
FC3A:A6 61                   LDX   CHG.NUM
FC3C:F0 09   FC47            BEQ   CHGS.ERR
FC3E:E0 20                   CPX   #ST.CNT
FC40:B0 05   FC47            BCS   CHGS.ERR
FC42:BD 90 F8                LDA   ST.FLINK,X
FC45:10 05   FC4C            BPL   CHGS005
FC47:             *
FC47:A9 00        CHGS.ERR   LDA   #BADSEGNUM
FC49:20 00 00                JSR   SYSERR           ; ERR EXIT
FC4C:             **********************************
FC4C:             * CASE OF CHANGE MODE
FC4C:             **********************************
FC4C:A4 62        CHGS005    LDY   CHG.MODE
FC4E:C0 01                   CPY   #1
FC50:90 0D   FC5F            BCC   CHGS010
FC52:F0 35   FC89            BEQ   CHGS020
FC54:C0 03                   CPY   #3
FC56:90 44   FC9C            BCC   CHGS030
FC58:F0 55   FCAF            BEQ   CHGS040
FC5A:             *
FC5A:A9 00                   LDA   #BADCHGMODE
FC5C:20 00 00                JSR   SYSERR           ; ERR EXIT
```

```
FC5F:              ***********************************
FC5F:              * CHANGE MODE = 0(BASE UP)
FC5F:              ***********************************
FC5F:              * CHG.NEW:=ST.BASE(SEG#)+PGCT
FC5F:              *
FC5F:18            CHGS010    CLC
FC60:BD B0 F8                 LDA   ST.BASEL,X
FC63:65 5C                    ADC   CHG.PGCTX
FC65:85 5E                    STA   CHG.NEW
FC67:BD D0 F8                 LDA   ST.BASEH,X
FC6A:65 5D                    ADC   CHG.PGCTX+1
FC6C:85 5F                    STA   CHG.NEW+1
FC6E:              *
FC6E:B0 0C   FC7C             BCS   CHGS014          ; OVERFLOW, PEG IT
FC70:              *
FC70:              * IF CHG.NEW <= ST.LIM(SEG#) THEN EXIT
FC70:              *
FC70:BD F0 F8                 LDA   ST.LIML,X
FC73:C5 5E                    CMP   CHG.NEW
FC75:BD 10 F9                 LDA   ST.LIMH,X
FC78:E5 5F                    SBC   CHG.NEW+1
FC7A:B0 0A   FC86             BCS   CHGS016
FC7C:              *
FC7C:              * OTHERWISE, CHG.NEW:=ST.LIM(SEG#)
FC7C:              *
FC7C:BD F0 F8      CHGS014    LDA   ST.LIML,X
FC7F:85 5E                    STA   CHG.NEW
FC81:BD 10 F9                 LDA   ST.LIMH,X
FC84:85 5F                    STA   CHG.NEW+1
FC86:              *
FC86:4C 48 FD      CHGS016    JMP   CHGS.EXIT
FC89:              ***********************************
FC89:              * CHANGE MODE = 1(BASE DOWN)
FC89:              ***********************************
FC89:              * CHG.NEW:=ST.BASE(SEG#)-PGCT
FC89:              *
FC89:38            CHGS020    SEC
FC8A:BD B0 F8                 LDA   ST.BASEL,X
FC8D:E5 5C                    SBC   CHG.PGCTX
FC8F:85 5E                    STA   CHG.NEW
FC91:BD D0 F8                 LDA   ST.BASEH,X
FC94:E5 5D                    SBC   CHG.PGCTX+1
FC96:85 5F                    STA   CHG.NEW+1
FC98:B0 3F   FCD9             BCS   CHGS050
FC9A:90 4A   FCE6             BCC   CHGS052          ; OVERFLOW, PEG IT
FC9C:              ***********************************
FC9C:              * CHANGE MODE = 2(LIMIT UP)
FC9C:              ***********************************
FC9C:              * CHG.NEW:=ST.LIM(SEG#)+PGCT
FC9C:              *
FC9C:18            CHGS030    CLC
FC9D:BD F0 F8                 LDA   ST.LIML,X
FCA0:65 5C                    ADC   CHG.PGCTX
FCA2:85 5E                    STA   CHG.NEW
FCA4:BD 10 F9                 LDA   ST.LIMH,X
FCA7:65 5D                    ADC   CHG.PGCTX+1
```

```
FCA9:85 5F                    STA   CHG.NEW+1
FCAB:90 2C   FCD9             BCC   CHGS050
FCAD:B0 37   FCE6             BCS   CHGS052          ; OVERFLOW, PEG IT
FCAF:              ***********************************
FCAF:              * CHANGE MODE = 3(LIMIT DOWN)
FCAF:              ***********************************
FCAF:              * CHG.NEW:=ST.LIM(SEG#)-PGCT
FCAF:              *
FCAF:38            CHGS040    SEC
FCB0:BD F0 F8                 LDA   ST.LIML,X
FCB3:E5 5C                    SBC   CHG.PGCTX
FCB5:85 5E                    STA   CHG.NEW
FCB7:BD 10 F9                 LDA   ST.LIMH,X
FCBA:E5 5D                    SBC   CHG.PGCTX+1
FCBC:85 5F                    STA   CHG.NEW+1
FCBE:90 0C   FCCC             BCC   CHGS044          ; OVERFLOW, PEG IT
FCC0:              *
FCC0:              * IF CHG.NEW >= ST.BASE(SEG#) THEN EXIT
FCC0:              *
FCC0:A5 5E                    LDA   CHG.NEW
FCC2:DD B0 F8                 CMP   ST.BASEL,X
FCC5:A5 5F                    LDA   CHG.NEW+1
FCC7:FD D0 F8                 SBC   ST.BASEH,X
FCCA:B0 0A   FCD6             BCS   CHGS046
FCCC:              *
FCCC:              * OTHERWISE CHG.NEW:=ST.BASE(SEG#)
FCCC:              *
FCCC:BD B0 F8      CHGS044    LDA   ST.BASEL,X
FCCF:85 5E                    STA   CHG.NEW
FCD1:BD D0 F8                 LDA   ST.BASEH,X
FCD4:85 5F                    STA   CHG.NEW+1
FCD6:              *
FCD6:4C 48 FD      CHGS046    JMP   CHGS.EXIT
FCD9:              *
FCD9:              * DETERMINE NEW BANK/PAGE'S REGION,
FCD9:              * IF NEW BANK/PAGE IS INVALID THEN
FCD9:              * SET TO BASE OR LIMIT (CASE CHANGE MODE)
FCD9:              *
FCD9:A6 5E         CHGS050    LDX   CHG.NEW
FCDB:A4 5F                    LDY   CHG.NEW+1
FCDD:20 24 FF                 JSR   REGION
FCE0:B0 04   FCE6             BCS   CHGS052
FCE2:D0 02   FCE6             BNE   CHGS052
FCE4:F0 17   FCFD             BEQ   CHGS100
FCE6:A5 62         CHGS052    LDA   CHG.MODE
FCE8:C9 01                    CMP   #1
FCEA:D0 07   FCF3             BNE   CHGS054
FCEC:A2 00                    LDX   #>VRT.BASE
FCEE:A0 00                    LDY   #<VRT.BASE
FCF0:4C F9 FC                 JMP   CHGS056
FCF3:AE 40 00      CHGS054    LDX   >VRT.LIM
FCF6:AC 41 00                 LDY   >VRT.LIM+1
FCF9:86 5E         CHGS056    STX   CHG.NEW
FCFB:84 5F                    STY   CHG.NEW+1
```

```
FCFD:              *
FCFD:              * COMPUTE BANK/PAGE OF ADJACENT SEGMENT, IF ANY
FCFD:              *   CASE CHANGE MODE
FCFD:              *
FCFD:A6 61         CHGS100    LDX   CHG.NUM
FCFF:A5 62                    LDA   CHG.MODE
FD01:C9 01                    CMP   #1
FD03:D0 20   FD25             BNE   CHGS200
FD05:              *   "1" IF ST.FLINK(SEG#)=0 THEN EXIT
FD05:BD 90 F8                 LDA   ST.FLINK,X
FD08:F0 3E   FD48             BEQ   CHGS.EXIT
FD0A:              *       X,Y:=ST.LIM(ST.FLINK(SEG#))+1
FD0A:A8                       TAY
FD0B:B9 F0 F8                 LDA   ST.LIML,Y
FD0E:AA                       TAX
FD0F:B9 10 F9                 LDA   ST.LIMH,Y
FD12:A8                       TAY
FD13:E8                       INX
FD14:D0 01   FD17             BNE   CHGS110
FD16:C8                       INY
FD17:              *       IF CHG.NEW < X,Y THEN CHG.NEW:=X,Y
FD17:C4 5F         CHGS110    CPY   CHG.NEW+1
FD19:90 2D   FD48             BCC   CHGS.EXIT
FD1B:F0 02   FD1F             BEQ   CHGS120
FD1D:B0 25   FD44             BCS   CHGS300
FD1F:E4 5E         CHGS120    CPX   CHG.NEW
FD21:90 25   FD48             BCC   CHGS.EXIT
FD23:B0 1F   FD44             BCS   CHGS300
FD25:              *   "2" IF ST.BLINK(SEG#)=0 THEN EXIT
FD25:BD 70 F8      CHGS200    LDA   ST.BLINK,X
FD28:F0 1E   FD48             BEQ   CHGS.EXIT
FD2A:              *       X,Y:= ST.BASE(ST.BLINK(SEG#))-1
FD2A:A8                       TAY
FD2B:B9 B0 F8                 LDA   ST.BASEL,Y
FD2E:AA                       TAX
FD2F:B9 D0 F8                 LDA   ST.BASEH,Y
FD32:A8                       TAY
FD33:8A                       TXA
FD34:D0 01   FD37             BNE   CHGS210
FD36:88                       DEY
FD37:CA            CHGS210    DEX
FD38:              *       IF CHG.NEW > X,Y THEN CHG.NEW:=X,Y
FD38:C4 5F                    CPY   CHG.NEW+1
FD3A:90 08   FD44             BCC   CHGS300
FD3C:F0 02   FD40             BEQ   CHGS220
FD3E:B0 08   FD48             BCS   CHGS.EXIT
FD40:E4 5E         CHGS220    CPX   CHG.NEW
FD42:B0 04   FD48             BCS   CHGS.EXIT
FD44:              *
FD44:86 5E         CHGS300    STX   CHG.NEW
FD46:84 5F                    STY   CHG.NEW+1
```

```
FD48:              ***********************************
FD48:              *
FD48:              * COMPUTE DELTA PAGE COUNT AND RETURN IT TO CALLER
FD48:              * (CASE OF CHG.MODE)
FD48:              *
FD48:              ***********************************
FD48:A6 61         CHGS.EXIT  LDX   CHG.NUM
FD4A:A0 00                    LDY   #0
FD4C:A5 62                    LDA   CHG.MODE
FD4E:C9 01                    CMP   #1
FD50:90 08   FD5A             BCC   CHGS500
FD52:F0 16   FD6A             BEQ   CHGS510
FD54:C9 03                    CMP   #3
FD56:90 22   FD7A             BCC   CHGS520
FD58:F0 30   FD8A             BEQ   CHGS530
FD5A:              *
FD5A:              * "0" -- PAGECOUNT:=NEW-BASE
FD5A:              *
FD5A:38            CHGS500    SEC
FD5B:A5 5E                    LDA   CHG.NEW
FD5D:FD B0 F8                 SBC   ST.BASEL,X
FD60:91 63                    STA   (CHG.PGCT),Y
FD62:A5 5F                    LDA   CHG.NEW+1
FD64:FD D0 F8                 SBC   ST.BASEH,X
FD67:4C 97 FD                 JMP   CHGS600
FD6A:              *
FD6A:              * "1" -- PAGECOUNT:=BASE-NEW
FD6A:              *
FD6A:38            CHGS510    SEC
FD6B:BD B0 F8                 LDA   ST.BASEL,X
FD6E:E5 5E                    SBC   CHG.NEW
FD70:91 63                    STA   (CHG.PGCT),Y
FD72:BD D0 F8                 LDA   ST.BASEH,X
FD75:E5 5F                    SBC   CHG.NEW+1
FD77:4C 97 FD                 JMP   CHGS600
FD7A:              *
FD7A:              * "2" -- PAGECOUNT:=NEW-LIM
FD7A:              *
FD7A:38            CHGS520    SEC
FD7B:A5 5E                    LDA   CHG.NEW
FD7D:FD F0 F8                 SBC   ST.LIML,X
FD80:91 63                    STA   (CHG.PGCT),Y
FD82:A5 5F                    LDA   CHG.NEW+1
FD84:FD 10 F9                 SBC   ST.LIMH,X
FD87:4C 97 FD                 JMP   CHGS600
FD8A:              *
FD8A:              * "3" -- PAGECOUNT:=LIM-NEW
FD8A:              *
FD8A:38            CHGS530    SEC
FD8B:BD F0 F8                 LDA   ST.LIML,X
FD8E:E5 5E                    SBC   CHG.NEW
FD90:91 63                    STA   (CHG.PGCT),Y
FD92:BD 10 F9                 LDA   ST.LIMH,X
FD95:E5 5F                    SBC   CHG.NEW+1
FD97:              *
FD97:C8            CHGS600    INY
```

```
FD98:91 63                    STA   (CHG.PGCT),Y
FD9A:              *
FD9A:              * IF NEW PAGE COUNT < REQUESTED PAGECOUNT THEN ERR
FD9A:              *
FD9A:AA                       TAX
FD9B:88                       DEY
FD9C:B1 63                    LDA   (CHG.PGCT),Y
FD9E:C5 5C                    CMP   CHG.PGCTX
FDA0:8A                       TXA
FDA1:E5 5D                    SBC   CHG.PGCTX+1
FDA3:B0 05   FDAA             BCS   CHGS610
FDA5:A9 00                    LDA   #SEGRQDN
FDA7:20 00 00                 JSR   SYSERR           ; ERR EXIT
FDAA:              *
FDAA:              * OTHERWISE, ENTER CHG.NEW IN SEGMENT TABLE AND EXIT
FDAA:              *
FDAA:A6 61         CHGS610    LDX   CHG.NUM
FDAC:A5 62                    LDA   CHG.MODE
FDAE:C9 02                    CMP   #2
FDB0:A5 5E                    LDA   CHG.NEW
FDB2:A4 5F                    LDY   CHG.NEW+1
FDB4:B0 09   FDBF             BCS   CHGS620
FDB6:              *
FDB6:9D B0 F8                 STA   ST.BASEL,X
FDB9:98                       TYA
FDBA:9D D0 F8                 STA   ST.BASEH,X
FDBD:18                       CLC
FDBE:60                       RTS                    ; NORMAL EXIT
FDBF:              *
FDBF:              *
FDBF:9D F0 F8      CHGS620    STA   ST.LIML,X
FDC2:98                       TYA
FDC3:9D 10 F9                 STA   ST.LIMH,X
FDC6:18                       CLC
FDC7:60                       RTS                    ; NORMAL EXIT
FDC8:                         CHN   MEMMGR.C.SRC
```

```
FDC8:                  ************************************************************
FDC8:                  *
FDC8:                  * GET.SEG.INFO(IN.SEGNUM; OUT.BASE.BKPG,LIMIT.BKPG,PGCT,SEGID)
FDC8:                  *
FDC8:                  ************************************************************
FDC8:                  *
FDC8:        FDC8      GET.SEG.INFO EQU *
FDC8:                  *
FDC8:                  * IF SEG# OUT OF BOUNDS OR ST.FLINK(SEG#)=ST.FREE THEN ERR
FDC8:                  *
FDC8:A6 61                        LDX   GSI.NUM
FDCA:F0 56   FE22                 BEQ   GSI.ERR          ; ERR - INVALID SEGNUM
FDCC:E0 20                        CPX   #ST.CNT
FDCE:B0 52   FE22                 BCS   GSI.ERR          ; ERR - INVALID SEGNUM
FDD0:BD 90 F8                     LDA   ST.FLINK,X
FDD3:30 4D   FE22                 BMI   GSI.ERR          ; ERR - INVALID SEGNUM
FDD5:                  *
FDD5:                  * RETURN BASE.BKPG TO CALLER
FDD5:                  *
FDD5:BC D0 F8                     LDY   ST.BASEH,X
FDD8:BD B0 F8                     LDA   ST.BASEL,X
FDDB:AA                           TAX
FDDC:20 05 FF                     JSR   CNVRT.XBP
FDDF:98                           TYA
FDE0:A0 01                        LDY   #1
FDE2:91 62                        STA   (GSI.BASE),Y
FDE4:88                           DEY
FDE5:8A                           TXA
FDE6:91 62                        STA   (GSI.BASE),Y
FDE8:                  *
FDE8:                  * RETURN LIMIT.BKPG TO CALLER
FDE8:                  *
FDE8:A6 61                        LDX   GSI.NUM
FDEA:BC 10 F9                     LDY   ST.LIMH,X
FDED:BD F0 F8                     LDA   ST.LIML,X
FDF0:AA                           TAX
FDF1:20 05 FF                     JSR   CNVRT.XBP
FDF4:98                           TYA
FDF5:A0 01                        LDY   #1
FDF7:91 64                        STA   (GSI.LIM),Y
FDF9:88                           DEY
FDFA:8A                           TXA
FDFB:91 64                        STA   (GSI.LIM),Y
FDFD:                  *
FDFD:                  * RETURN SEGID TO CALLER
FDFD:                  *
FDFD:A6 61                        LDX   GSI.NUM
FDFF:BD 30 F9                     LDA   ST.ID,X
FE02:91 68                        STA   (GSI.ID),Y
FE04:                  *
FE04:                  * COMPUTE PAGE COUNT
FE04:                  *
FE04:38                           SEC
FE05:BD F0 F8                     LDA   ST.LIML,X
FE08:FD B0 F8                     SBC   ST.BASEL,X
FE0B:A8                           TAY
```

```
FE0C:BD 10 F9                  LDA   ST.LIMH,X
FE0F:FD D0 F8                  SBC   ST.BASEH,X
FE12:AA                        TAX
FE13:C8                        INY
FE14:D0 01   FE17              BNE   GSI010
FE16:E8                        INX
FE17:               *
FE17:               * RETURN PAGE COUNT TO CALLER
FE17:               *
FE17:98             GSI010     TYA
FE18:A0 00                     LDY   #0
FE1A:91 66                     STA   (GSI.PGCT),Y
FE1C:C8                        INY
FE1D:8A                        TXA
FE1E:91 66                     STA   (GSI.PGCT),Y
FE20:               *
FE20:18                        CLC
FE21:60                        RTS                     ; NORMAL EXIT
FE22:               *
FE22:A9 00          GSI.ERR    LDA   #BADSEGNUM
FE24:20 00 00                  JSR   SYSERR            ; ERR EXIT
```

```
FE27:              ************************************************************
FE27:              *
FE27:              * GET.SEG.NUM(IN.BANKPAGE; OUT.SEGNUM) SYSTEM CALL
FE27:              *
FE27:              *
FE27:              ************************************************************
FE27:              *
FE27:        FE27  GET.SEG.NUM EQU  *
FE27:              *
FE27:              * CONVERT BANKPAGE TO INTERNAL FORMAT
FE27:              *
FE27:A6 61                    LDX   GSN.BKPG
FE29:A4 62                    LDY   GSN.BKPG+1
FE2B:20 C2 FE                 JSR   CNVRT.IBP
FE2E:B0 31   FE61             BCS   GSN.ERR          ; ERR - INVALID BANK PAGE
FE30:86 61                    STX   GSN.BKPG
FE32:84 62                    STY   GSN.BKPG+1
FE34:              *
FE34:              * QUIT IF NO ENTRIES IN SEG TABLE
FE34:              *
FE34:AD 6F F8                 LDA   ST.ENTRY
FE37:F0 29   FE62             BEQ   GSN.ERR1         ; ERR - SEG NOT FOUND
FE39:              *
FE39:              * L1: IF BANKPAGE>ST.LIM(SEG#) THEN ERR
FE39:              *
FE39:AA            GSN010     TAX
FE3A:BD F0 F8                 LDA   ST.LIML,X
FE3D:C5 61                    CMP   GSN.BKPG
FE3F:BD 10 F9                 LDA   ST.LIMH,X
FE42:E5 62                    SBC   GSN.BKPG+1
FE44:90 1C   FE62             BCC   GSN.ERR1         ; ERR - SEG NOT FOUND
FE46:              *
FE46:              * IF BANKPAGE>=ST.BASE(SEG#) THEN FOUND!
FE46:              *
FE46:A5 61                    LDA   GSN.BKPG
FE48:DD B0 F8                 CMP   ST.BASEL,X
FE4B:A5 62                    LDA   GSN.BKPG+1
FE4D:FD D0 F8                 SBC   ST.BASEH,X
FE50:B0 08   FE5A             BCS   GSN020
FE52:              *
FE52:              * SEG#:=ST.FLINK(SEG#); GOTO L1
FE52:              *
FE52:BD 90 F8                 LDA   ST.FLINK,X
FE55:F0 0B   FE62             BEQ   GSN.ERR1         ; ERR - SEG NOT FOUND
FE57:4C 39 FE                 JMP   GSN010
FE5A:              *
FE5A:              * RETURN SEG# TO CALLER
FE5A:              *
FE5A:A0 00         GSN020     LDY   #0
FE5C:8A                       TXA
FE5D:91 63                    STA   (GSN.NUM),Y
FE5F:18                       CLC
FE60:60                       RTS                    ; NORMAL EXIT
FE61:              *
FE61:60            GSN.ERR    RTS                    ; ERROR EXIT
FE62:              *
```

```
FE62:A9 00             GSN.ERR1   LDA   #SEGNOTFND
FE64:20 00 00                     JSR   SYSERR           ; ERROR EXIT
```

```
FE67:              ************************************************************
FE67:              *
FE67:              * RELEASE.SEG(IN.SEGNUM) SYSTEM CALL
FE67:              *
FE67:              ************************************************************
FE67:              *
FE67:        FE67  RELEASE.SEG EQU  *
FE67:              *
FE67:              * IF ST.FLINK(SEG#)=ST.FREE THEN ERR
FE67:              *
FE67:A6 61                    LDX   RLS.NUM
FE69:F0 0B   FE76             BEQ   RLS.ALL          ; RELEASE.SEG(SEG#=0)
FE6B:E0 20                    CPX   #ST.CNT
FE6D:B0 4E   FEBD             BCS   RLS.ERR          ; ERR - SEG# TOO LARGE
FE6F:BD 90 F8                 LDA   ST.FLINK,X
FE72:30 49   FEBD             BMI   RLS.ERR          ; ERR - INVALID SEGNUM
FE74:10 1F   FE95             BPL   REL.SEG          ; RELEASE.SEG(SEG#>0)
FE76:              ***********************************
FE76:              *
FE76:              * RELEASE ALL
FE76:              *
FE76:              ***********************************
FE76:AE 6F F8      RLS.ALL    LDX   ST.ENTRY
FE79:F0 18   FE93             BEQ   RLS0.EXIT
FE7B:86 61                    STX   RLS.NUM
FE7D:              *
FE7D:BD 30 F9      RLS0.LOOP  LDA   ST.ID,X
FE80:C9 10                    CMP   #$10             ; CARRY SET/CLEARED HERE
FE82:              *
FE82:BD 90 F8                 LDA   ST.FLINK,X
FE85:48                       PHA
FE86:90 03   FE8B             BCC   RLS006           ; IF ID=SYS SEG THEN SKIP
FE88:20 95 FE                 JSR   REL.SEG          ; RELEASE ONE SEGMENT
FE8B:68            RLS006     PLA
FE8C:F0 05   FE93             BEQ   RLS0.EXIT
FE8E:85 61                    STA   RLS.NUM
FE90:AA                       TAX
FE91:D0 EA   FE7D             BNE   RLS0.LOOP        ; ALWAYS TAKEN
FE93:              *
FE93:18            RLS0.EXIT  CLC
FE94:60                       RTS                    ; NORMAL EXIT ; ALL NON SYSTEM SEGMENTS RELEASED.
FE95:              ***********************************
FE95:              *
FE95:              * REL SEG
FE95:              *
FE95:              ***********************************
FE95:              * Y:=ST.FLINK(SEG#)
FE95:              * X:=ST.BLINK(SEG#)
FE95:              *
FE95:A8            REL.SEG    TAY
FE96:BD 70 F8                 LDA   ST.BLINK,X
FE99:AA                       TAX
FE9A:              *
FE9A:              * IF X<>0 THEN ST.FLINK(X):=Y
FE9A:              *         ELSE ST.ENTRY:=Y
FE9A:              *
```

```
FE9A:F0 07   FEA3             BEQ   RLS010
FE9C:98                       TYA
FE9D:9D 90 F8                 STA   ST.FLINK,X
FEA0:4C A7 FE                 JMP   RLS020
FEA3:8C 6F F8      RLS010     STY   ST.ENTRY
FEA6:              *
FEA6:              * IF Y<>0 THEN ST.BLINK(Y):=X
FEA6:              *
FEA6:98                       TYA
FEA7:F0 04   FEAD  RLS020     BEQ   RLS030
FEA9:8A                       TXA
FEAA:99 70 F8                 STA   ST.BLINK,Y
FEAD:              *
FEAD:              * ST.FLINK(SEG#):=ST.FREE
FEAD:              * ST.FREE:=SEG# AND #$80
FEAD:              *
FEAD:AD 6E F8      RLS030     LDA   ST.FREE
FEB0:A6 61                    LDX   RLS.NUM
FEB2:9D 90 F8                 STA   ST.FLINK,X
FEB5:8A                       TXA
FEB6:09 80                    ORA   #$80
FEB8:8D 6E F8                 STA   ST.FREE
FEBB:              *
FEBB:18                       CLC
FEBC:60                       RTS                    ; NORMAL EXIT
FEBD:              *
FEBD:A9 00         RLS.ERR    LDA   #BADSEGNUM
FEBF:20 00 00                 JSR   SYSERR           ; ERR EXIT
```

```
FEC2:              *************************************************************
FEC2:              *
FEC2:              * CONVERT INTERNAL BANK PAGE
FEC2:              *
FEC2:              * INPUT:  EXTERNAL BANK (X)
FEC2:              *            "     PAGE (Y)
FEC2:              * OUTPUT: INTERNAL BKPG LOW  (X)
FEC2:              *            "     BKPG HIGH (Y)
FEC2:              *         REGION (A) 0=>VIRT BANK
FEC2:              *                    1=>PHY BANK (0-$2000)
FEC2:              *                    2=>   "     ($A000-$FFFF)
FEC2:              * ERROR:  CARRY SET ("INVALID BANK PAGE")
FEC2:              *
FEC2:              *************************************************************
FEC2:              *
FEC2:        FEC2  CNVRT.IBP  EQU   *
FEC2:              *
FEC2:              * CONVERT FROM EXTERNAL TO INTERNAL FORMAT
FEC2:              *
FEC2:              * CASE OF BANK:  ADD PAGE BIAS
FEC2:              *
FEC2:98                       TYA
FEC3:E0 0F                    CPX   #$F
FEC5:F0 10   FED7             BEQ   CNVI010
FEC7:B0 18   FEE1             BCS   CNVI020
FEC9:              *
FEC9:C9 20                    CMP   #$20             ; BANK < "F"
FECB:90 33   FF00             BCC   CNVI.ERR1
FECD:C9 A0                    CMP   #$A0
FECF:B0 2F   FF00             BCS   CNVI.ERR1
FED1:38                       SEC
FED2:E9 20                    SBC   #$20
FED4:4C EC FE                 JMP   CNVI030
FED7:              *
FED7:C9 20         CNVI010    CMP   #$20             ; BANK = "F"
FED9:B0 25   FF00             BCS   CNVI.ERR1
FEDB:18                       CLC
FEDC:69 80                    ADC   #$80
FEDE:4C EC FE                 JMP   CNVI030
FEE1:              *
FEE1:E0 10         CNVI020    CPX   #$10             ; BANK = "10"
FEE3:D0 1B   FF00             BNE   CNVI.ERR1
FEE5:C9 A0                    CMP   #$A0
FEE7:90 17   FF00             BCC   CNVI.ERR1
FEE9:38                       SEC
FEEA:E9 80                    SBC   #$80
FEEC:              *
FEEC:A8            CNVI030    TAY                    ; SHIFT BANK RIGHT ONE BIT
FEED:8A                       TXA                    ; INTO HIGH BIT OF PAGE BYTE.
FEEE:4A                       LSR   A
FEEF:AA                       TAX
FEF0:98                       TYA
FEF1:90 02   FEF5             BCC   CNVI040
FEF3:09 80                    ORA   #$80
FEF5:              *
FEF5:              * EXCHANGE X & Y
```

```
FEF5:              *
FEF5:48            CNVI040    PHA
FEF6:8A                       TXA
FEF7:A8                       TAY
FEF8:68                       PLA
FEF9:AA                       TAX
FEFA:              *
FEFA:              * COMPUTE REGION (VIRT=0,PHY1=1,PHY2=2)
FEFA:              *
FEFA:20 24 FF                 JSR   REGION           ; REGION RETURNED IN A REG.
FEFD:B0 01   FF00             BCS   CNVI.ERR1        ; ERR - INVALID BANK PAGE
FEFF:              *
FEFF:60                       RTS                    ; NORMAL EXIT
FF00:              *
FF00:A9 00         CNVI.ERR1  LDA   #BADBKPG
FF02:20 00 00                 JSR   SYSERR
```

```
FF05:              ************************************************************
FF05:              *
FF05:              * CONVERT EXTERNAL BANK PAGE
FF05:              *
FF05:              * INPUT:  INTERNAL BKPG LOW  (X)
FF05:              *             "         HIGH (Y)
FF05:              * OUTPUT: EXTERNAL BANK (X)
FF05:              *             "    PAGE (Y)
FF05:              * ERROR:  NO ERROR CHECKING DONE.  ASSUMES THAT INTERNAL #S
FF05:              * ARE VALID.
FF05:              *
FF05:              ************************************************************
FF05:              *
FF05:        FF05  CNVRT.XBP  EQU   *
FF05:              *
FF05:              * CONVERT FROM INTERNAL TO EXTERNAL FORMAT
FF05:              *
FF05:8A                       TXA
FF06:0A                       ASL   A
FF07:8A                       TXA
FF08:29 7F                    AND   #$7F
FF0A:AA                       TAX
FF0B:98                       TYA
FF0C:2A                       ROL   A
FF0D:A8                       TAY
FF0E:              *
FF0E:              * CASE OF BANK: ADD PAGE BIAS
FF0E:              *
FF0E:8A                       TXA
FF0F:C0 0F                    CPY   #$F
FF11:F0 0B   FF1E             BEQ   CNVX020          ; BANK = "F"
FF13:B0 06   FF1B             BCS   CNVX010
FF15:              *
FF15:18                       CLC                    ; BANK < "F"
FF16:69 20                    ADC   #$20
FF18:4C 1E FF                 JMP   CNVX020
FF1B:              *
FF1B:18            CNVX010    CLC                    ; BANK = "10"
FF1C:69 80                    ADC   #$80
FF1E:              *
FF1E:              * EXCHANGE X & Y
FF1E:              *
FF1E:48            CNVX020    PHA
FF1F:98                       TYA
FF20:AA                       TAX
FF21:68                       PLA
FF22:A8                       TAY
FF23:60                       RTS                    ; NORMAL EXIT
```

```
FF24:              ************************************************************
FF24:              *
FF24:              * REGION
FF24:              *
FF24:              * INPUT:  INTERNAL BKPG LOW  (X)
FF24:              *             "         HIGH (Y)
FF24:              * OUTPUT: REGION (A)
FF24:              *         INTERNAL BKPG LOW  (X) UNCHANGED
FF24:              *             "         HIGH (Y)    "
FF24:              * ERROR:  CARRY SET ("INVALID BANK/PAGE")
FF24:              *
FF24:              ************************************************************
FF24:              *
FF24:        FF24  REGION     EQU   *
FF24:8E 50 F9                 STX   RGN.BKPG
FF27:8C 51 F9                 STY   RGN.BKPG+1
FF2A:              *
FF2A:              * IF BANKPAGE>PHY2LIM THEN ERR
FF2A:              *
FF2A:A9 7F                    LDA   #>PHY2LIM
FF2C:CD 50 F9                 CMP   RGN.BKPG
FF2F:A9 08                    LDA   #<PHY2LIM
FF31:ED 51 F9                 SBC   RGN.BKPG+1
FF34:90 3E   FF74             BCC   RGN.ERR          ; ERR - INVALID BANK PAGE
FF36:              *
FF36:              * IF BANKPAGE>=PHY2BASE THEN REGION:=2
FF36:              *
FF36:AD 50 F9                 LDA   RGN.BKPG
FF39:C9 20                    CMP   #>PHY2BASE
FF3B:AD 51 F9                 LDA   RGN.BKPG+1
FF3E:E9 08                    SBC   #<PHY2BASE
FF40:90 04   FF46             BCC   RGN010
FF42:A9 02                    LDA   #2
FF44:D0 2C   FF72             BNE   RGN040
FF46:              *
FF46:              * IF BANKPAGE>PHY1LIMIT THEN ERR
FF46:              *
FF46:A9 9F         RGN010     LDA   #>PHY1LIM
FF48:CD 50 F9                 CMP   RGN.BKPG
FF4B:A9 07                    LDA   #<PHY1LIM
FF4D:ED 51 F9                 SBC   RGN.BKPG+1
FF50:90 22   FF74             BCC   RGN.ERR          ; ERR - INVALID BANK PAGE
FF52:              *
FF52:              * IF BANKPAGE>=PHY1BASE THEN REGION:=1
FF52:              *
FF52:AD 50 F9                 LDA   RGN.BKPG
FF55:C9 80                    CMP   #>PHY1BASE
FF57:AD 51 F9                 LDA   RGN.BKPG+1
FF5A:E9 07                    SBC   #<PHY1BASE
FF5C:90 04   FF62             BCC   RGN020
FF5E:A9 01                    LDA   #1
FF60:D0 10   FF72             BNE   RGN040
FF62:              *
FF62:              * IF BANKPAGE>VIRTUAL LIMIT THEN ERR
FF62:              *
FF62:AD 40 00      RGN020     LDA   >VRT.LIM
```

```
FF65:CD 50 F9                  CMP   RGN.BKPG
FF68:AD 41 00                  LDA   >VRT.LIM+1
FF6B:ED 51 F9                  SBC   RGN.BKPG+1
FF6E:90 04   FF74              BCC   RGN.ERR
FF70:A9 00                     LDA   #0
FF72:              *
FF72:18            RGN040      CLC                      ; "N" FLAG ALWAYS REFLECTS REGION VAL IN A REG!
FF73:60                        RTS                      ; NORMAL EXIT
FF74:              *
FF74:38            RGN.ERR     SEC                      ; INVALID BANK PAGE
FF75:60                        RTS
```

```
FF76:              ************************************************************
FF76:              *
FF76:              * GET FREE
FF76:              *
FF76:              * INPUT:  PREVIOUS SEG # (A)
FF76:              * OUTPUT: NEW SEG #      (A)
FF76:              * ERROR:  CARRY SET ("SEG TBL FULL")
FF76:              *
FF76:              ************************************************************
FF76:              *
FF76:        FF76  GET.FREE   EQU   *
FF76:              *
FF76:              * SAVE PREV SEG # IN X
FF76:              * NOTE: PREV SEG # CARRIED IN X
FF76:              *       NEW SEG # CARRIED IN Y
FF76:              *
FF76:AA                       TAX
FF77:              *
FF77:              * IF NO FREE ENTRIES THEN ERR
FF77:              *
FF77:AD 6E F8                 LDA   ST.FREE
FF7A:C9 80                    CMP   #$80
FF7C:F0 3C   FFBA             BEQ   GTFR.ERR
FF7E:              *
FF7E:              * TURN OFF FREE FLAG (BIT7) AND DELINK FROM FREE LIST
FF7E:              *
FF7E:29 7F                    AND   #$7F
FF80:A8                       TAY
FF81:B9 90 F8                 LDA   ST.FLINK,Y
FF84:8D 6E F8                 STA   ST.FREE
FF87:              *
FF87:              * IF PREV SEG # IS NULL THEN LINK NEW ENTRY TO START
FF87:              * OF SEGMENT LIST
FF87:              *
FF87:E0 00                    CPX   #0
FF89:D0 11   FF9C             BNE   GTFR010
FF8B:AD 6F F8                 LDA   ST.ENTRY
FF8E:99 90 F8                 STA   ST.FLINK,Y
FF91:A9 00                    LDA   #0
FF93:99 70 F8                 STA   ST.BLINK,Y
FF96:8C 6F F8                 STY   ST.ENTRY
FF99:4C AA FF                 JMP   GTFR020
FF9C:              *
FF9C:              * OTHERWISE LINK NEW ENTRY TO PREV SEG #
FF9C:              *
FF9C:BD 90 F8      GTFR010    LDA   ST.FLINK,X
FF9F:99 90 F8                 STA   ST.FLINK,Y
FFA2:8A                       TXA
FFA3:99 70 F8                 STA   ST.BLINK,Y
FFA6:98                       TYA
FFA7:9D 90 F8                 STA   ST.FLINK,X
FFAA:              *
FFAA:              * IF ST.FLINK(NEW)<>NULL THEN
FFAA:              *    ST.BLINK(ST.FLINK(NEW)):=NEWSEG #
FFAA:B9 90 F8      GTFR020    LDA   ST.FLINK,Y
FFAD:F0 08   FFB7             BEQ   GTFR030
```

```
FFAF:B9 90 F8                 LDA   ST.FLINK,Y
FFB2:AA                       TAX
FFB3:98                       TYA
FFB4:9D 70 F8                 STA   ST.BLINK,X
FFB7:                  *
FFB7:                  * RETURN WITH NEW SEG #
FFB7:                  *
FFB7:98                GTFR030    TYA
FFB8:18                       CLC
FFB9:60                       RTS                    ; NORMAL EXIT
FFBA:                  *
FFBA:A9 00             GTFR.ERR   LDA   #SEGTBLFULL
FFBC:20 00 00                 JSR   SYSERR
FFBF:                  *

FFBF:                         LST   ON
FFBF:        FFBF      ZZEND      EQU   *
FFBF:        0751      ZZLEN      EQU   ZZEND-ZZORG
FFBF:        0000                 IFNE  ZZLEN-LENMEMMG
 S                            FAIL  2,"SOSORG        FILE IS INCORRECT FOR MEMMGR"
FFBF:                         FIN
```

```
X0009 BADBKPG       X000F BADCHGMODE    X0010 BADPGCNT      X0008 BADSCNUM
X000C BADSEGNUM     X000E BADSRCHMODE      57 BFS.BASE         5B BFS.BLINK
   59 BFS.LIM          55 BFS.PGCT       3200 BLABFM        ?2E00 BLABFMI
 6B52 BLABUFMG       6955 BLACFM         5E99 BLADISK3       64D9 BLADMGR
 68F4 BLAFMGR       ?2CF8 BLAGLOB       ?2AF8 BLAINIT        55C0 BLAIPL
 2000 BLALODR       ?6E6E BLAMEMMG       5466 BLAOMSG        5466 BLAPATCH
 665E BLASCMGR       6404 BLASERR        5A8B BLAUMGR          4D CFS.BASE0
   4F CFS.BASE1        48 CFS.BASE         4C CFS.BLINK        4A CFS.LIM
   51 CFS.NEXT         46 CFS.PGCT         52 CFS.PREV         53 CFS.PTR
   62 CHG.MODE         5E CHG.NEW          61 CHG.NUM          5C CHG.PGCTX
   63 CHG.PGCT       FC2F CHG.SEG        FC47 CHGS.ERR       FD48 CHGS.EXIT
 FC4C CHGS005        FC5F CHGS010        FC7C CHGS014        FC86 CHGS016
 FC89 CHGS020        FC9C CHGS030        FCAF CHGS040        FCCC CHGS044
 FCD6 CHGS046        FCD9 CHGS050        FCE6 CHGS052        FCF3 CHGS054
 FCF9 CHGS056        FCFD CHGS100        FD17 CHGS110        FD1F CHGS120
 FD25 CHGS200        FD37 CHGS210        FD40 CHGS220        FD44 CHGS300
 FD5A CHGS500        FD6A CHGS510        FD7A CHGS520        FD8A CHGS530
 FD97 CHGS600        FDAA CHGS610        FDBF CHGS620        FF00 CNVI.ERR1
 FED7 CNVI010        FEE1 CNVI020        FEEC CNVI030        FEF5 CNVI040
 FEC2 CNVRT.IBP      FF05 CNVRT.XBP      FF1B CNVX010        FF1E CNVX020
   65 F.BASE           45 F.ERR            62 F.ID             67 F.LIM
   69 F.NUM            63 F.PGCT           00 FALSE          FAEC FIND.ERR
 FA23 FIND.SEG       FA39 FIND001        FA48 FIND005        FA51 FIND010
 FA5F FIND015        FA6B FIND020        FA72 FIND030        FA8E FIND050
 FA9E FIND060        FAB8 FIND070        FB37 FRSG010        FB4F FRSG020
 FB60 FRSG030        FB7C FRSG035        FB95 FRSG040        FBAD FRSG050
 FBC0 FRSG052        FBD7 FRSG055        FBDF FRSG060        FC03 FRSG070
 FB30 FRSGI.EXIT     FB04 FRSGI010       FB1F FRSGI020         43 FX.PGCT
 FF76 GET.FREE       FDC8 GET.SEG.INFO   FE27 GET.SEG.NUM      62 GSI.BASE
 FE22 GSI.ERR          68 GSI.ID           64 GSI.LIM          61 GSI.NUM
   66 GSI.PGCT       FE17 GSI010           61 GSN.BKPG       FE61 GSN.ERR
 FE62 GSN.ERR1         63 GSN.NUM        FE39 GSN010         FE5A GSN020
 FFBA GTFR.ERR       FF9C GTFR010        FFAA GTFR020        FFB7 GTFR030
?0400 LENBFMI        2266 LENBFM         031C LENBUFMG       01FD LENCFM
 056B LENDISK3       0185 LENDMGR          61 LENFMGR       ?01B2 LENINIT
 04CB LENIPL         0AF8 LENLODR        0751 LENMEMMG       015A LENOMSG
   00 LENPATCH       0296 LENSCMGR         D5 LENSERR        040E LENUMGR
   60 M.RQCODE         60 M.TPARMX       F96F MMGR010        F972 MMGR020
 F975 MMGR030        F97E MMGR060       NF952 MMGR           F978 MMGR040
 F97B MMGR050        FC2E NFRPG.EXIT     FC05 NXTFRPG        FAF1 NXTFRSEG.I
 FB31 NXTFRSEG       BC00 ORGBFM         B800 ORGBFMI        F552 ORGBUFMG
 F355 ORGCFM         E899 ORGDISK3       EED9 ORGDMGR        FFBF ORGEND
 F2F4 ORGFMGR       ?18FC ORGGLOB        28F8 ORGINIT        DFC0 ORGIPL
 1E00 ORGLODR        F86E ORGMEMMG       DE66 ORGOMSG        DE66 ORGPATCH
 F05E ORGSCMGR       EE04 ORGSERR        E48B ORGUMGR        0780 PHY1BASE
 079F PHY1LIM        0820 PHY2BASE       087F PHY2LIM        FF24 REGION
 FE95 REL.SEG        FE67 RELEASE.SEG    F981 REQ.SEG        F950 RGN.BKPG
 FF74 RGN.ERR        FF46 RGN010         FF62 RGN020         FF72 RGN040
 FE76 RLS.ALL        FEBD RLS.ERR          61 RLS.NUM        FE93 RLS0.EXIT
 FE7D RLS0.LOOP      FE8B RLS006         FEA3 RLS010         FEA7 RLS020
 FEAD RLS030           61 RQ.BASE        FA14 RQ.ERR1        F98A RQ.ERR
 FA19 RQ.ERR2        FA1E RQ.ERR3          65 RQ.ID            63 RQ.LIM
   66 RQ.NUM           42 RQ.REGION      F98B RQ005          F9BF RQ010
 F9D5 RQ020          F9ED RQ030         X000D SEGNOTFND     X000A SEGRQDN
X000B SEGTBLFULL       61 SRCHMODE       F8D0 ST.BASEH       F8B0 ST.BASEL
 F870 ST.BLINK      N0020 ST.CNT        NF86F ST.ENTRY      NF890 ST.FLINK
```

```
NF86E ST.FREE        F930 ST.ID          F910 ST.LIMH        F8F0 ST.LIML
   07 ST.SIZ         F870 ST.TBL        X0007 SYSERR           80 TRUE
   00 VRT.BASE      N0040 VRT.LIM          40 ZPAGE          FFBF ZZEND
 0751 ZZLEN          F86E ZZORG
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED  1616
** FREE SPACE PAGE COUNT   75
```

```
SOURCE   FILE #01 =>PRINT
 INCLUDE FILE #02 =>SOSORG
SOURCE   FILE #03 =>EQUATES
SOURCE   FILE #04 =>PATH
SOURCE   FILE #05 =>VOLUME
SOURCE   FILE #06 =>CREATE
SOURCE   FILE #07 =>FNDFIL
SOURCE   FILE #08 =>ALLOC
SOURCE   FILE #09 =>POSN.OPEN
SOURCE   FILE #10 =>READ.WRITE
SOURCE   FILE #11 =>CLOSE.EOF
SOURCE   FILE #12 =>DESTROY
SOURCE   FILE #13 =>SWAPOUT.IN
```

```
0000:                  * 01-FEB-82
0000:                             REL
0000:        0000                 IBUFSIZ 1
0000:        0000                 SBUFSIZ 40
0000:                             INCLUDE SOSORG
0000:
****************************************************************************************************
0000:                  *  SOS KERNEL MODULE ORIGINS
0000:        1E00      ORGLODR    EQU   $1E00            ; ORIGIN OF SOS LOADER
0000:        28F8      ORGINIT    EQU   $28F8            ; ORIGIN OF INIT
0000:        18FC      ORGGLOB    EQU   $18FC            ; ORIGIN OF SYSGLOB
0000:        B800      ORGBFMI    EQU   $B800            ; ORIGIN OF BFM.INIT2 & BITMAPS
0000:        BC00      ORGBFM     EQU   $BC00            ; ORIGIN OF BFM
0000:        DE66      ORGPATCH   EQU   $DE66            ; ORIGIN OF PATCH AREA
0000:        DE66      ORGOMSG    EQU   $DE66            ; ORIGIN OF OPRMSG
0000:        DFC0      ORGIPL     EQU   $DFC0            ; ORIGIN OF IPL
0000:        E48B      ORGUMGR    EQU   $E48B            ; ORIGIN OF UMGR
0000:        E899      ORGDISK3   EQU   $E899            ; ORIGIN OF DISK3
0000:        EE04      ORGSERR    EQU   $EE04            ; ORIGIN OF SYSERR
0000:        EED9      ORGDMGR    EQU   $EED9            ; ORIGIN OF DEVMGR
0000:        F05E      ORGSCMGR   EQU   $F05E            ; ORIGIN OF SCMGR
0000:        F2F4      ORGFMGR    EQU   $F2F4            ; ORIGIN OF FMGR
0000:        F355      ORGCFM     EQU   $F355            ; ORIGIN OF CFMGR
0000:        F552      ORGBUFMG   EQU   $F552            ; ORIGIN OF BUFMGR
0000:        F86E      ORGMEMMG   EQU   $F86E            ; ORIGIN OF MEMMGR
0000:        FFBF      ORGEND     EQU   $FFBF            ; END MARKER
0000:
****************************************************************************************************
0000:                  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
0000:        0AF8      LENLODR    EQU   ORGINIT-ORGLODR  ; LENGTH OF SOS LOADER
0000:        01B2      LENINIT    EQU   $01B2            ; LENGTH OF INIT
0000:        0400      LENBFMI    EQU   ORGBFM-ORGBFMI   ; LENGTH OF BFM.INIT2 & BITMAPS
0000:        2266      LENBFM     EQU   ORGPATCH-ORGBFM  ; LENGTH OF BFM
0000:        0000      LENPATCH   EQU   ORGOMSG-ORGPATCH ; LENGTH OF PATCH AREA
0000:        015A      LENOMSG    EQU   ORGIPL-ORGOMSG   ; LENGTH OF OPRMSG
0000:        04CB      LENIPL     EQU   ORGUMGR-ORGIPL   ; LENGTH OF IPL
0000:        040E      LENUMGR    EQU   ORGDISK3-ORGUMGR ; LENGTH OF UMGR
0000:        056B      LENDISK3   EQU   ORGSERR-ORGDISK3 ; LENGTH OF DISK3
0000:        00D5      LENSERR    EQU   ORGDMGR-ORGSERR  ; LENGTH OF SYSERR
0000:        0185      LENDMGR    EQU   ORGSCMGR-ORGDMGR ; LENGTH OF DEVMGR
0000:        0296      LENSCMGR   EQU   ORGFMGR-ORGSCMGR ; LENGTH OF SCMGR
0000:        0061      LENFMGR    EQU   ORGCFM-ORGFMGR   ; LENGTH OF FMGR
0000:        01FD      LENCFM     EQU   ORGBUFMG-ORGCFM  ; ORIGIN OF CFMGR
0000:        031C      LENBUFMG   EQU   ORGMEMMG-ORGBUFMG ; LENGTH OF BUFMGR
0000:        0751      LENMEMMG   EQU   ORGEND-ORGMEMMG  ; LENGTH OF MEMMGR
0000:
****************************************************************************************************
0000:                  *      SOS BLOAD ADDRESSES
0000:        2000      BLALODR    EQU   $2000            ; BLOAD ADDRESS OF SOS LOADER
0000:        2AF8      BLAINIT    EQU   BLALODR+LENLODR  ; BLOAD ADDRESS OF INIT
0000:        2CF8      BLAGLOB    EQU   $2CF8            ; BLOAD ADDRESS OF SYSGLOB
0000:        2E00      BLABFMI    EQU   $2E00            ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
0000:        3200      BLABFM     EQU   $3200            ; BLOAD ADDRESS OF BFM
0000:        5466      BLAPATCH   EQU   BLABFM+LENBFM    ; BLOAD ADDRESS OF PATCH AREA
0000:        5466      BLAOMSG    EQU   BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
0000:        55C0      BLAIPL     EQU   BLAOMSG+LENOMSG  ; BLOAD ADDRESS OF IPL
0000:        5A8B      BLAUMGR    EQU   BLAIPL+LENIPL    ; BLOAD ADDRESS OF UMGR
0000:        5E99      BLADISK3   EQU   BLAUMGR+LENUMGR  ; BLOAD ADDRESS OF DISK3
0000:        6404      BLASERR    EQU   BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
```

```
0000:        64D9      BLADMGR    EQU   BLASERR+LENSERR  ; BLOAD ADDRESS OF DEVMGR
0000:        665E      BLASCMGR   EQU   BLADMGR+LENDMGR  ; BLOAD ADDRESS OF SCMGR
0000:        68F4      BLAFMGR    EQU   BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
0000:        6955      BLACFM     EQU   BLAFMGR+LENFMGR  ; BLOAD ADDRESS OF CFMGR
0000:        6B52      BLABUFMG   EQU   BLACFM+LENCFM    ; BLOAD ADDRESS OF BUFMGR
0000:        6E6E      BLAMEMMG   EQU   BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
0000:
****************************************************************************************************
BC00:        BC00                 ORG   ORGBFM           ; BITMAPS $B800-$BBFF
BC00:        BC00      ZZORG      EQU   *
BC00:                  ************************************************************
BC00:                  *          (C) COPYRIGHT 1981 BY APPLE COMPUTER INC.
BC00:                  *                     ALL RIGHTS RESERVED
BC00:                  ************************************************************
BC00:                             MSB   OFF
BC00:                             LST   VSYM
BC00:                             CHN   EQUATES
BC00:                  *
BC00:        BC00                 ENTRY BFMGR
BC00:                  *
BC00:                  * BFM INITIALIZATION ENTRIES
BC00:                  * (INIT CODE FOUND IN INIT.SRC)
BC00:                  *
BC00:        001C                 ENTRY BFMFCB1          ; FCB PAGE 1 ADDR
BC00:        001D                 ENTRY BFMFCB2          ; AND PAGE 2
BC00:        00BA                 ENTRY FCBZPP
BC00:        1400                 ENTRY SISTER
BC00:        1000                 ENTRY PATHBUF
BC00:        1100                 ENTRY VCB
BC00:        DB9F                 ENTRY WORKSPC
BC00:        0015                 ENTRY PFIXPTR
BC00:        00B8                 ENTRY BMAPAGE
BC00:        00BA                 ENTRY BMBPAGE
BC00:        0028                 ENTRY FCBADDRH
BC00:        001E                 ENTRY BMAMADR
BC00:        0024                 ENTRY BMBMADR
BC00:                  *
BC00:                  *
BC00:        0000                 EXTRN LEVEL            ; FILE LEVEL (LOW BYTE)
BC00:        0000                 EXTRN OPMSGRPLY         ; OPERATOR MESSAGE
BC00:        0000                 EXTRN DATETIME         ; THANKS TOM...
BC00:        0000                 EXTRN DMGR             ; THANKS BOB...
BC00:        0000                 EXTRN REQBUF           ;   "
BC00:        0000                 EXTRN REQFXBUF         ;   "
BC00:        0000                 EXTRN GETBUFADR        ;  "
BC00:        0000                 EXTRN RELBUF           ;   "
BC00:        0000                 EXTRN BLKDLST          ;   "
BC00:        0000                 EXTRN SERR
BC00:        0000                 EXTRN BACKMASK
BC00:                  *
BC00:                  * ERRORS
BC00:                  *
BC00:        0000                 EXTRN SYSERR
BC00:                  *
BC00:        0000                 EXTRN BADPATH          ; INVALID PATHNAME SYNTAX
BC00:        0000                 EXTRN FCBFULL          ; FILE CONTROL BLOCK FULL
BC00:        0000                 EXTRN BADREFNUM        ; INVALID REFNUM
```

```
BC00:        0000              EXTRN PATHNOTFND       ; PATHNAME NOT FOUND
BC00:        0000              EXTRN VNFERR           ; VOLUME NOT FOUND
BC00:        0000              EXTRN FNFERR           ; FILE NOT FOUND
BC00:        0000              EXTRN DUPERR           ; DUPLICATE FILE NAME ERROR
BC00:        0000              EXTRN DUPVOL           ; DUPLICATE VOLUME CAN'T BE LOGGED IN.
BC00:        0000              EXTRN OVRERR           ; NOT ENOUGH DISK SPACE FOR PREALLOCATION
BC00:        0000              EXTRN DIRFULL          ; DIRECTORY FULL ERROR
BC00:        0000              EXTRN CPTERR           ; FILE INCOMPATIBLE SOS VERSION
BC00:        0000              EXTRN TYPERR           ; NOT CURRENTLY SUPPORTED FILE TYPE
BC00:        0000              EXTRN EOFERR           ; POSITION ATTEMPTED BEYOND END OF FILE
BC00:        0000              EXTRN POSNERR          ; ILLEGAL POSITION (L.T. 0 OR G.T. $FFFFFF)
BC00:        0000              EXTRN ACCSERR          ; FILE ACCESS R/W REQUEST CONFLICTS WITH ATTRIBUTES.
BC00:        0000              EXTRN BTSERR           ; USER SUPPLIED BUFFER TOO SMALL
BC00:        0000              EXTRN FILBUSY          ; EITHER WRITE WAS REQUESTED OR WRITE ACCESS ALREADY ALLOCATED.
BC00:        0000              EXTRN NOTSOS           ; NOT A SOS DISKETTE
BC00:        0000              EXTRN BADLSTCNT        ; INVALID VALUE IN LIST PARAMETER
BC00:        0000              EXTRN XDISKSW          ; DISK SWITCHED
BC00:        0000              EXTRN NOTBLKDEV        ; NOT A BLOCK DEVICE
BC00:        0000              EXTRN XNOWRITE         ; DISK/MEDIA IS HARDWARE WRITE PROTECTED
BC00:        0000              EXTRN XIOERROR         ; INFORMATION ON BLOCK DEVICE NOT ACCESSABLE
BC00:        0000              EXTRN DIRERR           ; DIRECTORY ENTRY COUNT INCONSISTENT WITH ACTUAL ENTRIES
BC00:        0000              EXTRN BITMAPADR        ; BIT MAP DISK ADDRESS IMPOSSIBLE
BC00:                  *
BC00:                  * FATAL ERRORS
BC00:                  *
BC00:        0000              EXTRN SYSDEATH
BC00:                  *
BC00:        0000              EXTRN VCBERR           ; VOLUME CONTROL BLOCK NOT USABLE
BC00:        0000              EXTRN ALCERR           ; ALLOCATION BLOCKS INVALID
BC00:        0000              EXTRN TOOLONG          ; PATHNAME BUFFER OVERFLOW
```

```
BC00:              *
BC00:              * CONSTANTS
BC00:              *
BC00:        002F  DLIMIT     EQU   $2F              ; DELIMITER IS CURRENTLY AN ASCII '/'
BC00:        0001  SEEDTYP    EQU   1
BC00:        0002  SAPTYP     EQU   2
BC00:        0003  TRETYP     EQU   3
BC00:        000D  DIRTYP     EQU   $D
BC00:        000E  HEDTYP     EQU   $E
BC00:        0000  RDCMD      EQU   $0
BC00:        0001  WRTCMD     EQU   $1
BC00:        0009  RPTCMD     EQU   $9
BC00:        0002  STATCMD    EQU   $02              ; REQUEST STATUS OF BLOCK DEVICE. (BIT 0 = WRITE PROTECTED)
BC00:        0000  STATSUB    EQU   $0
BC00:        0020  PRETIME    EQU   $20              ; COMMAND NEEDS CURRENT DATE/TIME STAMP
BC00:        0040  PREREF     EQU   $40              ; COMMAND REQUIRES FCB ADDRESS AND VERIFICATION
BC00:        0080  PREPATH    EQU   $80              ; COMMAND HAS PATHNAME TO PREPROCESS
BC00:        1400  SISTER     EQU   $1400
BC00:              *
BC00:              * VOLUME STATUS CONSTANTS (BITS)
BC00:              *
BC00:        0040  DSWITCH    EQU   $40              ; FOR DISK SWITCHED ERROR RECOVERY.
BC00:              *
BC00:              * FILE STATUS CONSTANTS
BC00:              *
BC00:        0001  DATALC     EQU   $1               ; DATA BLOCK NOT ALLOCATED.
BC00:        0002  IDXALC     EQU   $2               ; INDEX NOT ALLOCATED
BC00:        0004  TOPALC     EQU   $4               ; TOP INDEX NOT ALLOCATED
BC00:        0008  STPMOD     EQU   $8               ; STORAGE TYPE MODIFIED
BC00:        0010  USEMOD     EQU   $10              ; FILE USAGE MODIFIED
BC00:        0020  EOFMOD     EQU   $20              ; END OF FILE MODIFIED
BC00:        0040  DATMOD     EQU   $40              ; DATA BLOCK MODIFIED
BC00:        0080  IDXMOD     EQU   $80              ; INDEX BLOCK MODIFIED
BC00:        0080  FCBMOD     EQU   $80              ; HAS FCB/DIRECTORY BEEN MODIFIED? (FLUSH)
BC00:              *
BC00:              * FILE ATTRIBUTES CONSTANTS
BC00:              *
BC00:        0001  READEN     EQU   $1               ; READ ENABLED
BC00:        0002  WRITEN     EQU   $2               ; WRITE ENABLED
BC00:        0010  NLINEN     EQU   $10              ; NEW LINE ENABLED
BC00:        0020  BKBITVAL   EQU   $20              ; FILE NEEDS BACKUP IF SET (BKBITFLG)
BC00:        0040  RENAMEN    EQU   $40              ; RENAME OK WHEN ON.
BC00:        0080  DSTROYEN   EQU   $80              ; DESTROY OK WHEN ON.
```

```
BC00:              * HEADER INDEX CONSTANTS
BC00:              *
BC00:        0000  HNLEN      EQU   $0               ; HEADER NAME LENGTH (OFFSET INTO HEADER)
BC00:              *HNAME EQU $1 ; HEADER NAME
BC00:        0010  HPENAB     EQU   $10              ; PASSWORD ENABLE BYTE
BC00:        0011  HPASS      EQU   $11              ; ENCODED PASSWORD
BC00:        0018  HCRDT      EQU   $18              ; HEADER CREATION DATE
BC00:              * HCRTM EQU $1A ; HEADER CREATION TIME
BC00:        001C  HVER       EQU   $1C              ; SOS VERSION THAT CREATED DIRECTORY
BC00:        001D  HCMP       EQU   $1D              ; BACKWARD COMPATIBLE WITH SOS VERSION
BC00:        001E  HATTR      EQU   $1E              ; HEADER ATTRIBUTES- PROTECT ETC.
BC00:              * HENTLN EQU $1F ; LENGTH OF EACH ENTRY
BC00:              * HMENT EQU $20 ; MAXIMUM NUMBER OF ENTRIES/BLOCK
BC00:        0021  HCENT      EQU   $21              ; CURRENT NUMBER OF FILES IN DIRECTORY
BC00:        0023  HRBLK      EQU   $23              ; OWNER'S DIRECTORY ADDRESS
BC00:        0025  HRENT      EQU   $25              ; OWNER'S DIRECTORY ENTRY NUMBER
BC00:        0026  HRELN      EQU   $26              ; OWNER'S DIRECTORY ENTRY LENGTH
BC00:        0023  VBMAP      EQU   HRBLK
BC00:        0025  VTBLK      EQU   HRENT            ; (USED FOR ROOT DIRECTORY ONLY)
BC00:              *
BC00:              * VOLUME CONTROL BLOCK INDEX CONSTANTS
BC00:              *
BC00:        0020  VCBSIZE    EQU   $20              ; CURRENT VCB IS 32 BYTES PER ENTRY (VER 0)
BC00:        0000  VCBNML     EQU   0                ; VOLUME NAME LENGTH BYTE
BC00:        0001  VCBNAM     EQU   1                ; VOLUME NAME
BC00:        0010  VCBDEV     EQU   $10              ; VOLUME'S DEVICE
BC00:        0011  VCBSTAT    EQU   $11              ; VOLUME STATUS. (80=FILES OPEN. 40=DISK SWITCHED.)
BC00:        0012  VCBTBLK    EQU   $12              ; TOTAL BLOCKS ON THIS VOLUME
BC00:        0014  VCBTFRE    EQU   $14              ; NUMBER OF UNUSED BLOCKS
BC00:        0016  VCBROOT    EQU   $16              ; ROOT DIRECTORY (DISK) ADDRESS
BC00:              *VCBMORG EQU $18 ; MAP ORGANIZATION (NOT SUPPORTED BY V 0)
BC00:              *VCBMBUF EQU $19 ; BIT MAP BUF NUM
BC00:        001A  VCBDMAP    EQU   $1A              ; FIRST (DISK) ADDRESS OF BITMAP(S)
BC00:        001C  VCBCMAP    EQU   $1C              ; RELATIVE ADDRESS OF BIT MAP WITH SPACE (ADD TO VCBDMAP)
BC00:              *VCBMNUM EQU $1D ; RELATIVE BIT MAP CURRENTLY IN MEMORY
BC00:        001E  VCBOPNC    EQU   $1E              ; CURRENT NUMBER OF OPEN FILES.
BC00:        001F  VCBSWAP    EQU   $1F              ; $8X IF VOLUME SWAPPED; $00 IF UNSWAPPED WHERE X=LOW ORDER BYTE
OF VCB ADR/16
BC00:              *
BC00:              * FILE CONTROL BLOCK INDEX CONSTANTS
BC00:              *
BC00:        0000  FCBREFN    EQU   0                ; FILE REFERENCE NUMBER (POSITION SENSITIVE)
BC00:        0001  FCBDEVN    EQU   1                ; DEVICE (NUMBER) ON WHICH FILE RESIDES
BC00:              *FCBHEAD EQU 2 ; BLOCK ADDRESS OF FILE'S DIRECTORY HEADER
BC00:              *FCBDIRB EQU 4 ; BLOCK ADDRESS OF FILE'S DIRECTORY
BC00:        0006  FCBENTN    EQU   6                ; ENTRY NUMBER WITHIN DIRECTORY BLOCK
BC00:        0007  FCBSTYP    EQU   7                ; STORAGE TYPE - SEED, SAPLING, TREE, ETC.
BC00:        0008  FCBSTAT    EQU   8                ; STATUS - INDEX/DATA/EOF/USAGE/TYPE MODIFIED.
BC00:        0009  FCBATTR    EQU   9                ; ATTRIBUTES - READ/WRITE ENABLE, NEWLINE ENABLE.
BC00:        000A  FCBNEWL    EQU   $A               ; NEW LINE TERMINATOR (ALL 8 BITS SIGNIFICANT).
BC00:        000B  FCBBUFN    EQU   $B               ; BUFFER NUMBER
BC00:        000C  FCBFRST    EQU   $C               ; FIRST BLOCK OF FILE
BC00:        000E  FCBIDXB    EQU   $E               ; BLOCK ADDRESS OF INDEX (0 IF NO INDEX)
BC00:        0010  FCBDATB    EQU   $10              ; BLOCK ADDRESS OF DATA
BC00:        0012  FCBMARK    EQU   $12              ; CURRENT FILE MARKER.
BC00:        0015  FCBEOF     EQU   $15              ; LOGICAL END OF FILE.
BC00:        0018  FCBUSE     EQU   $18              ; ACTUAL NUMBER OF BLOCKS ALLOCATED TO THIS FILE.
```

```
BC00:        001A  172 FCBSWAP    EQU   $1A              ; $8N = SWAPPED, $00 = UNSWAPPED VOLUME ("N" = VCB ENTRY NUMBER)
BC00:        001B  173 FCBLEVL    EQU   $1B              ; LEVEL AT WHICH THIS FILE WAS OPENED
BC00:        001C  174 FCBDIRTY   EQU   $1C              ; FCB MARKED AS MODIFIED
```

```
BC00:              *
BC00:              * ZERO PAGE STUFF
BC00:              *
BC00:        00A0  PAR        EQU   $A0
BC00:        00A0  COMMAND    EQU   PAR
BC00:        00A1  C.DNAMP    EQU   PAR+1
BC00:        00A1  C.PATH     EQU   PAR+1
BC00:        00A1  C.REFNUM   EQU   PAR+1
BC00:        00A2  C.ISNEWL   EQU   PAR+2
BC00:        00A2  C.OUTEOF   EQU   PAR+2
BC00:        00A2  C.BASE     EQU   PAR+2
BC00:        00A2  C.MRKPTR   EQU   PAR+2
BC00:        00A2  C.OUTBUF   EQU   PAR+2
BC00:        00A3  C.NWPATH   EQU   PAR+3
BC00:        00A3  C.FILIST   EQU   PAR+3
BC00:        00A3  C.NEWL     EQU   PAR+3
BC00:        00A3  C.OUTVOL   EQU   PAR+3
BC00:        00A3  C.OUTREF   EQU   PAR+3
BC00:        00A3  C.XLIST    EQU   PAR+3
BC00:        00A3  C.MAXPTH   EQU   PAR+3
BC00:        00A3  C.MARK     EQU   PAR+3
BC00:        00A3  C.NEWEOF   EQU   PAR+3
BC00:        00A4  C.BYTES    EQU   PAR+4
BC00:        00A5  C.FILSTLN  EQU   PAR+5
BC00:        00A5  C.OUTBLK   EQU   PAR+5
BC00:        00A5  C.OPLIST   EQU   PAR+5
BC00:        00A5  C.XLEN     EQU   PAR+5
BC00:        00A6  C.FILID    EQU   PAR+6
BC00:        00A6  C.OUTCNT   EQU   PAR+6
BC00:        00A7  C.OPLSTLN  EQU   PAR+7
BC00:        00A7  C.AUXID    EQU   PAR+7
BC00:        00A9  C.STOR     EQU   PAR+9
BC00:        00AA  C.EOFLL    EQU   PAR+$A
BC00:        00AB  C.EOFLH    EQU   PAR+$B
BC00:        00AC  C.EOFHL    EQU   PAR+$C
BC00:        00AD  DEBUPTR    EQU   PAR+$D           ; NOTE SAME AS BELOW
BC00:        00AD  C.EOFHH    EQU   PAR+$D
BC00:              * C.SPARE EQU PAR+$E
BC00:              *
BC00:        00C0  DEVICE     EQU   $C0
BC00:        00C0  DHPCMD     EQU   DEVICE
BC00:        00C1  UNITNUM    EQU   DEVICE+1
BC00:        00C2  DSTATREQ   EQU   DEVICE+2
BC00:        00C2  DBUFPL     EQU   DEVICE+2
BC00:        00C3  DBUFPH     EQU   DBUFPL+1
BC00:        00C3  DSTATBFL   EQU   DEVICE+3         ; TO PASS BACK BUSY, WRITE PROTECT, READ PROTECT.
BC00:        00C4  DSTATBFH   EQU   DSTATBFL+1
BC00:        00C4  RQCNTL     EQU   DEVICE+4
BC00:        00C5  RQCNTH     EQU   RQCNTL+1
BC00:        00C6  BLOKNML    EQU   DEVICE+6
BC00:        00C7  BLOKNMH    EQU   BLOKNML+1
BC00:        00C8  BRDPTR     EQU   DEVICE+8         ; (AND 9)
BC00:              *
BC00:        00C1  DVNAMP     EQU   DEVICE+1         ; USED FOR 'VOLUME' TO CALL
BC00:        00C3  DVDNUM     EQU   DEVICE+3         ; 'GET.DNUM' IN DEVICE MANAGER.
BC00:              *
```

```
BC00:        14C3  SISBPH     EQU   SISTER+DBUFPH
BC00:        14C4  SISDSTAT   EQU   SISTER+DSTATBFH
BC00:        14C9  SSBRDPH    EQU   SISTER+BRDPTR+1
BC00:              *
```

```
BC00:              *
BC00:              * ZERO PAGE TEMPORARIES
BC00:              *
BC00:        00B0  ZTEMPS     EQU   $B0
BC00:        00B0  PATHNML    EQU   ZTEMPS
BC00:        00B1  PATHNMH    EQU   PATHNML+1
BC00:        00B0  USRBUF     EQU   ZTEMPS
BC00:        00B2  TPATH      EQU   ZTEMPS+2
BC00:        00B4  WRKPATH    EQU   ZTEMPS+4
BC00:        00B2  TINDX      EQU   ZTEMPS+2
BC00:        00B4  DRBUFPL    EQU   ZTEMPS+4
BC00:        00B5  DRBUFPH    EQU   DRBUFPL+1
BC00:        00B6  VCBPTR     EQU   ZTEMPS+6
BC00:        00B8  BMADR      EQU   ZTEMPS+8
BC00:        00BA  FCBPTR     EQU   ZTEMPS+$A
BC00:        00BC  DATPTR     EQU   ZTEMPS+$C
BC00:        00BE  POSPTR     EQU   ZTEMPS+$E
BC00:              *
BC00:        000F  MAXTEMPS   EQU   $F
BC00:        14B0  SISTEMPS   EQU   SISTER+ZTEMPS
BC00:        14B3  SSTIDXH    EQU   SISTER+TINDX+1
BC00:        14A2  SISPATH    EQU   SISTER+C.PATH+1
BC00:        14A4  SSNWPATH   EQU   SISTER+C.NWPATH+1
BC00:        14B1  SISUSRBF   EQU   SISTER+USRBUF+1
BC00:        14A3  SISOUTBF   EQU   SISTER+C.OUTBUF+1
BC00:        14B3  SISTPATH   EQU   SISTER+TPATH+1
BC00:        14B9  SISBMADR   EQU   SISTER+BMADR+1
BC00:        14BB  SISFCBP    EQU   SISTER+FCBPTR+1
BC00:        14BD  SISDATP    EQU   SISTER+DATPTR+1
BC00:        14BF  SISPOSP    EQU   SISTER+POSPTR+1
BC00:              *
BC00:              *
BC00:              * ADDRESSES
BC00:              *
BC00:        1000  PATHBUF    EQU   $1000            ; NOTE: THIS IS $100 BYTES LONG.
BC00:        1100  VCB        EQU   $1100
BC00:        1200  GBUF       EQU   $1200            ; THRU $13FF
BC00:              *
BC00:              * INITIALIZATION EQUATES
BC00:              *
BC00:        001C  BFMFCB1    EQU   $1C              ; FCB PAGE 1 ADDR
BC00:        001D  BFMFCB2    EQU   $1D              ; FCB PAGE 2 ADDR
BC00:        00B8  BMAPAGE    EQU   <$B800           ; BIT MAP A ADDR
BC00:        00BA  BMBPAGE    EQU   <$BA00           ; BIT MAP B ADDR
BC00:        00BA  FCBZPP     EQU   FCBPTR
BC00:              *
BC00:              *
BC00:              *
```

```
0000:                         DSECT
0000:        0000             ORG   $0               ; (THE FOLLOWING DO NOT NEED TO BE ON ZERO PAGE. 7/16/80 JRH.)
0000:        0001      DATBLKL    DS    1
0001:        0001      DATBLKH    DS    1
0002:        0001      IDXADRL    DS    1               ; DISK ADDRESS OF INDEX BLOCK
0003:        0001      IDXADRH    DS    1
0004:        0001      REQL       DS    1
0005:        0001      REQH       DS    1
0006:        0001      INDXBLK    DS    1
0007:        0001      LEVELS     DS    1
0008:        0001      TOTENT     DS    1
0009:        0001      ENTCNTL    DS    1
000A:        0001      ENTCNTH    DS    1
000B:        0001      CNTENT     DS    1
000C:        0001      NOFREE     DS    1
000D:        0001      BMCNT      DS    1
000E:        0001      SAPTR      DS    1
000F:        0001      TREPTR     DS    1
0010:        0002      TLINK      DS    2
0012:        0002      FLINK      DS    2
0014:        0001      PATHCNT    DS    1
0015:        0002      PFIXPTR    DS    2
0017:        0001      BMPTR      DS    1
0018:        0001      BASVAL     DS    1
0019:        0001      HALF       DS    1
001A:                  *
001A:                  *
```

```
001A:                 *
001A:                 * BIT MAP INFO TABLES (A & B)
001A:                 *
001A:        0006     BMTABSZ    EQU   $6
001A:        0001     BMTAB      DS    1
001B:        0001     BMBUFBNK   DS    1
001C:        0001     BMASTAT    DS    1
001D:        0001     BMADEV     DS    1
001E:        0001     BMAMADR    DS    1
001F:        0002     BMADADR    DS    2
0021:        0001     BMACMAP    DS    1                ; SIMILAR TO VCBCMAP
0022:        0001     BMBSTAT    DS    1
0023:        0001     BMBDEV     DS    1
0024:        0001     BMBMADR    DS    1
0025:        0002                DS    2                ; BMBDADR
0027:        0001                DS    1                ; BMBCMAP
0028:                 *
0028:        0001     FCBADDRH   DS    1                ; FILE CONTROL BLOCK'S BUFFER ADDRESS.
0029:        0001     FCBANKNM   DS    1                ; AND BANK (SISTER PAGE) BYTE.
002A:        0001     TPOSLL     DS    1
002B:        0001     TPOSLH     DS    1
002C:        0001     TPOSHI     DS    1
002D:        0001     RWREQL     DS    1
002E:        0001     RWREQH     DS    1
002F:        0001     BULKCNT    DS    1
0030:        0001     NLCHAR     DS    1
0031:        0003     NPATHDEV   DS    3                ; FOR NEW PATHNAME DEVICE AND DIRECTORY HEADER ADDRESS
0034:        0001     IOACCESS   DS    1                ; USED TO DETERMINE IF A CALL HAS BEEN MADE TO THE DISK DEVICE
HANDLER
0035:        0001     DEVNUM     DS    1                ; CURRENT DEVICE TO BE ACCESSED.
0036:        0001     TOTDEVS    DS    1                ; USED FOR ACCESSING DRIVES IN NUMERIC ORDER
0037:        0001     CMDTEMP    DS    1                ; USED FOR TESTING REFNUM, TIME, AND DSKSWTCH (PRE)PROCESSING.
0038:        0001     DATELO     DS    1                ; DATE AND TIME MUST RESIDE ON ZERO PAGE.
0039:        0001     DATEHI     DS    1
003A:        0001     TIMELO     DS    1
003B:        0001     TIMEHI     DS    1
003C:                 *
003C:        0001     DUPLFLAG   DS    1                ; USED FOR DIFFERENCE BETWEEN VNFERR AND DUPVOL BY SYNPATH
003D:        0001     ZPGTEMP    DS    1                ; A ONE-BYTE UNSTABLE TEMPORARY
003E:        0001     VCBENTRY   DS    1                ; POINTER TO CURRENT VCB ENTRY
003F:                 *
BC00:                            DEND
BC00:                 *
BC00:                            CHN   PATH
```

```
BC00:             *
BC00:             *
BC00:             *
BC00:A6 A0        BFMGR      LDX   COMMAND          ; WHAT CALL?
BC02:             *
BC02:             *
BC02:             *
BC02:BD C3 BC                LDA   DISPTCH,X        ; TRANSLATE INTO COMMAND ADDRESS
BC05:0A                      ASL   A                ; (BIT 7 INDICATES IT'S GOT A PATHNAME TO PREPROCESS)
BC06:85 37                   STA   CMDTEMP
BC08:29 3F                   AND   #$3F             ; (BIT 6 IS REFNUM PREPROCESS, 5 IS FOR TIME, SO STRIP EM.)
BC0A:AA                      TAX
BC0B:BD 9F BC                LDA   CMDTABLE,X       ; MOVE ADDRESS FOR INDIRECT JUMP.
BC0E:8D E1 DB                STA   CMDADR
BC11:BD A0 BC                LDA   CMDTABLE+1,X     ; (HIGH BYTE)
BC14:8D E2 DB                STA   CMDADR+1
BC17:A9 11                   LDA   #<VCB
BC19:85 B7                   STA   VCBPTR+1         ; INSURE DEFAULT HI ADDRESS TO VCB BEFORE CALLS
BC1B:A9 20                   LDA   #BKBITVAL        ; INIT "BACKUP BIT FLAG"
BC1D:8D 57 D9                STA   BKBITFLG         ; TO SAY "FILE MODIFIED"
BC20:A0 0F                   LDY   #MAXTEMPS        ; ZERO OUT SISTER PAGE FOR TEMPS
BC22:A9 00                   LDA   #0
BC24:8D 00 00                STA   SERR             ; MAKE GLOBAL ERROR SAY "NONE"
BC27:8D BB D5                STA   DSWGLOB          ; "DISK SWITCH GLOBAL"
BC2A:85 3C                   STA   DUPLFLAG         ; "DUPLICATE VOLUME ON LINE"
BC2C:8D 17 C5                STA   CFLAG            ; SET "CREATE" TO NO
BC2F:8D 1A C5                STA   BLOKSAVE
BC32:8D 1B C5                STA   BLOKSAVE+1       ; SET PARENT DIRECTORY TO NULL
BC35:99 B0 14     CLRSIS     STA   SISTEMPS,Y
BC38:88                      DEY
BC39:10 FA   BC35            BPL   CLRSIS           ; CARRY IS UNDISTURBED BY THIS LOOP
BC3B:90 05   BC42            BCC   NOPATH
BC3D:20 D5 BC                JSR   SETPATH          ; GO PROCESS PATHNAME BEFORE CALLING COMMAND
BC40:B0 56   BC98            BCS   ERRORSYS         ; BRANCH IF BAD NAME.
BC42:06 37        NOPATH     ASL   CMDTEMP          ; TEST FOR REFNUM PREPROCESSING
BC44:90 05   BC4B            BCC   NOPREREF
BC46:20 75 BE                JSR   FINDFCB          ; GO SET UP POINTERS TO FCB AND VCB OF THIS FILE.
BC49:B0 4D   BC98            BCS   ERRORSYS         ; BRANCH IF ANY ERRORS ARE ENCOUNTERED.
BC4B:06 37        NOPREREF   ASL   CMDTEMP          ; LASTLY CHECK FOR NECESSITY OF TIME STAMP.
BC4D:90 05   BC54            BCC   TSWVRFY
BC4F:A2 38                   LDX   #DATELO          ; PASS Z PAGE ADDRESS OF WHERE TO RETURN DATE/TIME
BC51:20 00 00                JSR   DATETIME         ; (NO ERROR POSIBLE)
BC54:A6 A0        TSWVRFY    LDX   COMMAND          ; TEST FOR NECESSITY OF VOLUME VERIFICATION
BC56:A9 E0                   LDA   #PREPATH+PREREF+PRETIME ; TO ENSURE VCB IS SET
BC58:3D C3 BC                AND   DISPTCH,X
BC5B:F0 23   BC80            BEQ   EXECUTE
BC5D:A0 11                   LDY   #VCBSTAT
BC5F:B1 B6                   LDA   (VCBPTR),Y
BC61:29 40                   AND   #DSWITCH         ; WAS THE VOLUME PREVIOUSLY SWITCHED?
BC63:F0 1B   BC80            BEQ   EXECUTE
BC65:88                      DEY                    ; GET DEVICE NUMBER
BC66:B1 B6                   LDA   (VCBPTR),Y
BC68:85 35                   STA   DEVNUM
BC6A:20 0A C9     DVERIFY    JSR   VERFYVOL         ; SEE IF PROPER VOLUME NOW ON LINE
BC6D:90 09   BC78            BCC   CLRDSWT          ; BRANCH IF YES
BC6F:20 2F DD                JSR   USRREQ           ; OTHERWISE REQUEST IT BE PUT ON LINE
```

```
BC72:90 F6   BC6A                BCC   DVERIFY          ; USER SEZ S/HE DID: CHECK IT OUT
BC74:A9 00                       LDA   #VNFERR          ; VOLUME NOT FOUND IF USER REFUSES
BC76:D0 20   BC98                BNE   ERRORSYS          ; REPORT ERROR (BRANCH ALWAYS)
BC78:A0 11          CLRDSWT      LDY   #VCBSTAT         ; GET VOLUME
BC7A:B1 B6                       LDA   (VCBPTR),Y       ; STATUS
BC7C:29 BF                       AND   #$FF-DSWITCH     ; TURN OFF DISK SWITCH
BC7E:91 B6                       STA   (VCBPTR),Y       ; SO WE WON'T VERIFY NEXT TIME
BC80:20 9C BC       EXECUTE      JSR   GOCMD            ; EXECUTE COMMAND
BC83:90 16   BC9B                BCC   GOODOP           ; BRANCH IF SUCCESSFUL
BC85:C9 00                       CMP   #XDISKSW         ; DISK SWITCH?
BC87:D0 0F   BC98                BNE   ERRORSYS         ; NO, REPORT SOME OTHER
BC89:A0 11                       LDY   #VCBSTAT         ; MARK VCB WITH SWITCH
BC8B:B1 B6                       LDA   (VCBPTR),Y
BC8D:29 BF                       AND   #$FF-DSWITCH     ; TO ENSURE VOLUME VERIFIED
BC8F:10 02   BC93                BPL   ERRCMD           ; NO FILES OPEN SO DSWITCH CANT APPLY
BC91:09 40                       ORA   #DSWITCH
BC93:91 B6          ERRCMD       STA   (VCBPTR),Y
BC95:4C 00 BC                    JMP   BFMGR            ; TRY THE COMMAND AGAIN
BC98:               *
BC98:20 00 00       ERRORSYS     JSR   SYSERR
BC9B:60             GOODOP       RTS                    ; GOOD RETURN
BC9C:               *
BC9C:6C E1 DB       GOCMD        JMP   (CMDADR)
BC9F:               *
```

```
BC9F:              *
BC9F:        BC9F  CMDTABLE   EQU   *
BC9F:F1 C0                    DW    CREATE
BCA1:71 DA                    DW    DESTROY
BCA3:68 D9                    DW    RENAME
BCA5:10 D9                    DW    SETINFO
BCA7:AF D8                    DW    GETINFO
BCA9:DE BF                    DW    VOLUME
BCAB:08 BE                    DW    SETPREFX
BCAD:3D BE                    DW    GETPREFX
BCAF:B0 CF                    DW    OPEN
BCB1:93 D8                    DW    NEWLINE
BCB3:54 D1                    DW    READ
BCB5:58 D3                    DW    WRITE
BCB7:D5 D5                    DW    CLOSE
BCB9:49 D6                    DW    FLUSH
BCBB:B2 CC                    DW    SETMARK
BCBD:9B CC                    DW    GETMARK
BCBF:90 D7                    DW    SETEOF
BCC1:7E D8                    DW    GETEOF
BCC3:              *
BCC3:        BCC3  DISPTCH    EQU   *
BCC3:A0                       DFB   PREPATH+PRETIME+0 ; CREATE
BCC4:A1                       DFB   PREPATH+PRETIME+1 ; DESTROY
BCC5:A2                       DFB   PREPATH+PRETIME+2 ; RENAME
BCC6:A3                       DFB   PREPATH+PRETIME+3 ; SETINFO
BCC7:84                       DFB   PREPATH+4         ; GETINFO
BCC8:05                       DFB   5                 ; VOLUME
BCC9:06                       DFB   6                 ; SETPREFIX, PATHNAME MOVED TO PREFIX BUFFER
BCCA:07                       DFB   7                 ; GETPREFIX
BCCB:88                       DFB   PREPATH+8         ; OPEN
BCCC:49                       DFB   PREREF+$9         ; NEWLINE
BCCD:4A                       DFB   PREREF+$A         ; READ
BCCE:4B                       DFB   PREREF+$B         ; WRITE
BCCF:2C                       DFB   PRETIME+$C        ; CLOSE
BCD0:2D                       DFB   PRETIME+$D        ; FLUSH, REFNUM MAY BE ZERO TO FLUSH ALL.
BCD1:4E                       DFB   PREREF+$E         ; SETMARK
BCD2:4F                       DFB   PREREF+$F         ; GETMARK
BCD3:50                       DFB   PREREF+$10        ; SET EOF
BCD4:51                       DFB   PREREF+$11        ; GET EOF
BCD5:              *
```

```
BCD5:               *
BCD5:A5 A1          SETPATH    LDA   C.PATH           ; FOR A REGULAR PATHNAME,
BCD7:85 B2                     STA   TPATH            ; SET UP TEMP POINTER TO PROCESS
BCD9:A5 A2                     LDA   C.PATH+1         ; PATHNAME AND CHECK FOR SYNTAX ERRORS
BCDB:85 B3                     STA   TPATH+1
BCDD:AD A2 14                  LDA   SISPATH
BCE0:8D B3 14                  STA   SISTPATH         ; (LEAVE CALL PARAMETERS ALONE!)
BCE3:               * DROP INTO 'SYNPATH'
BCE3:               *
BCE3:A9 00          SYNPATH    LDA   #>PATHBUF        ; SET UP DEFAULT ADDRESS FOR
BCE5:85 B0                     STA   PATHNML          ; SYNTAXED PATHNAME -
BCE7:85 B4                     STA   WRKPATH          ; LENGTH, NAME, LENGTH, NAME, ETC...
BCE9:A9 10                     LDA   #<PATHBUF
BCEB:85 B1                     STA   PATHNMH
BCED:85 B5                     STA   WRKPATH+1        ; (ASSUMES FULL PATHNAME, NO PREFIX).
BCEF:A2 00                     LDX   #0               ; USE INDEXED INDIRECT FOR SOURCE PATHNAME
BCF1:8A                        TXA                    ; SET BEGINNING OF PATH
BCF2:81 B0                     STA   (PATHNML,X)      ; TO ZERO TO INDICATE NOTHING PROCESSED.
BCF4:A8                        TAY
BCF5:A1 B2                     LDA   (TPATH,X)        ; GET TOTAL LENGTH OF SOURCE PATHNAME
BCF7:30 76   BD6F              BMI   ERRSYN
BCF9:F0 74   BD6F              BEQ   ERRSYN
BCFB:85 14                     STA   PATHCNT          ; (THIS IS USED AS A 'COUNT-DOWN')
BCFD:20 01 BE                  JSR   INCTPTH          ; INCREMENT SOURCE POINTER
BD00:A1 B2                     LDA   (TPATH,X)        ; GET FIRST CHARACTER OF PATHNAME
BD02:C9 2F                     CMP   #DLIMIT          ; IS IT A FULL PATHNAME (NO PREFIX)?
BD04:F0 79   BD7F              BEQ   BUMPATH          ; YES, WE'RE READY TO DO IT.
BD06:C9 2E                     CMP   #$2E             ; IS IT A DRIVE NAME '.'?
BD08:D0 69   BD73              BNE   ADPREFIX         ; NO, ADD PREFIX TO BEGINNING
BD0A:A1 B2          DRIVENAM   LDA   (TPATH,X)        ; MOVE DRIVE NAME FOR VOLUME CALL
BD0C:C9 2F                     CMP   #DLIMIT          ; HAVE WE MOVED ENTIRE NAME?
BD0E:F0 0C   BD1C              BEQ   PREVOLM          ; YES, PROCESS IT.
BD10:C8                        INY                    ; (IF THIS IS THE FIRST, MAKE ROOM FOR LENGTH OF NAME)
BD11:91 B4                     STA   (WRKPATH),Y
BD13:20 01 BE                  JSR   INCTPTH          ; BUMP POINTER TO GIVEN NAME.
BD16:C6 14                     DEC   PATHCNT
BD18:D0 F0   BD0A              BNE   DRIVENAM
BD1A:F0 05   BD21              BEQ   PREVOLM1
BD1C:               *
```

```
BD1C:20 01 BE      PREVOLM    JSR   INCTPTH          ; MAKE IT SO POINTING PAST DELIMITER.
BD1F:C6 14                    DEC   PATHCNT
BD21:98            PREVOLM1   TYA                    ; SAVE LENGTH OF DRIVE NAME.
BD22:81 B4                    STA   (WRKPATH,X)
BD24:A9 00                    LDA   #>PATHBUF        ; POINT AT PATHNAME BUFFER FOR DEVICE ID CALL.
BD26:85 C1                    STA   DVNAMP
BD28:A9 10                    LDA   #<PATHBUF
BD2A:85 C2                    STA   DVNAMP+1
BD2C:A9 00                    LDA   #0               ; MAKE VIRTUAL POINT AT SWITCHED IN BANK.
BD2E:8D C2 14                 STA   SISTER+DVNAMP+1
BD31:20 24 BF                 JSR   SRCHDEV          ; GO IDENTIFY WHICH VOLUME
BD34:90 0B   BD41             BCC   PREVOLM2         ; BRANCH IF NO ERROR
BD36:C9 00                    CMP   #VNFERR          ; WAS IT REPORTED AS 'VOLUME NOT FOUND'?
BD38:D0 37   BD71             BNE   SPTHERR          ; NO SOME OTHER ERROR WAS ENCOUNTERED.
BD3A:A6 3C                    LDX   DUPLFLAG         ; YES, WAS IT NOT FOUND BECAUSE SOME OTHER 'OPEN' VOLUME HAS
SAME NAME?
BD3C:F0 33   BD71             BEQ   SPTHERR          ; NO, IT SIMPLY WASN'T FOUND.
BD3E:A9 00                    LDA   #DUPVOL          ; (CARRY IS SET)
BD40:60                       RTS
BD41:              *
BD41:A0 00         PREVOLM2   LDY   #0               ; (X CONTAINS AN INDEX TO VCB)
BD43:BD 00 11                 LDA   VCB,X            ; GET VOLUME NAME LENGTH.
BD46:99 00 10                 STA   PATHBUF,Y
BD49:E8            SPATH2     INX                    ; MOVE VOLUME NAME INTO PATH NAME BUFFER IN
BD4A:C8                       INY                    ; PLACE OF DISK DEVICE NAME ('.D1' OR SIMULAR)
BD4B:BD 00 11                 LDA   VCB,X
BD4E:99 00 10                 STA   PATHBUF,Y
BD51:CC 00 10                 CPY   PATHBUF          ; HAVE ALL CHARACTERS BEEN MOVED?
BD54:D0 F3   BD49             BNE   SPATH2
BD56:A2 00                    LDX   #0               ; RESET X FOR INDEXING
BD58:86 B0                    STX   PATHNML
BD5A:A9 10                    LDA   #<PATHBUF
BD5C:85 B1                    STA   PATHNMH
BD5E:A5 14                    LDA   PATHCNT          ; IS THAT ALL THERE IS?
BD60:D0 04   BD66             BNE   SPATH3           ; NO, MORE TO COME...
BD62:18                       CLC
BD63:4C D6 BD                 JMP   ENDPATH
BD66:              *
BD66:C8            SPATH3     INY                    ; BUMP TO END OF NAME+1
BD67:84 B4                    STY   WRKPATH          ; RESET WORKPATH POINTER TO CURRENT.
BD69:A9 00                    LDA   #0               ; RESET PATHNAME BUFFER POINTER.
BD6B:A0 10                    LDY   #<PATHBUF
BD6D:D0 0A   BD79             BNE   NOPREFX          ; BRANCH ALWAYS...
BD6F:              *
BD6F:A9 00         ERRSYN     LDA   #BADPATH         ; RETURN SYNTAX ERROR
BD71:38            SPTHERR    SEC
BD72:60                       RTS
BD73:              *
BD73:AD 15 00      ADPREFIX   LDA   PFIXPTR          ; GET POINTER TO BEGINNING OF THE
BD76:AC 16 00                 LDY   PFIXPTR+1        ; PREFIX.
BD79:85 B0         NOPREFX    STA   PATHNML
BD7B:84 B1                    STY   PATHNMH          ; IF NO PRESET PREFIX, THIS IS THE SAME AS
BD7D:D0 08   BD87             BNE   FRSTCHAR         ; PATHBUF ADDRESS. (BRANCH ALWAYS TAKEN)
BD7F:              *
```

```
BD7F:               *
BD7F:C6 14          BUMPPATH   DEC   PATHCNT          ; FIRST ADJUST COUNT
BD81:18                        CLC                    ; (JUST IN CASE OF LAST CHARACTER)
BD82:F0 52   BDD6              BEQ   ENDPATH          ; (MUST OF HAD TRAILING SPACES)
BD84:20 01 BE                  JSR   INCTPTH
BD87:A0 00          FRSTCHAR   LDY   #0               ; INIT COUNT FOR THIS PORTION OF THE
BD89:98                        TYA                    ; PATHNAME. ALSO PRESET LENGTH TO ZERO IN
BD8A:81 B4                     STA   (WRKPATH,X)      ; CASE OF TRAILING SPACES.
BD8C:A1 B2                     LDA   (TPATH,X)        ; GET CHARACTER.
BD8E:29 7F                     AND   #$7F             ; IGNORE HIGH BIT.
BD90:C9 20                     CMP   #$20             ; IS IT A LEADING SPACE?
BD92:F0 EB   BD7F              BEQ   BUMPPATH         ; IF SO, IGNORE IT.
BD94:C9 5B                     CMP   #$5B             ; IS IT GREATER THAN (UPPER CASE) A 'Z'?
BD96:90 06   BD9E              BCC   ALFA1            ; NO, MAKE SURE IT'S AN ALPHA CHARACTER
BD98:29 5F                     AND   #$5F             ; YES, ASSUME IT'S LOWER CASE, AND UPSHIFT
BD9A:C9 5B                     CMP   #$5B             ; WAS IT TRULY LOWER CASE?
BD9C:B0 D1   BD6F              BCS   ERRSYN           ; NO, GIVE ERROR.
BD9E:               *
BD9E:C9 41          ALFA1      CMP   #$41             ; IS IT LESS THAN 'A'?
BDA0:90 CD   BD6F              BCC   ERRSYN           ; YES! IT'S CRAP...
BDA2:B0 22   BDC6              BCS   SAVPATH          ; NO, IT'S GOOD. SAVE IT.
BDA4:               *
BDA4:A1 B2          NXTCHAR    LDA   (TPATH,X)        ; GET THE NEXT CHARACTER.
BDA6:29 7F                     AND   #$7F             ; THESE CHARACTERS MAY BE ALPHA, NUMERIC,
BDA8:C9 5B                     CMP   #$5B             ; OR A PERIOD - ONLY THE FIRST HAD TO BE ALPHA
BDAA:90 06   BDB2              BCC   ALFA2            ; BRANCH IF LESS THAN 'Z'
BDAC:29 5F                     AND   #$5F             ; UPSHIFT LOWER CASE.
BDAE:C9 5B                     CMP   #$5B             ; NOW IS IT VALID?
BDB0:B0 BD   BD6F              BCS   ERRSYN           ; NOPE.
BDB2:               *
BDB2:C9 41          ALFA2      CMP   #$41             ; IS IT GREATER THAN 'A'?
BDB4:B0 10   BDC6              BCS   SAVPATH          ; YUP, IT IS WORTH SAVIN.
BDB6:C9 3A                     CMP   #$3A             ; >9?
BDB8:B0 04   BDBE              BCS   TSTDLIM          ; YES
BDBA:C9 30                     CMP   #$30             ; NO, <0?
BDBC:B0 08   BDC6              BCS   SAVPATH          ; NO, IT'S VALID NUMERIC.
BDBE:C9 2F          TSTDLIM    CMP   #DLIMIT          ; IS IT THE DELIMITER?
BDC0:F0 14   BDD6              BEQ   ENDPATH          ; YES. CARRY SET INDICATES MORE TO COME.
BDC2:C9 2E                     CMP   #$2E             ; IS IT A '.' (PERIOD)?
BDC4:D0 A9   BD6F              BNE   ERRSYN           ; NO, IT'S AN ERROR (#@&##@!)
BDC6:18             SAVPATH    CLC
BDC7:C8                        INY                    ; BUMP NAME LENGTH
BDC8:91 B4                     STA   (WRKPATH),Y
BDCA:C6 14                     DEC   PATHCNT          ; IF ZERO, THAT WAS THE LAST CHARACTER
BDCC:F0 08   BDD6              BEQ   ENDPATH          ; (CARRY CLEAR INDICATES END OF PATH)
BDCE:E6 B2                     INC   TPATH            ; BUMP POINTER TO SOURCE PATHNAME.
BDD0:D0 D2   BDA4              BNE   NXTCHAR
BDD2:E6 B3                     INC   TPATH+1          ; HIGH ORDER, WHEN NECESSARY.
BDD4:D0 CE   BDA4              BNE   NXTCHAR          ; BRANCH ALWAYS.
```

```
BDD6:              *
BDD6:98            ENDPATH    TYA                     ; GET CURRENT NAME LENGTH
BDD7:81 B4                    STA   (WRKPATH,X)       ; AND PUT IT IN FRONT OF NAME
BDD9:90 12   BDED             BCC   LSTNAME           ; BRANCH IF THAT WAS THE LAST OF PATH
BDDB:C9 10                    CMP   #$10              ; WAS THE NAME ILLEGALLY LONG?
BDDD:B0 1F   BDFE             BCS   ERRSYN1           ; YES, REPORT IT.
BDDF:A0 00                    LDY   #0
BDE1:38                       SEC                     ; ADJUST WORK POINTER TO END OF PREVIOUS NAME.
BDE2:65 B4                    ADC   WRKPATH
BDE4:85 B4                    STA   WRKPATH           ; REPLACE OLD POINTER.
BDE6:90 97   BD7F             BCC   BUMPATH           ; DO NEXT NAME.
BDE8:A9 00                    LDA   #TOOLONG          ; THIS IS A NEVER ERROR!
BDEA:20 00 00                 JSR   SYSDEATH          ; (NEVER RETURNS).
BDED:              *
BDED:F0 07   BDF6  LSTNAME    BEQ   TSTVALD
BDEF:C9 10                    CMP   #$10              ; MAKE SURE LAST ISN'T TOO LONG
BDF1:B0 0B   BDFE             BCS   ERRSYN1
BDF3:C8                       INY                     ; PUT A ZERO AT END OF PROCESSED PATHNAME
BDF4:A9 00                    LDA   #0
BDF6:91 B4         TSTVALD    STA   (WRKPATH),Y
BDF8:A1 B0                    LDA   (PATHNML,X)       ; SURE THERE IS A PATHNAME
BDFA:F0 02   BDFE             BEQ   ERRSYN1           ; IF NOT, REPORT ERROR.
BDFC:18                       CLC                     ; INDICATE NO ERROR.
BDFD:60                       RTS
BDFE:              *
BDFE:4C 6F BD      ERRSYN1    JMP   ERRSYN
BE01:              *
BE01:E6 B2         INCTPTH    INC   TPATH             ; POINT AT NEXT CHARACTER
BE03:D0 02   BE07             BNE   INCPTH1
BE05:E6 B3                    INC   TPATH+1
BE07:60            INCPTH1    RTS
BE08:              *
```

```
BE08:20 D5 BC      SETPREFX   JSR   SETPATH          ; CALL IS MADE HERE SO A 'NUL' PATH MAY BE DETECTED.
BE0B:90 0E   BE1B             BCC   SETPRFX1         ; BRANCH IF PATHNAME OK
BE0D:AA                       TAX                    ; SAVE ERROR CODE
BE0E:A0 00                    LDY   #0
BE10:B1 A1                    LDA   (C.PATH),Y       ; TEST FOR A NUL PATHNAME
BE12:F0 02   BE16             BEQ   RESETPFX         ; BRANCH IF PREFIX TO BE RESET.
BE14:8A                       TXA                    ; RESTORE ERROR CODE
BE15:60                       RTS
BE16:8D 15 00      RESETPFX   STA   PFIXPTR
BE19:18                       CLC
BE1A:60                       RTS
BE1B:A5 B0         SETPRFX1   LDA   PATHNML          ; MAKE SURE NAME STARTED WITH A '/' DELIMITER.
BE1D:D0 DF   BDFE             BNE   ERRSYN1          ; BRANCH IF IT DID.
BE1F:A4 B4                    LDY   WRKPATH          ; FIND THE END OF THE INPUT PREFIX
BE21:18                       CLC                    ; ADD LAST LOCAL NAME LENGTH TO FIND TRUE END.
BE22:B1 B0                    LDA   (PATHNML),Y
BE24:D0 04   BE2A             BNE   SETPRFX3
BE26:88                       DEY
BE27:98                       TYA
BE28:D0 03   BE2D             BNE   SETPRFX4
BE2A:65 B4         SETPRFX3   ADC   WRKPATH
BE2C:A8                       TAY
BE2D:49 FF         SETPRFX4   EOR   #$FF             ; GET COMPLIMENT TO FIND BEGINNING ADDRESS.
BE2F:8D 15 00                 STA   PFIXPTR          ; OF NEW PREFIX IN THE PREFIX BUFFER
BE32:85 B4                    STA   WRKPATH          ; (PREFIX ALWAYS ENDS AT THE LAST BYTE OF BUFFER)
BE34:B1 B0         MOVPRFX    LDA   (PATHNML),Y
BE36:91 B4                    STA   (WRKPATH),Y      ; MOVE IN NEW PREFIX
BE38:88                       DEY
BE39:10 F9   BE34             BPL   MOVPRFX
BE3B:18                       CLC                    ; AND WE'RE FINISHED!
BE3C:60                       RTS                    ; NO ERRORS POSIBLE FROM THIS ROUTINE.
BE3D:              *
```

```
BE3D:                  *
BE3D:18                GETPREFX   CLC                    ; CALCULATE HOW BIG A BUFFER IS NEEDED TO
BE3E:AD 15 00                     LDA   PFIXPTR          ; PASS THE PREFIX BACK TO THE USER.
BE41:49 FF                        EOR   #$FF             ; (EVEN IF NO PREFIX, 1 BYTE IS NEEDED TO SHOW 0 LENGTH)
BE43:69 02                        ADC   #2               ; ADD 2 FOR LEADING AND ENDING "/".
BE45:C5 A3                        CMP   C.MAXPTH         ; IS THERE ENOUGH SPACE IN USER'S BUFFER?
BE47:90 03   BE4C                 BCC   SENDPRFX         ; BRANCH IF YES
BE49:A9 00                        LDA   #BTSERR          ; TELL USER BUFFER IS TOO SMALL.
BE4B:60                           RTS                    ; (CARRY IS SET TO INDICATE ERROR.)
BE4C:                  *
BE4C:A0 00             SENDPRFX   LDY   #0               ; SAVE TOTAL LENGTH OF STRING TO BE RETURNED
BE4E:91 A1                        STA   (C.PATH),Y
BE50:A8                           TAY
BE51:88                           DEY                    ; DISCOUNT TRAILING DELIMITER.
BE52:F0 1C   BE70                 BEQ   NULPREFX         ; BRANCH IF PREFIX IS SET TO NUL.
BE54:C8                           INY
BE55:AE 15 00                     LDX   PFIXPTR          ; GET BEGINNING ADDRESS OF PREFIX AGAIN
BE58:CA                           DEX
BE59:86 B4                        STX   WRKPATH
BE5B:A9 10                        LDA   #<PATHBUF
BE5D:85 B5                        STA   WRKPATH+1
BE5F:A9 2F             SNDLMIT    LDA   #DLIMIT          ; PLACE DELIMITER BEFORE, BETWEEN, AND AFTER LOCAL NAMES.
BE61:91 A1                        STA   (C.PATH),Y
BE63:88                SNDPRFX1   DEY
BE64:F0 0D   BE73                 BEQ   GOTPRFX          ; BRANCH IF ALL OF PREFIX IS TRANSFERED.
BE66:B1 B4                        LDA   (WRKPATH),Y
BE68:91 A1                        STA   (C.PATH),Y       ; ASSUME IT'S A CHARACTER.
BE6A:29 F0                        AND   #$F0             ; NOW TEST TO SEE IF IT WAS A LOCAL LENGTH.
BE6C:F0 F1   BE5F                 BEQ   SNDLMIT          ; BRANCH IF IT WAS.
BE6E:D0 F3   BE63                 BNE   SNDPRFX1         ; GO MOVE NEXT CHAR IF IT WASN'T (ALWAYS TAKEN).
BE70:98                NULPREFX   TYA                    ; RETURN NUL STRING.
BE71:91 A1                        STA   (C.PATH),Y
BE73:18                GOTPRFX    CLC                    ; INDICATE NO ERROR.
BE74:60                           RTS
```

```
BE75:              *
BE75:AD 28 00      FINDFCB    LDA   FCBADDRH         ; INITIALIZE INDIRECT POINTER TO
BE78:85 BB                    STA   FCBPTR+1         ; FILE CONTROL BLOCK (ALLOCATED WHEN SYSTEM
BE7A:A9 00                    LDA   #0               ; WAS FIRST BOOTED).
BE7C:85 BA                    STA   FCBPTR           ; NOTE: ALWAYS STARTS ON PAGE BOUNDARY.
BE7E:A5 29                    LDA   FCBANKNM         ; SET SISTE PAGE BYTE TOO...
BE80:8D BB 14                 STA   SISFCBP
BE83:A4 A1                    LDY   C.REFNUM         ; GET REQUESTED REFERENCE
BE85:30 7A   BF01             BMI   ERRNOTBLK        ; BRANCH IF IT'S NOT A BLOCK DEVICE REFERENCE
BE87:88                       DEY                    ; (SHOULD BE IN THE RANGE OF 1-16 BEFORE DECREMENT)
BE88:C0 10                    CPY   #$10             ; IS IT A VALID REFNUM?
BE8A:B0 71   BEFD             BCS   REEFER           ; NO, THE USER'S SMOKIN DOPE!
BE8C:98                       TYA                    ; TO FIND ASSOCIATED FILE CONTROL STUFF,
BE8D:0A                       ASL   A                ; MULTIPLY (REFNUM-1) BY 32.
BE8E:0A                       ASL   A
BE8F:0A                       ASL   A
BE90:0A                       ASL   A
BE91:0A                       ASL   A
BE92:90 02   BE96             BCC   SVFCBLO          ; BRANCH IF IT'S WITHIN FIRST HALF OF FCB
BE94:E6 BB                    INC   FCBPTR+1         ; BUMP TO SECOND HAVE (REFNUM>8)
BE96:85 BA         SVFCBLO    STA   FCBPTR           ; SAVE LOW ADDRESS OF REFERENCED FCB
BE98:A5 A1                    LDA   C.REFNUM         ; NOW VERIFY THAT FILE IS OPEN.
BE9A:A0 00                    LDY   #FCBREFN
BE9C:D1 BA                    CMP   (FCBPTR),Y       ; SHOULD BE EQUAL!
BE9E:D0 59   BEF9             BNE   ERRNOREF         ; BRANCH IF THEY'RE NOT
BEA0:A0 0B         FNDFCBUF   LDY   #FCBBUFN         ; IT'S A LEGAL FILE, NOW SET UP
BEA2:B1 BA                    LDA   (FCBPTR),Y        ; INDIRECT POINTERS TO DATA
BEA4:A2 BC         GTBUFFRS   LDX   #DATPTR          ; (AND INDEX) BUFFER(S) IN ZERO PAGE
BEA6:20 00 00                 JSR   GETBUFADR        ; GET BUFFER ADDRESS UNLESS
BEA9:B0 55   BF00             BCS   REEFER1          ; BOB HAS BEEN SMOKIN DOPE...
BEAB:A9 02                    LDA   #2               ; (ASSUME AN INDEX BLOCK BUFFER IS ALSO PRESENT)
BEAD:65 BD                    ADC   DATPTR+1
BEAF:85 B3                    STA   TINDX+1
BEB1:A5 BC                    LDA   DATPTR
BEB3:85 B2                    STA   TINDX
BEB5:AD BD 14                 LDA   SISDATP
BEB8:8D B3 14                 STA   SSTIDXH
BEBB:A0 01                    LDY   #FCBDEVN
BEBD:B1 BA                    LDA   (FCBPTR),Y       ; MAKE SURE DEVICE
BEBF:8D B4 DB                 STA   D.DEV            ; NUMBER TEMPS MATCH
BEC2:85 35                    STA   DEVNUM           ; CURRENT FILE'S DEVICE
BEC4:A9 00                    LDA   #0               ; LOOK AT ALL VOLUMES LOGGED IN
BEC6:AA            FNDFVOL    TAX
BEC7:BD 10 11                 LDA   VCB+VCBDEV,X     ; GET VOLUMES DEVICE NUMBER
BECA:D1 BA                    CMP   (FCBPTR),Y       ; HVE WE FOUND A MATCH.
BECC:D0 20   BEEE             BNE   FNDFV1
BECE:A0 1A                    LDY   #FCBSWAP         ; SWAP BYTES
BED0:BD 1F 11                 LDA   VCB+VCBSWAP,X    ; MISMATCH
BED3:D1 BA                    CMP   (FCBPTR),Y       ; MEANS FILE BELONGS
BED5:D0 15   BEEC             BNE   FNDFV.1          ; TO ANOTHER VOLUME
BED7:BD 00 11                 LDA   VCB,X            ; IS THIS AN OPEN DEVICE?
BEDA:F0 10   BEEC             BEQ   FNDFV.1          ; NO, TRY ANOTHER VOLUME
BEDC:20 05 BF                 JSR   FVOLFOUND        ; YES, SAVE VCB ADDRESS
BEDF:BD 1F 11                 LDA   VCB+VCBSWAP,X    ; SWAPPED?
BEE2:F0 1C   BF00             BEQ   REEFER1          ; NO, RETURN CALMLY TO USER
BEE4:20 51 DC                 JSR   SWAPIN           ; YES, SWAP ME IN
```

```
BEE7:90 17   BF00              BCC   REEFER1          ; RETURN WITHOUT ERROR
BEE9:A9 00                     LDA   #XIOERROR        ; USER REFUSED TO MOUNT PROPER VOLUME
BEEB:60                        RTS
BEEC:              *
BEEC:A0 01         FNDFV.1     LDY   #FCBDEVN         ; RELOAD Y WITH DEVICE INDEX
BEEE:8A            FNDFV1      TXA
BEEF:18                        CLC
BEF0:69 20                     ADC   #VCBSIZE
BEF2:90 D2   BEC6              BCC   FNDFVOL          ; LOOP UNTIL FOUND
BEF4:A9 00                     LDA   #VCBERR          ; OTHERWISE DIE A SYSTEM DEATH!
BEF6:20 00 00                  JSR   SYSDEATH
```

```
BEF9:              *
BEF9:A9 00         ERRNOREF   LDA   #0               ; DROP A ZERO INTO THIS FCB TO
BEFB:91 BA                    STA   (FCBPTR),Y       ; SHOW FREE FCB
BEFD:              *
BEFD:A9 00         REEFER     LDA   #BADREFNUM       ; TELL USER THAT REQUESTED REFNUM
BEFF:38                       SEC                    ; IS ILLEGAL (OUT OF RANGE) FOR THIS CALL.
BF00:60            REEFER1    RTS
BF01:              *
BF01:A9 00         ERRNOTBLK  LDA   #NOTBLKDEV       ; TELL USER THAT SPECIFIED DEVICE IS NOT A BLOCK DEVICE
BF03:38                       SEC
BF04:60                       RTS
BF05:              *
BF05:        BF05  SVCBADR    EQU   *
BF05:86 B6         FVOLFOUND  STX   VCBPTR
BF07:A9 11                    LDA   #VCB/256
BF09:85 B7                    STA   VCBPTR+1
BF0B:18                       CLC                    ; INDICATE LEGAL REFNUM
BF0C:60                       RTS
```

```
BF0D:              * NAME    : GETDNUM
BF0D:              * FUNCTION: GET DEVICE NUMBER
BF0D:              * INPUT   : DVNAMP SETUP
BF0D:              * OUTPUT  : DEVNUM IN 'SCRTCH'
BF0D:              *         : 'BPL' IF NOT BLOCK DEV
BF0D:              *         : 'BCS' IF NO DEVICE
BF0D:              * VOLATILE: ALL REGS
BF0D:              *
BF0D:        BF0D  GETDNUM    EQU   *
BF0D:A9 E4                    LDA   #>SCRTCH+1       ; SET UP POINTER TO SCRATCH AREA
BF0F:85 C3                    STA   DVDNUM           ; TO RECIEVE DEVICE NUMBER.
BF11:A9 DB                    LDA   #SCRHIGH
BF13:85 C4                    STA   DVDNUM+1
BF15:A9 00                    LDA   #0               ; PLACE A ZERO IN BANK BYTE SINCE
BF17:8D C4 14                 STA   SISTER+DVDNUM+1  ; IT'S NOT IN A BANK.
BF1A:85 B7                    STA   VCBPTR+1
BF1C:A9 04                    LDA   #4               ; THE 'GET.DNUM' COMMAND.
BF1E:85 C0                    STA   DHPCMD
BF20:20 3E CF                 JSR   RPEATIO0         ; CALL BOB FOR THE INFO.
BF23:60                       RTS                    ; RETURN WITH DEVMGR CC'S
```

```
BF24:              *
BF24:              * NAME    : SRCHDEV
BF24:              * FUNCTION: SEARCH FOR A VOLUME
BF24:              *
BF24:        BF24  SRCHDEV    EQU   *
BF24:20 0D BF                 JSR   GETDNUM          ; GET DEVNUM
BF27:B0 54   BF7D             BCS   VOLERR1          ; BRANCH IF ANY ERROR OTHER THAN NOTBLOCKDEV
BF29:10 D6   BF01             BPL   ERRNOTBLK        ; BRANCH IF NOT A BLOCK DEVICE
BF2B:A9 00                    LDA   #0               ; NOW SEARCH FOR A VOL WITH THE
BF2D:8D 78 BF                 STA   NFOPEN           ; INIT TEMP VCB POINTER
BF30:AA            VOLOOK     TAX                    ; SAME DEVNUM AS SCRTCH
BF31:BD 11 11                 LDA   VCB+VCBSTAT,X    ; ANY FILES OPEN?
BF34:D0 03   BF39             BNE   VLOOK00          ; BRANCH IF SOME FILE OPEN
BF36:8E 78 BF                 STX   NFOPEN           ; ELSE SAVE THE VCB ENTRY PTR
BF39:        BF39  VLOOK00    EQU   *
BF39:BD 1F 11                 LDA   VCB+VCBSWAP,X    ; VOLUME SWAPPED OUT?
BF3C:D0 08   BF46             BNE   VNOTEQ           ; YES, CANT BE THE ACTIVE VOL
BF3E:BD 10 11                 LDA   VCB+VCBDEV,X
BF41:4D E4 DB                 EOR   SCRTCH+1
BF44:F0 05   BF4B             BEQ   VLOOK0           ; BRANCH IF MATCH.
BF46:BD 00 11      VNOTEQ     LDA   VCB,X            ; IS THIS A FREE VCB?
BF49:D0 48   BF93             BNE   VLOOK2           ; BRANCH IF NOT FREE, OTHEWISE TAKE NEXT BRANCH.
BF4B:5D 00 11      VLOOK0     EOR   VCB,X            ; TEST FOR A VOLUME NAME LENGTH
BF4E:F0 40   BF90             BEQ   VLOOK1           ; BRANCH IF VCB FREE
BF50:20 05 BF                 JSR   SVCBADR          ; SAVE CURRENT ADDRESS OF VCB.
BF53:BD 11 11                 LDA   VCB+VCBSTAT,X    ; TEST FOR ANY OPEN FILES.
BF56:10 4B   BFA3             BPL   VLOOK3           ; LOG THE VOLUME IN JUST TO BE SURE
BF58:AD E4 DB                 LDA   SCRTCH+1         ; SET UP
BF5B:85 35                    STA   DEVNUM           ; DEVICE NUMBER ARGUMENT
BF5D:8A                       TXA                    ; SAVE PTR TO VCB
BF5E:48                       PHA                    ; ON STACK
BF5F:20 0A C9                 JSR   VERFYVOL         ; COMPARES VCBPTR TO DEVNUM CONTENTS
BF62:90 15   BF79             BCC   VNOSWIT
BF64:C9 00                    CMP   #VNFERR          ; SEE IF NOTHING IN DRIVE
BF66:F0 24   BF8C             BEQ   VLOOK7           ; BRANCH IF NOTHING IN DRIVE
BF68:20 65 C4                 JSR   TSTSOS           ; IS THE VOLUME AN UNRECOGNIZED SOS OR (UCSD OR DOS)?
BF6B:B0 1B   BF88             BCS   KNOTSOS          ; DEFINITELY NOT SOS FORMAT
BF6D:A2 00                    LDX   #0               ; START VCB SCAN AT BEGINNING
BF6F:20 02 C8                 JSR   SNSWIT1          ; FIND A FREE VCB AND LOG IN THE NEW GUY
BF72:B0 0B   BF7F             BCS   VNOSWIT1         ; CAN'T LOG IN NEW GUY--KEEP OLD
BF74:68                       PLA
BF75:A6 B6                    LDX   VCBPTR           ; PASS BACK X AS NEW VCB
BF77:60                       RTS
BF78:              *
BF78:        0001  NFOPEN     DS    1                ; TEMP VCB PTR FOR VCB W/ NO FILES OPEN
BF79:              *
BF79:18            VNOSWIT    CLC                    ; RETURN IT TO USER
BF7A:68                       PLA                    ; REMEMBER OLD VCB PTR
BF7B:AA                       TAX                    ; AND PASS BACK TO USER
BF7C:60                       RTS
BF7D:              ; RETURN TO CALLER X=POINTER TO VCB.
BF7D:              *
BF7D:38            VOLERR1    SEC                    ; RETURN SOME VOLUME ERROR
BF7E:60                       RTS
BF7F:C9 00         VNOSWIT1   CMP   #DUPVOL
BF81:D0 09   BF8C             BNE   VLOOK7           ; REPORT OTHER ERROR FROM LOGGING IN NEW VOL AS VNF
```

```
BF83:AA                       TAX
BF84:68                       PLA                    ; MAKE STACK CORRECT
BF85:8A                       TXA                    ; RESTORE ERROR CODE
BF86:38                       SEC
BF87:60                       RTS                    ; IF DUPLICATE VOLUME ERROR, RETURN FACT TO USER
BF88:68            KNOTSOS    PLA                    ; MAKE STACK CORRECT
BF89:A9 00                    LDA   #NOTSOS          ; FOR THE PASCAL FOLK
BF8B:60                       RTS                    ; NOTSOS MEANS UCSD OR DOS OR BAD SOS VOLUME
BF8C:              *
BF8C:68            VLOOK7     PLA                    ; THROW AWAY OLD VCB PTR
BF8D:4C D2 BF                 JMP   NOVOLM           ; AND REPORT VOLUME NOT FOUND
BF90:              *
BF90:20 05 BF      VLOOK1     JSR   SVCBADR          ; SAVE ADDRESS OF FREE VCB.
BF93:8A            VLOOK2     TXA                    ; BUMP TO NEXT VOLUME ENTRY.
BF94:18                       CLC
BF95:69 20                    ADC   #VCBSIZE
BF97:90 97   BF30             BCC   VOLOOK           ; BRANCH IF MORE TO CHECK.
BF99:A6 B7                    LDX   VCBPTR+1         ; FREE VCB YET FOUND?
BF9B:D0 06   BFA3             BNE   VLOOK3           ; BRANCH IF YES
BF9D:AE 78 BF                 LDX   NFOPEN           ; SAVE POSSIBLE FREE VCB
BFA0:20 05 BF                 JSR   SVCBADR          ; AND SAVE PTR PERMANENTLY
BFA3:A5 B7         VLOOK3     LDA   VCBPTR+1         ; WAS A FREE VCB FOUND?
BFA5:F0 2B   BFD2             BEQ   NOVOLM           ; BRANCH IF VOLUME CAN'T BE LOGGED IN.
BFA7:AD E4 DB                 LDA   SCRTCH+1         ; GET DEVICE NUMBER
BFAA:85 35                    STA   DEVNUM           ; SAVE DEVICE NUMBER.
BFAC:A9 01                    LDA   #1               ; FAKE OUT 'LOKVOL'
BFAE:8D E3 DB                 STA   SCRTCH           ; TO THINK TO LOOK ONLY ONCE.
BFB1:85 36                    STA   TOTDEVS
BFB3:A9 11                    LDA   #<VCB
BFB5:85 B7                    STA   VCBPTR+1
BFB7:85 B1                    STA   PATHNMH          ; (TO MAKE HARMLESS)
BFB9:A9 00                    LDA   #0
BFBB:8D B1 14                 STA   SISTER+PATHNMH
BFBE:A6 B6                    LDX   VCBPTR
BFC0:86 B0                    STX   PATHNML
BFC2:9D 00 11                 STA   VCB,X            ; FORCE CURRENT VOLUME OFF LINE, THEN LOG WHATS THERE.
BFC5:20 AA C7                 JSR   FREEVCB          ; GO READ ROOT DIRECTORY.
BFC8:B0 0B   BFD5             BCS   RTVOLNAM         ; RETURN ANY ERRORS
BFCA:A6 B6                    LDX   VCBPTR           ; MAKE SURE VOLUME WAS LOGGED IN
BFCC:BD 00 11                 LDA   VCB,X
BFCF:F0 01   BFD2             BEQ   NOVOLM           ; RETURN ERROR
BFD1:60                       RTS                    ; ELSE RETURN NORMALLY
BFD2:A9 00         NOVOLM     LDA   #VNFERR          ; TELL USER 'NO VOLUME'
BFD4:38                       SEC
BFD5:AA            RTVOLNAM   TAX                    ; SAVE REAL ERROR WHILE DUPLICATE IS CHECKED
BFD6:A5 3C                    LDA   DUPLFLAG
BFD8:F0 02   BFDC             BEQ   RTV1             ; BRANCH IF NOT DUPLICATE
BFDA:A2 00                    LDX   #DUPVOL
BFDC:8A            RTV1       TXA                    ; RECALL ERROR
BFDD:60                       RTS
BFDE:              *
BFDE:                         CHN   VOLUME
```

```
BFDE:              ****************************************
BFDE:              * NAME    : VOLUME
BFDE:              * FUNCTION: RETURN VOLUME INFO
BFDE:              * INPUT   : DEVICE NAME
BFDE:              * OUTPUT  : THE INFO
BFDE:              * VOLATILE: ALL REGS
BFDE:              ****************************************
BFDE:              *
BFDE:        BFDE  VOLUME     EQU   *
BFDE:A5 A1                    LDA   C.DNAMP          ; TRANSFER DEVICE NAME
BFE0:85 C1                    STA   DVNAMP           ; NAME FOR DMGR
BFE2:A5 A2                    LDA   C.DNAMP+1
BFE4:85 C2                    STA   DVNAMP+1
BFE6:AD A2 14                 LDA   SISTER+C.DNAMP+1 ; AND XTND
BFE9:8D C2 14                 STA   SISTER+DVNAMP+1
BFEC:20 0D BF                 JSR   GETDNUM          ; GET DEVNUM
BFEF:90 01   BFF2             BCC   VOL7             ; =>SOME KINDA ERROR
BFF1:60                       RTS                    ; RETURN ERROR
BFF2:30 05   BFF9  VOL7       BMI   VOL2             ; =>IT'S GOOD...
BFF4:A9 00                    LDA   #NOTBLKDEV       ; NOT BLOCKED
BFF6:4C CC C0                 JMP   VOLERR           ; =>RETURN THE ERROR
BFF9:              *
BFF9:              * UNCONDITIONALLY READ ROOT DIRECTORY:
BFF9:              *
BFF9:        BFF9  VOL2       EQU   *
BFF9:AD E4 DB                 LDA   SCRTCH+1
BFFC:85 35                    STA   DEVNUM           ; SETUP DEV NUMBER
BFFE:A9 02                    LDA   #2               ; BLKNUM=2
C000:A2 00                    LDX   #0
C002:20 1E C9                 JSR   GETROT0          ; GET IT PLEASE
C005:A9 00                    LDA   #VNFERR          ; ERROR CODE
C007:90 01   C00A             BCC   VOL8             ; BRANCH IF NO ERROR ON READ
C009:60                       RTS                    ; =>ERROR, PASS IT ON.
C00A:              *
C00A:A9 00         VOL8       LDA   #>VCB            ; SET VCBPTR TO THE
C00C:85 B6                    STA   VCBPTR           ; FIRST OF THEM
C00E:A9 11                    LDA   #<VCB
C010:85 B7                    STA   VCBPTR+1
C012:              *
C012:              * IS THIS VOLUME SOS OR OTHER?
C012:              *
C012:20 65 C4                 JSR   TSTSOS           ; WHICH KIND?
C015:90 03   C01A             BCC   VLOGGED          ; =>IT'S SOS
C017:4C CE C0                 JMP   VNOTSOS          ; =>NOT SOS
C01A:              *
C01A:              * IS THIS SOS VOLUME LOGGED IN?
C01A:              *
C01A:        C01A  VLOGGED    EQU   *
C01A:20 F2 C8                 JSR   CMPVCB           ; DOES VOLNAME MATCH?
C01D:90 07   C026             BCC   VFOUND           ; =>YES, WE KNOW ABOUT IT.
C01F:20 E9 C0                 JSR   VNXTVCB          ; BUMP TO NEXT
C022:90 F6   C01A             BCC   VLOGGED          ; =>TRY 'EM ALL...
C024:B0 31   C057             BCS   VNEW             ; =>NOT FOUND, IT'S NEW (BRANCH ALWAYS)
C026:              *
C026:              *
C026:              * IT'S BEEN LOGGED IN BEFORE:
```

```
C026:               *  IS IT SWAPPED IN OR OUT?
C026:               *
C026:        C026   VFOUND     EQU   *
C026:A0 1F                     LDY   #VCBSWAP         ; INDEX TO IT
C028:B1 B6                     LDA   (VCBPTR),Y       ; SWAPPED?
C02A:10 13   C03F              BPL   VFOUND1          ; =>IN. RETURN THE INFO
C02C:               *
C02C:               * SWAPPED OUT. BEFORE WE SWAP IT
C02C:               *  IN, MAKE SURE IT BELONGS ON
C02C:               *  THIS DEVICE!
C02C:               *
C02C:A0 10                     LDY   #VCBDEV          ; INDEX TO IT
C02E:B1 B6                     LDA   (VCBPTR),Y       ; GET ITS DEVICE
C030:C5 35                     CMP   DEVNUM           ; CORRECT DEVICE?
C032:F0 05   C039              BEQ   VSWAPIN          ; =>YES
C034:A9 00                     LDA   #DUPVOL          ; IF FOR ANOTHER DEV,
C036:4C CC C0                  JMP   VOLERR           ; THEN IT'S AN ERROR!
C039:               *
C039:               * NOW SWAP-IN THIS VOLUME:
C039:               *
C039:        C039   VSWAPIN    EQU   *
C039:20 51 DC                  JSR   SWAPIN           ; SWAP IT IN
C03C:4C 9E C0                  JMP   VINFO            ; AND RETURN THE INFO
C03F:               *
C03F:A0 10          VFOUND1    LDY   #VCBDEV
C041:B1 B6                     LDA   (VCBPTR),Y       ; SAME DEVICES?
C043:C5 35                     CMP   DEVNUM
C045:F0 57   C09E              BEQ   VINFO            ; YES; RETURN THE INFORMATION
C047:A0 11                     LDY   #VCBSTAT
C049:B1 B6                     LDA   (VCBPTR),Y       ; OPEN FILES?
C04B:10 04   C051              BPL   VFOUND2          ; BRANCH IF NOT
C04D:A9 00                     LDA   #DUPVOL
C04F:D0 7B   C0CC              BNE   VOLERR           ; ELSE REPORT DUPLICATE VOLUME ERROR (BRANCH ALWAYS)
C051:A0 00          VFOUND2    LDY   #VCBNML          ; MOVE THE LOGIN TO THIS NEW DEVICE
C053:A9 00                     LDA   #0               ; BY UNLOGGING THE OLD
C055:91 B6                     STA   (VCBPTR),Y       ; AND LOGGING IN THE NEW (DROP INTO VNEW)
C057:               ****************************************
C057:               *
C057:               * IT'S A BRAND NEW VOLUME.
C057:               *  GUESS WE'LL HAVE TO LOG IT IN:
C057:               *
C057:        C057   VNEW       EQU   *
C057:A5 35                     LDA   DEVNUM           ; PASS A REG TO SWAPOUT
C059:20 F6 DB                  JSR   SWAPOUT          ; SWAP ANY ACTIVE VOL ON THIS DEVICE
C05C:90 03   C061              BCC   VNEW1            ; BRANCH ON NO ERROR
C05E:A9 00                     LDA   #XIOERROR
C060:60                        RTS
C061:A9 00          VNEW1      LDA   #>VCB            ; FIND AN EMPTY VCB
C063:85 B6                     STA   VCBPTR
C065:A9 11                     LDA   #<VCB
C067:85 B7                     STA   VCBPTR+1
C069:A0 00          VFREE      LDY   #VCBNML
C06B:B1 B6                     LDA   (VCBPTR),Y       ; EMPTY VCB?
C06D:F0 2C   C09B              BEQ   VLOGIN           ; ITS FREE, USE IT
C06F:A0 10                     LDY   #VCBDEV
C071:B1 B6                     LDA   (VCBPTR),Y       ; OR ONE WITH SAME DEVICE
```

```
C073:C5 35                    CMP   DEVNUM
C075:D0 10   C087             BNE   VFREEX           ; BRANCH IF NO DEVICE MATCH
C077:A0 11                    LDY   #VCBSTAT
C079:B1 B6                    LDA   (VCBPTR),Y       ; AND NO OPEN FILES
C07B:10 1E   C09B             BPL   VLOGIN           ; BRANCH IF OK TO REUSE THIS VCB
C07D:A5 35                    LDA   DEVNUM           ; THEN WE MUST SWAP OUT THIS VOLUME
C07F:20 F6 DB                 JSR   SWAPOUT
C082:90 03   C087             BCC   VFREEX           ; SWAPOUT PROCEEDED OK
C084:A9 00                    LDA   #XIOERROR        ; ELSE REPORT ERROR
C086:60                       RTS
C087:20 E9 C0      VFREEX     JSR   VNXTVCB          ; TRY NEXT
C08A:90 DD   C069             BCC   VFREE            ; MORE TO COME
C08C:              * RAN OUT OF MT'S ... FIND W/O FILES
C08C:A0 11         VNFIL      LDY   #VCBSTAT
C08E:B1 B6                    LDA   (VCBPTR),Y
C090:10 09   C09B             BPL   VLOGIN
C092:20 E9 C0                 JSR   VNXTVCB
C095:90 F5   C08C             BCC   VNFIL
C097:              * ALL OPEN ... REPORT VCBFULL
C097:A9 00                    LDA   #FCBFULL
C099:D0 31   C0CC             BNE   VOLERR
C09B:        C09B  VLOGIN     EQU   *
C09B:20 8F C8                 JSR   LOGVCB           ; AND LOGIN THIS ONE
C09E:              ****************************************
C09E:              *
C09E:              * RETURN ALL THE NICE INFO:
C09E:              *
C09E:        C09E  VINFO      EQU   *
C09E:A9 00                    LDA   #0
C0A0:A0 14                    LDY   #VCBTFRE         ; FETCH VOLUME FREE BLOCK COUNT
C0A2:91 B6                    STA   (VCBPTR),Y       ; FORCE RESCAN OF ALL
C0A4:C8                       INY                    ; BITMAPS
C0A5:91 B6                    STA   (VCBPTR),Y       ; TO MAKE SURE VCB INFO CURRENT
C0A7:85 04                    STA   REQL             ; FREE BLOCKS
C0A9:85 05                    STA   REQH
C0AB:20 4C C9                 JSR   TSFRBLK
C0AE:              *
C0AE:A6 B6                    LDX   VCBPTR           ; GET VCB INDEX
C0B0:A0 00                    LDY   #0
C0B2:        C0B2  VINFO1     EQU   *
C0B2:BD 12 11                 LDA   VCB+VCBTBLK,X    ; MOVE TOTAL
C0B5:91 A5                    STA   (C.OUTBLK),Y     ; BLOCKS AVAIL
C0B7:E8                       INX
C0B8:C8                       INY
C0B9:C0 04                    CPY   #4               ; AND FREE ONES TOO
C0BB:D0 F5   C0B2             BNE   VINFO1
C0BD:              *
C0BD:A0 00                    LDY   #0               ; NOW DO VOLNAME
C0BF:B1 B6                    LDA   (VCBPTR),Y
C0C1:A8                       TAY
C0C2:        C0C2  VINFO2     EQU   *
C0C2:B1 B6                    LDA   (VCBPTR),Y
C0C4:91 A3                    STA   (C.OUTVOL),Y
C0C6:88                       DEY
C0C7:10 F9   C0C2             BPL   VINFO2
C0C9:18                       CLC
```

```
C0CA:90 01   C0CD              BCC   VOLRET           ; =>DONE
C0CC:              *
C0CC:        C0CC  VOLERR      EQU   *
C0CC:38                        SEC
C0CD:        C0CD  VOLRET      EQU   *
C0CD:60                        RTS
```

```
C0CE:              ****************************************
C0CE:              * THIS ISN'T A SOS VOLUME. MARK
C0CE:              *  THE ACTIVE VOL THIS DEVICE
C0CE:              *  SO THAT IT GETS CHECKED LATER:
C0CE:              *
C0CE:        C0CE  VNOTSOS    EQU   *
C0CE:A0 10                    LDY   #VCBDEV          ; IS VCB FOR THIS
C0D0:B1 B6                    LDA   (VCBPTR),Y       ; DEVICE?
C0D2:C5 35                    CMP   DEVNUM
C0D4:D0 0A   C0E0             BNE   VNS2
C0D6:A0 11                    LDY   #VCBSTAT         ; INDEX TO IT
C0D8:B1 B6                    LDA   (VCBPTR),Y       ; GET STATUS
C0DA:10 04   C0E0             BPL   VNS2             ; =>NOT ACTIVE.
C0DC:09 40                    ORA   #DSWITCH         ; SET 'SWITCHEROO'
C0DE:91 B6                    STA   (VCBPTR),Y       ; PUT IT BACK
C0E0:              *
C0E0:        C0E0  VNS2       EQU   *
C0E0:20 E9 C0                 JSR   VNXTVCB          ; GET NEXT VCB
C0E3:90 E9   C0CE             BCC   VNOTSOS          ; =>TRY 'EM ALL.
C0E5:              *
C0E5:A9 00                    LDA   #NOTSOS          ; GIVE THE ERROR
C0E7:D0 E3   C0CC             BNE   VOLERR           ; (BRANCH ALWAYS)
```

```
C0E9:              * NAME    : VNXTVCB
C0E9:              * FUNCTION: BUMP VCBPTR TO NEXT VCB
C0E9:              * INPUT   : NOTHING
C0E9:              * OUTPUT  : VCBPTR UPDATED
C0E9:              *         : 'BCC' IF MORE TO GO
C0E9:              *         : 'BCS' IF DONE
C0E9:              * VOLATILE: AC
C0E9:              *
C0E9:        C0E9  VNXTVCB    EQU   *
C0E9:A5 B6                    LDA   VCBPTR
C0EB:18                       CLC
C0EC:69 20                    ADC   #VCBSIZE         ; BUMP IT
C0EE:85 B6                    STA   VCBPTR
C0F0:60                       RTS                    ; CARRY SET IF END OF PAGE
C0F1:                         CHN   CREATE
```

```
C0F1:        C0F1      CREATE     EQU   *
C0F1:EE 17 C5                     INC   CFLAG            ; SAY WE ARE IN CREATE (DIR EXTEND)
C0F4:20 93 C4                     JSR   LOOKFILE         ; CHECK FOR DUPLICATE / GET FREE ENTRY
C0F7:B0 04   C0FD                 BCS   TSTFNF           ; ERROR CODE IN ACC MAY BE 'FILE NOT FOUND'
C0F9:A9 00                        LDA   #DUPERR          ; TELL EM A FILE OF THAT NAME ALREADY EXISTS
C0FB:38                CRERR1     SEC                    ; INDICATE ERROR ENCOUNTERED
C0FC:60                           RTS                    ; RETURN ERROR IN ACC.
C0FD:                  *
C0FD:C9 00             TSTFNF     CMP   #FNFERR          ; 'FILE NOT FOUND' IS WHAT WE WANT
C0FF:D0 FA   C0FB                 BNE   CRERR1           ; PASS BACK OTHER ERROR.
C101:A5 0C                        LDA   NOFREE           ; TEST FOR DIRECTORY SPACE
C103:D0 04   C109                 BNE   CREAT1           ; BRANCH IF VALID FREE ENTRY WAS FOUND.
C105:A9 00                        LDA   #DIRFULL         ; RETURN DIRECTORY FULL ERROR
C107:38                           SEC
C108:60                           RTS
C109:                  *
C109:A0 09             CREAT1     LDY   #$9              ; SET UP DEFAULT PARAMETERS FOR CREATE
C10B:A9 00                        LDA   #0               ; IN THE SPACE DIRECTLY FOLLOWING THE
C10D:99 A6 00          ZERCALL    STA   C.FILID,Y        ; CALL SPECFCATION AND THEN
C110:88                           DEY                    ; CHECK FOR ADDITIONAL PARAMETERS FROM
C111:10 FA   C10D                 BPL   ZERCALL          ; USER'S CALL SPEC VIA 'C.CLIST'
C113:A9 01                        LDA   #SEEDTYP         ; DEFAULT TYPE IS 'SEED' TREE INDEX
C115:85 A9                        STA   C.STOR
C117:A4 A5                        LDY   C.XLEN           ; GET THE LENGTH OF THE CALL XTENSION LIST
C119:F0 10   C12B                 BEQ   CRENAM           ; IF ZERO THEN USE DEFAULTS
C11B:88                           DEY                    ; (SINCE THE POINTER IS AT BYTE 0)
C11C:C0 09                        CPY   #$9              ; MAKE SURE WE DON'T HAVE TOO MANY PARAMETERS
C11E:90 03   C123                 BCC   MOVPARM          ; MOVE 'EM IF REASONABLE COUNT.
C120:A9 00                        LDA   #BADLSTCNT       ; INVALID LIST COUNT
C122:60                           RTS                    ; RETURN ERROR.
C123:                  *
C123:B1 A3             MOVPARM    LDA   (C.XLIST),Y      ; MOVE IN THE USER SPECIFIED
C125:99 A6 00                     STA   C.FILID,Y        ; PARAMETERS. VALIDITY IS CHECKED
C128:88                           DEY                    ; AT VARIOUS POINTS FURTHER ALONG IN
C129:10 F8   C123                 BPL   MOVPARM          ; THIS PROCESS.
C12B:A0 00             CRENAM     LDY   #0               ; MOVE LOCAL FILE NAME TO ENTRY BUFFER.
C12D:B1 B0                        LDA   (PATHNML),Y      ; GET LENGTH OF LOCAL NAME
C12F:A8                           TAY
C130:B1 B0             CRENAM1    LDA   (PATHNML),Y
C132:99 BA DB                     STA   DFIL+D.STOR,Y
C135:88                           DEY                    ; (MOVE ALL, INCLUDING LENGTH BYTE.)
C136:10 F8   C130                 BPL   CRENAM1
C138:A5 A6                        LDA   C.FILID          ; MOVE FILE AND AUX ID.
C13A:8D CA DB                     STA   DFIL+D.FILID
C13D:A5 A7                        LDA   C.AUXID
C13F:8D D9 DB                     STA   DFIL+D.AUXID
C142:A5 A8                        LDA   C.AUXID+1
C144:8D DA DB                     STA   DFIL+D.AUXID+1
C147:A9 C3                        LDA   #READEN+WRITEN+RENAMEN+DSTROYEN
C149:8D D8 DB                     STA   DFIL+D.ATTR
C14C:AD B5 DB                     LDA   D.HEAD           ; SAVE FILE'S HEADER ADDRESS TOO.
C14F:8D DF DB                     STA   DFIL+D.DHDR
C152:AD B6 DB                     LDA   D.HEAD+1
C155:8D E0 DB                     STA   DFIL+D.DHDR+1
C158:20 87 D5                     JSR   TWRPROT1         ; CAN WE WRITE TO THIS DISKETTE?
C15B:B0 9E   C0FB                 BCS   CRERR1
```

```
C15D:A5 A9          58            LDA   C.STOR           ; NOW TEST STORAGE TYPE FOR TREE TYPE FILES
C15F:C9 04          59            CMP   #4               ; NOTE: THIS IS HARD CODED SINCE ALL TREES ARE LESS THAN 4
***********
C161:90 03   C166   60            BCC   SEED             ; BRANCH IF SOME TYPE OF TREE (SEED, SAPLING...)
C163:4C E3 C2       61            JMP   NOTREE           ; GO TEST FOR SOME OTHER TYPE (SUCH AS DIRECTORY).
```

```
C166:                  *
C166:A2 01             SEED       LDX   #SEEDTYP         ; START OUT ASSUMING A SEED FILE
C168:A5 AD                        LDA   C.EOFHH          ; TEST FOR OUT OF RANGE PREALLOCATION
C16A:F0 04   C170                 BEQ   SEED1            ; (HOPEFULLY BRANCH ALWAYS)
C16C:A9 00             OVFLOW     LDA   #OVRERR          ; REPORT UNABLE TO SATISFY REQUEST.
C16E:38                           SEC                    ; INDICATE ERROR
C16F:60                           RTS
C170:                  *
C170:A5 AC             SEED1      LDA   C.EOFHL          ; CALCULATE THE NUMBER OF
C172:8D D1 DB                     STA   DFIL+D.EOF+2     ; BLOCKS NEEDED FOR PRE-ALLOCATION
C175:4A                           LSR   A
C176:A8                           TAY                    ; Y HOLDS THE NUMBER OF INDEX BLOCKS NEEDED
C177:85 01                        STA   DATBLKH
C179:A5 AB                        LDA   C.EOFLH          ; (CARRY UNDISTURBED FROM LAST SHIFT)
C17B:8D D0 DB                     STA   DFIL+D.EOF+1
C17E:6A                           ROR   A                ; WE NOW HAVE THE LOW ORDER COUNT OF NEEDED DATA BLOCKS
C17F:85 00                        STA   DATBLKL
C181:A5 AA                        LDA   C.EOFLL
C183:8D CF DB                     STA   DFIL+D.EOF       ; (CARRY IN TACT FROM LOW COUNT)
C186:D0 02   C18A                 BNE   INCDATA          ; BUMP THE COUNT ON DATA BLOCKS IF REQUEST
C188:90 07   C191                 BCC   TSTSAP           ; IS NOT A MULTIPLE OF 512.
C18A:E6 00             INCDATA    INC   DATBLKL
C18C:D0 03   C191                 BNE   TSTSAP
C18E:C8                           INY                    ; MUST INCREASE NUMBER OF INDEXES ALSO.
C18F:E6 01                        INC   DATBLKH
C191:98                TSTSAP     TYA                    ; IF NON ZERO, THEN IT'S AT LEAST A SAPLING.
C192:D0 10   C1A4                 BNE   SAPLING
C194:A5 00                        LDA   DATBLKL          ; TO QUALIFY AS AN HONEST SEED,
C196:D0 04   C19C                 BNE   TSTSEED          ; THEN ONE OR LESS DATA BLOCKS REQUESTED
C198:E6 00                        INC   DATBLKL          ; (MUST BE AT LEAST ONE BLOCK ALLOCATED
C19A:D0 14   C1B0                 BNE   CREALC           ; TYPE IS SEED. BRANCH ALWAYS
C19C:C9 01             TSTSEED    CMP   #1               ; IF GREATER THAN ONE, IT'S NOT A SEED.
C19E:F0 10   C1B0                 BEQ   CREALC           ; IT IS A SEED. CONTINUE CREATION
C1A0:E8                           INX                    ; THE TYPE IS SAPLING.
C1A1:C8                           INY                    ; ONE INDEX BLOCK IS NEEDED.
C1A2:D0 0C   C1B0                 BNE   CREALC           ; BRANCH ALWAYS
```

```
C1A4:              *
C1A4:E8            SAPLING    INX                    ; TYPE IS AT LEAST SAPLING.
C1A5:C9 01                    CMP   #1               ; NO MORE THAN ONE INDEX BLOCK FOR A SAPLING
C1A7:D0 04   C1AD             BNE   TREE
C1A9:A5 00                    LDA   DATBLKL          ; MUST BE SURE THIS IS REAL MAX SAPLING (128K FILE)
C1AB:F0 03   C1B0             BEQ   CREALC           ; BRANCH IF IT IS.
C1AD:C8            TREE       INY                    ; ACCOUNT FOR ADDITIONAL 2ND LEVEL INDEX
C1AE:              *
C1AE:E8                       INX                    ; TYPE IS TREE (2 LEVEL INDEX)
C1AF:C8                       INY                    ; ADD AN EXTRA INDEX BLOCK FOR TOP INDEX
C1B0:84 06         CREALC     STY   INDXBLK          ; STORE INDEX BLOCK COUNT
C1B2:8A                       TXA                    ; PUT STORAGE TYPE IN DIRECTORY ENTRY
C1B3:0A                       ASL   A
C1B4:0A                       ASL   A
C1B5:0A                       ASL   A
C1B6:0A                       ASL   A
C1B7:0D BA DB                 ORA   DFIL+D.STOR
C1BA:8D BA DB                 STA   DFIL+D.STOR
C1BD:86 07                    STX   LEVELS           ; SAVE NUMBER OF INDEX LEVELS FOR PREALLOCATION.
C1BF:98                       TYA                    ; NOW FIGURE THE TOTAL NUMBER OF
C1C0:18                       CLC                    ; BLOCKS NEEDED (DATA + INDEX BLOCKS)
C1C1:65 00                    ADC   DATBLKL
C1C3:8D CD DB                 STA   DFIL+D.USAGE     ; (MIGHT AS WELL RECORD IT IN DIR
C1C6:85 04                    STA   REQL             ; WHILE WE'RE AT IT.)
C1C8:A5 01                    LDA   DATBLKH
C1CA:69 00                    ADC   #0               ; UPDATE HI BYTE TOO
C1CC:8D CE DB                 STA   DFIL+D.USAGE+1
C1CF:85 05                    STA   REQH
C1D1:AE B4 DB                 LDX   D.DEV            ; PASS ALONG THE DEVICE WE'RE TALKIN ABOUT.
C1D4:20 4C C9                 JSR   TSFRBLK          ; 'TEST FREE BLOCKS' FINDS OUT IF ENOUGH FREE SPACE EXISTS
C1D7:B0 93   C16C             BCS   OVFLOW           ; BRANCH IF NOT ENOUGH SPACE.
C1D9:20 9C CA                 JSR   ALC1BLK          ; GO ALLOCATE FIRST BLOCK
C1DC:B0 57   C235             BCS   CRERR
C1DE:8D CB DB                 STA   DFIL+D.FRST      ; (RETURNS ACC=LOW Y=HIGH)
C1E1:85 02                    STA   IDXADRL          ; SAVE AS ADDRESS FOR INCORE INDEX ALSO.
C1E3:8C CC DB                 STY   DFIL+D.FRST+1
C1E6:84 03                    STY   IDXADRH
C1E8:20 C4 C2                 JSR   ZERGBUF          ; GO CLEAN OUT GBUF
C1EB:20 0A CB                 JSR   GTTINDX          ; GET TEMPORARY SPACE FOR AN INDEX BLOCK
C1EE:20 D1 C2                 JSR   ZTMPIDX          ; AND ZERO IT OUT.
C1F1:A6 07                    LDX   LEVELS
C1F3:CA                       DEX                    ; TEST FOR NUMBER OF LEVELS NEEDED.
C1F4:F0 4A   C240             BEQ   ENDCRE           ; BRANCH IF SEED FILE.
C1F6:CA                       DEX                    ; IS IT A SAPLING PRE-ALLOCATION.
C1F7:F0 3E   C237             BEQ   SAPFILE
C1F9:A4 06                    LDY   INDXBLK          ; LOAD NUMBER OF INDEX BLOCKS NEEDED
C1FB:88                       DEY                    ; REMOVE THE ONE JUST ALLOCATED.
C1FC:84 04                    STY   REQL
C1FE:84 06                    STY   INDXBLK
C200:20 6E CA                 JSR   ALCIDXS          ; GO ALLOCATE INDEXES FOR LOWER INDEX BLOCKS.
C203:B0 30   C235             BCS   CRERR
C205:20 8C CC                 JSR   WRTDFRST         ; GO WRITE TREE TOP INDEX BLOCK.
C208:B0 2B   C235             BCS   CRERR            ; BRANCH IF UNABLE TO DO THIS.
C20A:A9 00                    LDA   #0               ; INIT INDEX POINTER
C20C:85 0F                    STA   TREPTR
```

```
C20E:A4 0F          FILLTREE   LDY   TREPTR
C210:B1 B2                     LDA   (TINDX),Y       ; GET ADDRESS OF LOWER BLOCK
C212:85 02                     STA   IDXADRL
C214:E6 B3                     INC   TINDX+1         ; BUMP TO PAGE 2 TO GET HI ADDRESS.
C216:B1 B2                     LDA   (TINDX),Y       ; GET HIGH ADDRESS.
C218:85 03                     STA   IDXADRH
C21A:C6 B3                     DEC   TINDX+1         ; CLEAN UP AFTER SELF...
C21C:C6 06                     DEC   INDXBLK         ; IS THIS THE LAST BLOCK ALLOCATED?
C21E:F0 17   C237              BEQ   LSTSAP          ; YES, ALLOCATE PARTIAL FILLED INDEX BLOCK
C220:A9 00                     LDA   #0              ; ALLOCATE ALL 256 INDEXES
C222:85 04                     STA   REQL
C224:20 6D C2                  JSR   SAPINDX         ; AND WRITE ZEROED DATA BLOCKS.
C227:B0 0C   C235              BCS   CRERR           ; STOP IF ERROR ENCOUNTERED.
C229:20 78 CC                  JSR   WRTINDX         ; WRITE INDEX BLOCK
C22C:B0 07   C235              BCS   CRERR           ; HOPEFULLY NEVER TAKEN.
C22E:E6 0F                     INC   TREPTR
C230:20 90 CC                  JSR   RDFRST          ; READ IN TOP INDEX AGAIN.
C233:90 D9   C20E              BCC   FILLTREE        ; BRANCH IF NO ERROR.
C235:38             CRERR      SEC                   ; JUST IN CASE IT WAS CLEAR.
C236:60                        RTS                   ; RETURN ERROR.
C237:               *
C237:               *
C237:        C237   SAPFILE    EQU   *
C237:A5 00          LSTSAP     LDA   DATBLKL         ; GET NUMBER OF DATA BLOCKS (LOW BYTE) REQUESTED.
C239:85 04                     STA   REQL
C23B:20 6D C2                  JSR   SAPINDX         ; GO ALLOCATE DATA BLOCKS AND WRITE EM.
C23E:B0 F5   C235              BCS   CRERR
C240:20 78 CC       ENDCRE     JSR   WRTINDX         ; GO WRITE INDEX BLOCK. (FOR SEED THIS IS DATA.)
C243:B0 F0   C235              BCS   CRERR
C245:A2 03                     LDX   #3              ; MOVE CREATION TIME FOR THIS ENTRY
C247:B5 38          TRETIME    LDA   DATELO,X
C249:9D D2 DB                  STA   DFIL+D.CREDT,X
C24C:CA                        DEX
C24D:10 F8   C247              BPL   TRETIME
C24F:EE A9 DB       ENDCRE0    INC   H.FCNT          ; ADD ONE TO TOTAL NUMBER OF FILES IN SPECIFIED DIRECTORY.
C252:D0 0D   C261              BNE   ENDCRE1
C254:EE AA DB                  INC   H.FCNT+1
C257:A2 03                     LDX   #3              ; ENSURE MOD
C259:B5 38          ENDCRX     LDA   DATELO,X        ; DATE/TIME
C25B:9D DB DB                  STA   DFIL+D.MODDT,X  ; IS
C25E:CA                        DEX                   ; INITIALIZED
C25F:10 F8   C259              BPL   ENDCRX
C261:AE B4 DB       ENDCRE1    LDX   D.DEV           ; UPDATE APPROPRIATE BIT MAP
C264:20 E4 CB                  JSR   UPBMAP
C267:B0 79   C2E2              BCS   CRERR2          ; BRANCH ON BITMAP UPDATE ERR
C269:20 F0 C3                  JSR   DREVISE         ; UPDATE DIRECTORY LAST
C26C:60                        RTS                   ; RETURN ERRORS OR OK RESULT
C26D:               *
```

```
C26D:20 D1 C2      SAPINDX    JSR   ZTMPIDX          ; ZERO OUT ANY STUFF LEFT OVER.
C270:A5 04                    LDA   REQL             ; PRESERVE REQUEST COUNT
C272:85 10                    STA   TLINK
C274:20 6E CA                 JSR   ALCIDXS          ; GO ALLOCATE REQUESTED NUMBER OF BLOCKS.
C277:B0 BC   C235             BCS   CRERR
C279:A0 00                    LDY   #0               ; THEN WRITE ZEROS TO DATA BLOCKS.
C27B:84 0E                    STY   SAPTR            ; USE AS POINTER TO INDEX BLOCK
C27D:B1 B2                    LDA   (TINDX),Y        ; GET DATA BLOCK ADDRESS (LOW BYTE).
C27F:85 C6                    STA   BLOKNML
C281:E6 B3                    INC   TINDX+1
C283:B1 B2                    LDA   (TINDX),Y        ; GET HIGH ADRRESS OF PRE-ALLOCATED DATA BLOCK.
C285:85 C7                    STA   BLOKNMH
C287:C6 B3                    DEC   TINDX+1          ; (RESET BUFFER ADDRESS)
C289:20 54 CC                 JSR   WRTGBUF          ; WRITE DATA BLOCK
C28C:B0 A7   C235             BCS   CRERR
C28E:A5 10                    LDA   TLINK            ; GET NUMBER REQUESTED AGAIN
C290:85 04                    STA   REQL
C292:A4 0E         DATINIT    LDY   SAPTR            ; GET POINTER TO INDEX BLOCK AGAIN.
C294:C8                       INY                    ; ANTICIPATE DOIN' THE NEXT DATA BLOCK
C295:C6 04                    DEC   REQL             ; DO WE INDEED HAVE ANOTHER BLOCK TO WRITE.
C297:F0 23   C2BC             BEQ   DATDONE          ; NO, ALL DONE (CARRY CLEAR).
C299:84 0E                    STY   SAPTR            ; USE AS POINTER TO INDEX BLOCK
C29B:B1 B2                    LDA   (TINDX),Y        ; GET DATA BLOCK ADDRESS (LOW BYTE).
C29D:85 C6                    STA   BLOKNML
C29F:E6 B3                    INC   TINDX+1          ; BUMP HI ADDR OF INDEX BUFFER TO ACCESS HIGH ADDR.
C2A1:AA                       TAX                    ; WAS LOW ADDRESS A ZERO?
C2A2:D0 09   C2AD             BNE   DATIT1           ; IF NOT, NO NEED TO CHECK VALIDITH OF HI BYTE
C2A4:D1 B2                    CMP   (TINDX),Y
C2A6:D0 05   C2AD             BNE   DATIT1           ; BOTH BYTES CAN'T BE ZERO.
C2A8:A9 00                    LDA   #ALCERR
C2AA:20 00 00                 JSR   SYSDEATH
C2AD:B1 B2         DATIT1     LDA   (TINDX),Y        ; GET HIGH ADRRESS OF PRE-ALLOCATED DATA BLOCK.
C2AF:85 C7                    STA   BLOKNMH
C2B1:C6 B3                    DEC   TINDX+1          ; (RESET BUFFER ADDRESS)
C2B3:A9 12                    LDA   #GBUF/256
C2B5:85 C3                    STA   DBUFPH           ; RESET TO ADDR TO GBUF JUST TO BE SURE.
C2B7:20 BD C2                 JSR   REPEATIO         ; WRITE DATA BLOCK
C2BA:90 D6   C292             BCC   DATINIT
C2BC:60            DATDONE    RTS                    ; RETURN STATUS (CARRY SET IF ERROR)
C2BD:              *
C2BD:        C2BD  REPEATIO   EQU   *
C2BD:A9 09                    LDA   #RPTCMD
C2BF:85 C0                    STA   DHPCMD
C2C1:4C 3A CF                 JMP   RPEATIO1
C2C4:              *
C2C4:A0 00         ZERGBUF    LDY   #0               ; ZERO OUT THE GENERAL PURPOSE BUFFER
C2C6:98                       TYA
C2C7:99 00 12      ZGBUF      STA   GBUF,Y           ; WIPE OUT BOTH PAGES
C2CA:99 00 13                 STA   GBUF+$100,Y      ; WITH SAME LOOP.
C2CD:C8                       INY
C2CE:D0 F7   C2C7             BNE   ZGBUF
C2D0:60                       RTS
C2D1:              *
C2D1:              *
C2D1:A0 00         ZTMPIDX    LDY   #0               ; ZERO OUT TEMPORARY INDEX BLOCK
C2D3:98                       TYA
```

```
C2D4:91 B2         261 ZINDX1     STA   (TINDX),Y        ; THIS HAS TO BE DONE A
C2D6:C8            262            INY                    ; TIME SINCE IT'S INDIRECT.
C2D7:D0 FB   C2D4  263            BNE   ZINDX1
C2D9:E6 B3         264            INC   TINDX+1
C2DB:91 B2         265 ZINDX2     STA   (TINDX),Y
C2DD:C8            266            INY
C2DE:D0 FB   C2DB  267            BNE   ZINDX2
C2E0:C6 B3         268            DEC   TINDX+1          ; RESTORE PROPER ADDRESS
C2E2:60            269 CRERR2     RTS
```

```
C2E3:C9 0D         NOTREE     CMP   #DIRTYP          ; IS A DIRECTORY TO BE CREATED?
C2E5:F0 03   C2EA             BEQ   ISDIR            ; YES, DO SO...
C2E7:4C 61 C4                 JMP   NOTDIR           ; NO, TRY NEXT TYPE.
C2EA:              *
C2EA:A5 AD         ISDIR      LDA   C.EOFHH          ; CAN'T CREATE A DIRECTORY LARGER THAN
C2EC:05 AC                    ORA   C.EOFHL          ; 127 BLOCKS (THAT'S HUGE!)
C2EE:F0 04   C2F4             BEQ   ISDIR1           ; BRANCH IF WITHIN LIMITS, OTHEWISE
C2F0:A9 00         DIROVR     LDA   #OVRERR          ; REQUESTED DIRECTORY SIZE CAN'T BE
C2F2:38                       SEC                    ; CREATED. SET CARRY TO INDICATE ERROR.
C2F3:60                       RTS
C2F4:              *
C2F4:A5 AB         ISDIR1     LDA   C.EOFLH          ; CALCULATE HOW MANY BLOCKS WILL
C2F6:4A                       LSR   A                ; BE NEEDED FOR THIS NEW DIRECTORY.
C2F7:A8                       TAY                    ; (SAVE INITIAL COUNT IN Y)
C2F8:A5 AA                    LDA   C.EOFLL          ; IF REQUESTED EOF IS NOT AN EVEN BLOCK
C2FA:D0 02   C2FE             BNE   DADD1            ; SIZE, THEN ROUND UP.
C2FC:90 01   C2FF             BCC   TSDIRSZ          ; BRANCH IF ROUNING UNNECESSARY.
C2FE:C8            DADD1      INY                    ; ADD ONE TO BLOCK COUNT.
C2FF:98            TSDIRSZ    TYA                    ; TEST TO BE SURE SIZE IS GREATER THAN ZERO
C300:F0 FC   C2FE             BEQ   DADD1            ; IF ZERO THEN SIZE=1
C302:8D CD DB                 STA   DFIL+D.USAGE     ; SAVE NUMBER OF BLOCKS TO BE USED.
C305:85 04                    STA   REQL
C307:0A                       ASL   A                ; NOW SAVE ADJUSTED END OF FILE
C308:8D D0 DB                 STA   DFIL+D.EOF+1
C30B:A9 00                    LDA   #0
C30D:8D CF DB                 STA   DFIL+D.EOF
C310:8D D1 DB                 STA   DFIL+D.EOF+2
C313:85 05                    STA   REQH             ; REQUESTED NUMBER OF BLOCKS NEVER EXCEEDS 128.
C315:20 4C C9                 JSR   TSFRBLK          ; TEST TO BE SURE ENOUGH DISK SPACE IS FREE.
C318:B0 D6   C2F0             BCS   DIROVR           ; BRANCH IF REQUEST TOO LARGE.
C31A:20 C4 C2                 JSR   ZERGBUF          ; CLEAR CRAP FROM GBUF.
C31D:20 9C CA                 JSR   ALC1BLK          ; GET ADDRESS OF FIRST (HEADER) BLOCK.
C320:B0 C0   C2E2             BCS   CRERR2
C322:8D CB DB                 STA   DFIL+D.FRST
C325:85 10                    STA   TLINK
C327:8C CC DB                 STY   DFIL+D.FRST+1
C32A:84 11                    STY   TLINK+1          ; (TLINK IS FOR REVERSE LINKAGE.)
C32C:AD CF C3                 LDA   SOSTMPL          ; STORE SOS STAMP IN NEW DIRECTORY
C32F:8D 00 12                 STA   GBUF
C332:AD D0 C3                 LDA   SOSTMPH
C335:8D 01 12                 STA   GBUF+1
C338:A0 04                    LDY   #4               ; MOVE OTHER VARIOUS THINGS
C33A:D0 06   C342             BNE   DRSTUF1          ; BRANCH ALWAYS
C33C:B9 B7 DB      DRSTUF     LDA   D.ENTBLK,Y       ; MOVE OWNING ENTRY'S
C33F:99 27 12                 STA   GBUF+HRBLK+4,Y   ; BLOCK ADDRESSES AND NUMBER TO NEW HEADER.
C342:B9 D1 C3      DRSTUF1    LDA   SOSVER,Y         ; MOVE VERSION, COMPATABLITY,
C345:99 20 12                 STA   GBUF+HVER+4,Y    ; ATTRIBUTES, AND ENTRY SIZE
C348:88                       DEY
C349:10 F1   C33C             BPL   DRSTUF
C34B:AD A7 DB                 LDA   H.ENTLN          ; OVER WRITE LAST BYTE MOVED IN ABOVE LOOP WITH
C34E:8D 2A 12                 STA   GBUF+HRELN+4     ; THE PARENT DIRECTORY ENTRY LENGTH.
C351:AD BA DB                 LDA   DFIL+D.STOR      ; SET HEADER TYPE AND NAME
C354:A8                       TAY
C355:09 E0                    ORA   #HEDTYP*16
C357:8D 04 12                 STA   GBUF+HNLEN+4
C35A:98                       TYA                    ; (AND WHILE WE'RE AT IT SET DIRECTORY TYPE)
```

```
C35B:09 D0                    ORA   #DIRTYP*16
C35D:8D BA DB                 STA   DFIL+D.STOR
C360:              *
C360:B9 BA DB      MVHNAME    LDA   DFIL+D.STOR,Y
C363:99 04 12                 STA   GBUF+HNLEN+4,Y   ; MOVE HEADER NAME
C366:88                       DEY
C367:D0 F7   C360             BNE   MVHNAME
C369:A2 03                    LDX   #3               ; GET CURRENT DATE.
C36B:B5 38         CRETIME    LDA   DATELO,X
C36D:9D 1C 12                 STA   GBUF+HCRDT+4,X   ; SAVE AS HEADER CREATION TIME
C370:9D D2 DB                 STA   DFIL+D.CREDT,X   ; AND DATE OF FILE CREATE.
C373:CA                       DEX
C374:10 F5   C36B             BPL   CRETIME
C376:A9 76                    LDA   #$76
C378:8D 14 12                 STA   GBUF+HPENAB+4    ; DUMMY PASSWORD
C37B:C6 04                    DEC   REQL             ; TEST FOR ONE BLOCK DIRECTORY
C37D:F0 2D   C3AC             BEQ   DIRCREND         ; IT IS, FINISH UP.
C37F:20 B4 C3                 JSR   DIRWRT           ; GO WRITE FIRST DIRECTORY BLOCK AND ALLOCATE NEXT
C382:B0 4A   C3CE             BCS   DERROR           ; PASS BACK ERROR.
C384:20 C4 C2                 JSR   ZERGBUF          ; CLEAN OUT GENERAL BUFFER AGAIN.
C387:A5 10         CRNXTDIR   LDA   TLINK            ; MOVE LAST BLOCK ADDRESS
C389:8D 00 12                 STA   GBUF             ; AS BACKWARD LINK.
C38C:A5 11                    LDA   TLINK+1
C38E:8D 01 12                 STA   GBUF+1
C391:A5 12                    LDA   FLINK            ; MAKE FORWARD LINK INTO CURRENT ADDRESS
C393:85 10                    STA   TLINK
C395:A5 13                    LDA   FLINK+1
C397:85 11                    STA   TLINK+1
C399:C6 04                    DEC   REQL             ; IS THIS THE LAST BLOCK?
C39B:F0 0F   C3AC             BEQ   DIRCREND
C39D:20 B4 C3                 JSR   DIRWRT           ; WRITE THIS BLOCK AND ALLOCATE NEXT.
C3A0:B0 2C   C3CE             BCS   DERROR
C3A2:A9 00                    LDA   #0               ; ZERO OUT FORWARD LINK
C3A4:8D 02 12                 STA   GBUF+2
C3A7:8D 03 12                 STA   GBUF+3
C3AA:F0 DB   C387             BEQ   CRNXTDIR         ; BRANCH ALWAYS
C3AC:              *
C3AC:20 C3 C3      DIRCREND   JSR   DIRWRT1          ; WRITE LAST BLOCK OF THIS DIRECTORY
C3AF:B0 1D   C3CE             BCS   DERROR
C3B1:4C 4F C2                 JMP   ENDCRE0          ; FINISH UP WRITING OWNER DIRECTORY STUFF.
C3B4:              *
C3B4:20 9C CA      DIRWRT     JSR   ALC1BLK          ; GET ADDRESS OF NEXT BLOCK.
C3B7:B0 15   C3CE             BCS   DERROR
C3B9:8D 02 12                 STA   GBUF+2
C3BC:8C 03 12                 STY   GBUF+3           ; SAVE LINK ADDRESS
C3BF:85 12                    STA   FLINK
C3C1:84 13                    STY   FLINK+1
C3C3:A5 10         DIRWRT1    LDA   TLINK            ; GET ADDRESS OF CURRENT BLOCK
C3C5:85 C6                    STA   BLOKNML
C3C7:A5 11                    LDA   TLINK+1
C3C9:85 C7                    STA   BLOKNMH
C3CB:4C 54 CC                 JMP   WRTGBUF          ; GO WRITE IT OUT
```

```
C3CE:              *
C3CE:        C3CE  ERRGBUF    EQU   *
C3CE:60            DERROR     RTS
C3CF:              *
C3CF:              *
C3CF:00            SOSTMPL    DFB   $0               ; THE FOLLOWING TWO BYTES ARE THE 'SOS STAMP'
C3D0:00            SOSTMPH    DFB   $0
C3D1:              *
C3D1:00 00 00 27   SOSVER     DFB   0,0,0,$27,13
C3D6:              *
C3D6:              *
C3D6:        C3D6  RNDTAB     EQU   *
C3D6:A9 12         ENTCALC    LDA   #GBUF/256        ; SET HIGH ADDRESS OF DIRECTORY ENTRY INDEX POINTER
C3D8:85 B5                    STA   DRBUFPH
C3DA:A9 04                    LDA   #4               ; CALCULATE ADDRESS OF ENTRY BASED
C3DC:AE B9 DB                 LDX   D.ENTNUM         ; ON THE ENTRY NUMBER
C3DF:18            ECALC0     CLC
C3E0:CA            ECALC1     DEX                    ; ADDR=GBUF+((ENTNUM-1)*ENTLEN)
C3E1:F0 09   C3EC             BEQ   ECALC2
C3E3:6D A7 DB                 ADC   H.ENTLN
C3E6:90 F8   C3E0             BCC   ECALC1
C3E8:E6 B5                    INC   DRBUFPH          ; BUMP HI ADDRESS
C3EA:B0 F3   C3DF             BCS   ECALC0           ; BRANCH ALWAYS.
C3EC:              *
C3EC:85 B4         ECALC2     STA   DRBUFPL          ; SAVE NEWLY CALCULATED LOW ADDRESS
C3EE:60                       RTS
```

```
C3EF:60            DERROR2    RTS
C3F0:              *
C3F0:A5 38         DREVISE    LDA   DATELO           ; IF NO CLOCK,
C3F2:F0 0A   C3FE             BEQ   DREVISE1         ; THEN DON'T TOUCH MOD T/D
C3F4:A2 03                    LDX   #3               ; MOVE LAST MODIFICATION DATE/TIME TO ENTRY BEING UPDATED.
C3F6:B5 38         MODTIME    LDA   DATELO,X
C3F8:9D DB DB                 STA   DFIL+D.MODDT,X
C3FB:CA                       DEX
C3FC:10 F8   C3F6             BPL   MODTIME
C3FE:              *
C3FE:AD D8 DB      DREVISE1   LDA   DFIL+D.ATTR      ; MARK ENTRY AS BACKUPABLE
C401:0D 57 D9                 ORA   BKBITFLG         ; BIT 5 = BACKUP NEEDED BIT
C404:8D D8 DB                 STA   DFIL+D.ATTR
C407:AD B4 DB                 LDA   D.DEV            ; GET DEVICE NUMBER OF DIRECTORY
C40A:85 35                    STA   DEVNUM           ; TO BE REVISED.
C40C:AD B7 DB                 LDA   D.ENTBLK         ; AND ADDRESS OF DIRECTORY BLOCK
C40F:85 C6                    STA   BLOKNML          ; THAT CONTAINS THE ENTRY.
C411:AD B8 DB                 LDA   D.ENTBLK+1
C414:85 C7                    STA   BLOKNMH
C416:20 58 CC                 JSR   RDGBUF           ; READ BLOCK INTO GENERAL PURPOSE BUFFER.
C419:B0 B3   C3CE             BCS   ERRGBUF
C41B:20 D6 C3                 JSR   ENTCALC          ; FIX UP POINTER TO ENTRY LOCATION WITHIN GBUF.
C41E:AC A7 DB                 LDY   H.ENTLN          ; NOW MOVE 'D.' STUFF TO DIRECTORY.
C421:88                       DEY
C422:B9 BA DB      MVDENT     LDA   DFIL+D.STOR,Y
C425:91 B4                    STA   (DRBUFPL),Y
C427:88                       DEY
C428:10 F8   C422             BPL   MVDENT
C42A:AD B5 DB                 LDA   D.HEAD           ; IS THE ENTRY BLOCK THE SAME AS THE
C42D:C5 C6                    CMP   BLOKNML          ; ENTRY'S HEADER BLOCK?
C42F:D0 07   C438             BNE   SVENTDIR         ; NO, SAVE ENTRY BLOCK
C431:AD B6 DB                 LDA   D.HEAD+1         ; MAYBE, TEST HIGH ADDRESSES
C434:C5 C7                    CMP   BLOKNMH
C436:F0 14   C44C             BEQ   UPHEAD           ; BRANCH IF THEY ARE THE SAME BLOCK.
C438:20 54 CC      SVENTDIR   JSR   WRTGBUF          ; WRITE UPDATED DIRECTORY BLOCK
C43B:B0 B2   C3EF             BCS   DERROR2          ; RETURN ANY ERROR.
C43D:AD B5 DB                 LDA   D.HEAD           ; GET ADDRESS OF HEADER BLOCK
C440:85 C6                    STA   BLOKNML
C442:AD B6 DB                 LDA   D.HEAD+1
C445:85 C7                    STA   BLOKNMH
C447:20 58 CC                 JSR   RDGBUF           ; READ IN HEADER BLOCK FOR MODIFICATION
C44A:B0 A3   C3EF             BCS   DERROR2
C44C:A0 01         UPHEAD     LDY   #1               ; UPDATE CURRENT NUMBER OF FILES IN THIS DIRECTORY
C44E:B9 A9 DB      UPHED1     LDA   H.FCNT,Y
C451:99 25 12                 STA   GBUF+HCENT+4,Y   ; (CURRENT ENTRY COUNT)
C454:88                       DEY
C455:10 F7   C44E             BPL   UPHED1
C457:AD A6 DB                 LDA   H.ATTR           ; ALSO UPDATE HEADER'S ATTRIBUTES.
C45A:8D 22 12                 STA   GBUF+HATTR+4
C45D:20 54 CC                 JSR   WRTGBUF          ; GO WRITE UPDATED HEADER
C460:60            DERROR1    RTS                    ; IMPLICITLY RETURN ANY ERRORS
C461:              *
```

```
C461:                  *
C461:A9 00             NOTDIR     LDA   #TYPERR          ; NOT TREE OR DIRECTORY- NOT A RECOGNIZED TYPE!
C463:38                TSTERR     SEC
C464:60                           RTS                    ; DO NOTHING.
C465:                  *
C465:                  *
C465:AD 00 12          TSTSOS     LDA   GBUF             ; TEST SOS STAMP
C468:CD CF C3                     CMP   SOSTMPL
C46B:D0 F6   C463                 BNE   TSTERR
C46D:AD 01 12                     LDA   GBUF+1
C470:CD D0 C3                     CMP   SOSTMPH
C473:D0 EE   C463                 BNE   TSTERR
C475:AD 04 12                     LDA   GBUF+4           ; TEST FOR HEADER
C478:29 E0                        AND   #$E0
C47A:C9 E0                        CMP   #HEDTYP*16
C47C:D0 E5   C463                 BNE   TSTERR           ; BRANCH IF NOT SOS HEADER (NO ERROR NUMBER)
C47E:18                           CLC                    ; INDICATE NO ERROR
C47F:60                           RTS
C480:                  *
C480:                             CHN   FNDFIL
```

```
C480:                *
C480:                *
C480:20 93 C4          FINDFILE   JSR   LOOKFILE         ; SEE IF FILE EXISTS
C483:B0 0D   C492                 BCS   NOFIND           ; BRANCH IF AN ERROR WAS ENCOUNTERED
C485:AC A7 DB          MOVENTRY   LDY   H.ENTLN          ; MOVE ENTIRE ENTRY INFO TO A SAFE AREA
C488:B1 B4             MOVENT1    LDA   (DRBUFPL),Y
C48A:99 BA DB                     STA   DFIL+D.STOR,Y
C48D:88                           DEY
C48E:10 F8   C488                 BPL   MOVENT1
C490:A9 00                        LDA   #0               ; TO INDICATE ALL IS WELL
C492:60                NOFIND     RTS                    ; RETURN CONDITION CODES.
```

```
C493:               *
C493:               *
C493:20 92 C6       LOOKFILE   JSR   PREPROOT         ; FIND VOLUME AND SET UP OTHER BORING STUFF
C496:B0 57   C4EF              BCS   FNDERR           ; PASS BACK ANY ERROR ENCOUNTERED
C498:A0 00                     LDY   #0               ; TEST TO SEE IF ONLY ROOT WAS SPECIFIED.
C49A:B1 B0                     LDA   (PATHNML),Y
C49C:D0 2F   C4CD              BNE   LOOKFIL0         ; BRANCH IF MORE THAN ROOT.
C49E:A9 12                     LDA   #GBUF/256        ;  OTHERWISE, REPORT A BADPATH ERROR
C4A0:85 B5                     STA   DRBUFPH          ; (BUT FIRST CREATE A PHANTOM ENTRY FOR OPEN)
C4A2:A9 04                     LDA   #4
C4A4:85 B4                     STA   DRBUFPL
C4A6:A0 1F                     LDY   #D.AUXID         ; FIRST MOVE IN ID, AND DATE STUFF.
C4A8:B1 B4          PHANTM1    LDA   (DRBUFPL),Y
C4AA:99 BA DB                  STA   DFIL,Y
C4AD:88                        DEY
C4AE:C0 17                     CPY   #D.CREDT-1
C4B0:D0 F6   C4A8              BNE   PHANTM1
C4B2:B9 B5 C4       PHANTM2    LDA   ROOTSTUF-D.FILID,Y
C4B5:99 BA DB                  STA   DFIL,Y
C4B8:88                        DEY
C4B9:C0 0F                     CPY   #D.FILID-1
C4BB:D0 F5   C4B2              BNE   PHANTM2
C4BD:A9 D0                     LDA   #DIRTYP*$10      ; FAKE DIRECTORY FILE
C4BF:8D BA DB                  STA   DFIL+D.STOR
C4C2:A9 00                     LDA   #BADPATH         ; (CARRY IS SET)
C4C4:60                        RTS
C4C5:               *
C4C5:00 02 00 04    ROOTSTUF   DFB   0,2,0,4
C4C9:00 00 08 00               DFB   0,0,8,0
C4CD:               *
C4CD:A9 00          LOOKFIL0   LDA   #0               ; RESET FREE ENTRY INDICATOR
C4CF:85 0C                     STA   NOFREE
C4D1:38                        SEC                    ; INDICATE THAT THE DIRECTORY TO BE SEARCHED HAS HEADER IN THIS
BLOCK
C4D2:A9 00          LOOKFIL1   LDA   #0               ; RESET ENTRY COUNTER
C4D4:85 08                     STA   TOTENT
C4D6:20 4D C6                  JSR   LOOKNAM          ; LOOK FOR NAME POINTED TO BY 'PATHNML'
C4D9:90 16   C4F1              BCC   NAMFOJMP         ; BRANCH IF NAME WAS FOUND.
C4DB:A5 09                     LDA   ENTCNTL          ; HAVE WE LOOKED AT ALL OF THE
C4DD:E5 08                     SBC   TOTENT           ;  ENTRIES IN THIS DIRECTORY?
C4DF:90 08   C4E9              BCC   DCRENTH          ; MAYBE, CHECK HI COUNT.
C4E1:D0 11   C4F4              BNE   LOOKFIL2         ; NO, READ NEXT DIRECTORY BLOCK
C4E3:C5 0A                     CMP   ENTCNTH          ; HAS THE LAST ENTRY BEEN LOOKED AT (ACC=0)
C4E5:F0 35   C51C              BEQ   ERRFNF           ; YES, GIVE 'FILE NOT FOUND' ERROR.
C4E7:D0 0B   C4F4              BNE   LOOKFIL2         ; BRANCH ALWAYS.
C4E9:C6 0A          DCRENTH    DEC   ENTCNTH          ; SHOULD BE AT LEAST 1
C4EB:10 07   C4F4              BPL   LOOKFIL2         ; (THIS SHOULD BE BRANCH ALWAYS...)
C4ED:A9 00          ERRDIR     LDA   #DIRERR          ; REPORT DIRECTORY MESSED UP.
C4EF:38             FNDERR     SEC                    ; INDICATE ERROR HAS BEEN ENCOUNTERED.
C4F0:60                        RTS
C4F1:4C D1 C5       NAMFOJMP   JMP   NAMFOUND         ; AVOID BRANCH OUT OF RANGE
C4F4:               *
```

```
C4F4:85 09          LOOKFIL2   STA   ENTCNTL          ; KEEP RUNNING COUNT
C4F6:A9 12                     LDA   #GBUF/256        ; RESET INDIRECT POINTER
C4F8:85 B5                     STA   DRBUFPH
C4FA:AD 02 12                  LDA   GBUF+2           ; GET LINK TO NEXT DIRECTORY BLOCK
C4FD:D0 05   C504              BNE   NXTDIR0          ; (IF THERE IS ONE)
C4FF:CD 03 12                  CMP   GBUF+3           ; ARE BOTH ZERO, I.E. NO LINK?
C502:F0 E9   C4ED              BEQ   ERRDIR           ; IF SO, THEN NOT ALL ENTRIES WERE ACCOUNTED FOR.
C504:85 C6          NXTDIR0    STA   BLOKNML
C506:AD 03 12                  LDA   GBUF+3
C509:85 C7                     STA   BLOKNMH
C50B:20 58 CC                  JSR   RDGBUF           ; GO READ THE NEXT LINKED DIRECTORY IN.
C50E:90 C2   C4D2              BCC   LOOKFIL1         ; BRANCH IF NO ERROR.
C510:60                        RTS                    ; RETURN ERROR (IN ACCUMULATOR).
C511:4C B0 C5       TELFREEX   JMP   TELFREE
C514:               *
C514:4C C0 C5       FNF0X      JMP   FNF0             ; AVOID BRANCH OUT OF RANGE
C517:               *
C517:        0001   CFLAG      DS    1                ; AM I CREATING?
C518:        0002   TTSAVE     DS    2                ; CURRENT BLOCK ADDR
C51A:        0002   BLOKSAVE   DS    2                ; PARENT DIR ADDR
C51C:               *
C51C:A5 0C          ERRFNF     LDA   NOFREE           ; WAS ANY FREE ENTRY FOUND?
C51E:D0 F4   C514              BNE   FNF0X
C520:AD 02 12                  LDA   GBUF+2           ; TEST LINK
C523:D0 EC   C511              BNE   TELFREEX
C525:CD 03 12                  CMP   GBUF+3           ; IF BOTH ARE ZERO, THEN GIVE UP
C528:D0 E7   C511              BNE   TELFREEX         ; BRANCH IF NOT LAST DIR BLOCK
C52A:AD 17 C5                  LDA   CFLAG            ; DOING A CREATE?
C52D:F0 E5   C514              BEQ   FNF0X            ; NO, SIMPLY REPORT NOT FOUND
C52F:               *
C52F:               * EXTEND THE DIRECTORY BY A BLOCK
C52F:               *
C52F:AD 1A C5                  LDA   BLOKSAVE         ; BUT NOT
C532:0D 1B C5                  ORA   BLOKSAVE+1       ;    IF A ROOT DIRECTORY!
C535:F0 DD   C514              BEQ   FNF0X            ;        FORU BLOCKS HARD CODED
C537:AD 76 CC                  LDA   TTLINK           ; FETCH CURRENT DIRECTORY
C53A:85 10                     STA   TLINK            ; ADDR (GBUF)
C53C:AD 77 CC                  LDA   TTLINK+1         ; AND ALLLOCATE A NEW
C53F:85 11                     STA   TLINK+1          ; BY LINKING TO CURRENT
C541:20 B4 C3                  JSR   DIRWRT
C544:B0 7A   C5C0              BCS   FNF0             ; RATS! NO SPACE SAY "DIRFULL"
C546:               *
C546:               * SAVE CURRENT BLOCK ADDR
C546:               *
C546:AD 76 CC                  LDA   TTLINK
C549:8D 18 C5                  STA   TTSAVE
C54C:AD 77 CC                  LDA   TTLINK+1
C54F:8D 19 C5                  STA   TTSAVE+1
C552:               *
C552:               * FETCH DESCENDENT
C552:               *
C552:AD 02 12                  LDA   GBUF+2
C555:85 C6                     STA   BLOKNML
C557:AD 03 12                  LDA   GBUF+3
C55A:85 C7                     STA   BLOKNMH
C55C:20 C4 C2                  JSR   ZERGBUF          ; INIT THE NEW DIR BLOCK
```

```
C55F:              *
C55F:              * AND INSERT BACK POINTER
C55F:              * TO "CURRENT BLOCK"
C55F:              *
C55F:AD 18 C5                 LDA   TTSAVE
C562:8D 00 12                 STA   GBUF
C565:AD 19 C5                 LDA   TTSAVE+1
C568:8D 01 12                 STA   GBUF+1
C56B:20 54 CC                 JSR   WRTGBUF
C56E:B0 5D   C5CD             BCS   ERTS
C570:              *
C570:              * UPDATE DIR'S HEADER IN PARENT
C570:              *
C570:AD 1A C5                 LDA   BLOKSAVE
C573:85 C6                    STA   BLOKNML          ; PREPARE TO READ PARENT
C575:AE 1B C5                 LDX   BLOKSAVE+1
C578:86 C7                    STX   BLOKNMH
C57A:20 58 CC                 JSR   RDGBUF           ; FETCH PARENT
C57D:A0 13                    LDY   #D.USAGE         ; BUMP BLOCKS USED BY HEADER
C57F:B1 AD                    LDA   (DEBUPTR),Y
C581:38                       SEC
C582:69 00                    ADC   #0               ; BY JUST ONE BLOCK
C584:91 AD                    STA   (DEBUPTR),Y
C586:C8                       INY
C587:B1 AD                    LDA   (DEBUPTR),Y      ; TWO BYTE BLOCKS USED
C589:69 00                    ADC   #0
C58B:91 AD                    STA   (DEBUPTR),Y
C58D:A0 16                    LDY   #D.EOF+1         ; INCREASE EOF BY $200
C58F:B1 AD                    LDA   (DEBUPTR),Y
C591:18                       CLC
C592:69 02                    ADC   #2
C594:91 AD                    STA   (DEBUPTR),Y
C596:C8                       INY
C597:B1 AD                    LDA   (DEBUPTR),Y
C599:69 00                    ADC   #0
C59B:91 AD                    STA   (DEBUPTR),Y
C59D:20 54 CC                 JSR   WRTGBUF          ; REWRITE PARENT DIR BLOCK
C5A0:AD 19 C5                 LDA   TTSAVE+1         ; REFETCH CURRENT DIR BLOCK
C5A3:85 C7                    STA   BLOKNMH
C5A5:AD 18 C5                 LDA   TTSAVE
C5A8:85 C6                    STA   BLOKNML
C5AA:20 58 CC                 JSR   RDGBUF           ; BACK FROM THE SHADOWS AGAIN
C5AD:4C 1C C5                 JMP   ERRFNF           ; VOILA! WE HAVE EXTENDED THE DIRECTORY!
C5B0:              *
C5B0:8D B7 DB      TELFREE    STA   D.ENTBLK
C5B3:AD 03 12                 LDA   GBUF+3
C5B6:8D B8 DB                 STA   D.ENTBLK+1       ; ASSUME FIRST ENTRY OF NEXT BLOCK
C5B9:A9 01                    LDA   #1               ;  IS FREE FOR USE.
C5BB:8D B9 DB                 STA   D.ENTNUM
C5BE:85 0C                    STA   NOFREE           ;  MARK D.ENTNUM AS VALID (FOR CREATE)
C5C0:A0 00         FNF0       LDY   #0               ; TEST FOR 'FILE NOT FOUND' VERSUS 'PATH NOT FOUND'
C5C2:B1 B0                    LDA   (PATHNML),Y
C5C4:A8                       TAY
C5C5:C8                       INY
C5C6:B1 B0                    LDA   (PATHNML),Y      ; IF NON-ZERO THEN 'PATH NOT FOUND'
C5C8:38            ERRPATH1   SEC                    ; IN EITHER CASE, INDICATE ERROR.
```

```
C5C9:F0 03   C5CE                 BEQ   FNF1
C5CB:A9 00                        LDA   #PATHNOTFND      ; REPORT NO SUCH PATH.
C5CD:60                ERTS       RTS
C5CE:A9 00             FNF1       LDA   #FNFERR          ; REPORT FILE NOT FOUND.
C5D0:60                           RTS
```

```
C5D1:              *
C5D1:B1 B0         NAMFOUND   LDA   (PATHNML),Y      ; (Y=0)
C5D3:38                       SEC
C5D4:65 B0                    ADC   PATHNML          ; TEST FOR LAST NAME IN PATH
C5D6:A8                       TAY                    ; IF ZERO, THEN THAT WAS LAST NAME
C5D7:18                       CLC                    ; TO INDICATE SUCCESS
C5D8:B9 00 10                 LDA   PATHBUF,Y
C5DB:F0 59   C636             BEQ   FILFOUND
C5DD:              *NOW CHANGE THE PATHNAME POINTER TO POINT AT THE NEXT NAME IN THE PATH
C5DD:84 B0                    STY   PATHNML
C5DF:A5 B4                    LDA   DRBUFPL          ; SAVE PARENTS
C5E1:85 AD                    STA   DEBUPTR          ; ENTRY POINTER
C5E3:A5 B5                    LDA   DRBUFPH
C5E5:85 AE                    STA   DEBUPTR+1        ; IN CASE ENTRY ON PAGE 2
C5E7:A5 C6                    LDA   BLOKNML          ; ADDRESS (DIR EXTEND)
C5E9:8D 1A C5                 STA   BLOKSAVE
C5EC:A5 C7                    LDA   BLOKNMH
C5EE:8D 1B C5                 STA   BLOKSAVE+1
C5F1:A0 00                    LDY   #D.STOR          ; BE SURE THIS IS A DIRECTORY ENTRY
C5F3:B1 B4                    LDA   (DRBUFPL),Y      ; HIGH NIBBLE WILL TELL
C5F5:29 F0                    AND   #$F0
C5F7:C9 D0                    CMP   #DIRTYP*16       ; IS IT A SUB-DIRECTORY?
C5F9:D0 CD   C5C8             BNE   ERRPATH1         ; REPORT THE USER'S MISTAKE
C5FB:A0 11                    LDY   #D.FRST          ; GET ADDRESS OF FIRST SUB-DIRECTORY BLOCK
C5FD:B1 B4                    LDA   (DRBUFPL),Y
C5FF:85 C6                    STA   BLOKNML          ; (NO CHECKING IS DONE HERE FOR A VALID
C601:C8                       INY                    ;  BLOCK NUMBER... )
C602:8D B5 DB                 STA   D.HEAD           ; SAVE AS FILE'S HEADER BLOCK TOO.
C605:B1 B4                    LDA   (DRBUFPL),Y
C607:85 C7                    STA   BLOKNMH
C609:8D B6 DB                 STA   D.HEAD+1
C60C:20 58 CC                 JSR   RDGBUF           ; READ SUB-DIRECTORY INTO GBUF
C60F:B0 11   C622             BCS   FNDERR1          ; RETURN IMMEDIATELY ANY ERROR ENCOUNTERED.
C611:AD 25 12                 LDA   GBUF+HCENT+4     ; GET THE NUMBER OF FILES
C614:85 09                    STA   ENTCNTL          ;  CONTAINED IN THIS DIRECTORY
C616:AD 26 12                 LDA   GBUF+HCENT+5
C619:85 0A                    STA   ENTCNTH
C61B:AD 21 12                 LDA   GBUF+HCMP+4      ; TEST BACKWARD COMPATIBILITY
C61E:F0 04   C624             BEQ   MOVHEAD
C620:A9 00         ERRCOMP    LDA   #CPTERR          ; TELL THEM THIS DIRECTORY IS NOT COMPATABLE
C622:        C622  NONAME     EQU   *
C622:38            FNDERR1    SEC
C623:60                       RTS
C624:20 2A C6      MOVHEAD    JSR   MOVHED0          ; MOVE INFO ABOUT THIS DIRECTORY
C627:4C CD C4                 JMP   LOOKFIL0         ; DO NEXT LOCAL PATHNAME
C62A:              *
C62A:A2 0A         MOVHED0    LDX   #$A              ; MOVE INFO ABOUT THIS DIRECTORY
C62C:BD 1C 12      MOVHED1    LDA   GBUF+HCRDT+4,X
C62F:9D A0 DB                 STA   H.CREDT,X
C632:CA                       DEX
C633:10 F7   C62C             BPL   MOVHED1
C635:60                       RTS
C636:              *
```

```
C636:              *
C636:              *
C636:        C636  FILFOUND   EQU   *
C636:AD A8 DB      ENTADR     LDA   H.MAXENT         ; FIGURE OUT WHICH IS ENTRY NUMBER THIS IS.
C639:38                       SEC
C63A:E5 0B                    SBC   CNTENT           ; MAX ENTRIES - COUNT ENTRIES + 1 = ENTRY NUMBER
C63C:69 00                    ADC   #0               ; (CARRY IS/WAS SET)
C63E:8D B9 DB                 STA   D.ENTNUM
C641:A5 C6                    LDA   BLOKNML
C643:8D B7 DB                 STA   D.ENTBLK
C646:A5 C7                    LDA   BLOKNMH          ; AND INDICATE BLOCK NUMBER OF THIS DIRECTORY.
C648:8D B8 DB                 STA   D.ENTBLK+1
C64B:18                       CLC
C64C:60                       RTS
C64D:              *
C64D:AD A8 DB      LOOKNAM    LDA   H.MAXENT         ; RESET COUNT OF FILES PER BLOCK
C650:85 0B                    STA   CNTENT
C652:A9 12                    LDA   #GBUF/256
C654:85 B5                    STA   DRBUFPH
C656:A9 04                    LDA   #4
C658:85 B4         LOKNAM1    STA   DRBUFPL          ; RESET INDIRECT POINTER TO GBUF
C65A:B0 25   C681             BCS   LOKNAM2          ; BRANCH IF THIS BLOCK CONTAINS A HEADER
C65C:A0 00                    LDY   #D.STOR
C65E:B1 B4                    LDA   (DRBUFPL),Y      ; GET LENGTH OF NAME IN DIRECTORY
C660:D0 0B   C66D             BNE   ISNAME           ; BRANCH IF THERE IS A NAME.
C662:A5 0C                    LDA   NOFREE           ; TEST TO SEE IF A FREE ENTRY HAS BEEN DECLARED.
C664:D0 1B   C681             BNE   LOKNAM2          ; YES BUMP TO NEXT ENTRY
C666:20 36 C6                 JSR   ENTADR           ; SET ADDRESS FOR CURRENT ENTRY
C669:E6 0C                    INC   NOFREE           ; INDICATE A FREE SPOT HAS BEEN FOUND
C66B:D0 14   C681             BNE   LOKNAM2          ; BRANCH ALWAYS.
C66D:              *
C66D:29 0F         ISNAME     AND   #$F              ; STRIP TYPE (THIS IS CHECKED BY 'FILFOUND')
C66F:E6 08                    INC   TOTENT           ; (BUMP COUNT OF VALID FILES FOUND)
C671:D1 B0                    CMP   (PATHNML),Y      ; ARE BOTH NAMES OF THE SAME LENGTH?
C673:D0 0C   C681             BNE   LOKNAM2          ; NO, BUMP TO NEXT ENTRY
C675:A8                       TAY
C676:B1 B4         CMPNAME    LDA   (DRBUFPL),Y      ; COMPARE NAMES LETTER BY LETTER
C678:D1 B0                    CMP   (PATHNML),Y
C67A:D0 05   C681             BNE   LOKNAM2
C67C:88                       DEY                    ; HAVE ALL LETTERS BEEN COMPARED?
C67D:D0 F7   C676             BNE   CMPNAME          ; NO, CONTINUE..
C67F:18                       CLC                    ; BY GOLLY, WE GOT US A MATCH!
C680:60                       RTS
C681:              *
C681:C6 0B         LOKNAM2    DEC   CNTENT           ; HAVE WE CHECKED ALL POSSIBLE ENTRIES IN THIS BLOCK?
C683:F0 9D   C622             BEQ   NONAME           ; YES, GIVE UP.
C685:AD A7 DB                 LDA   H.ENTLN          ; ADD ENTRY LENGTH TO CURRENT POINTER
C688:18                       CLC
C689:65 B4                    ADC   DRBUFPL
C68B:90 CB   C658             BCC   LOKNAM1          ; BRANCH IF WE'RE STILL IN THE FIRST PAGE.
C68D:E6 B5                    INC   DRBUFPH          ; LOOK ON SECOND PAGE
C68F:18                       CLC                    ; CARRY SHOULD ALWAYS BE CLEAR BEFORE LOOKING AT NEXT.
C690:90 C6   C658             BCC   LOKNAM1          ; BRANCH ALWAYS...
```

```
C692:              *
C692:              *
C692:20 1E C7      PREPROOT   JSR   FINDVOL          ; FIND CORRECT VOLUME AND DEVICE NUMBER
C695:90 05   C69C             BCC   ROOT1            ; BRANCH IF IT WAS FOUND.
C697:20 62 C7      ROOT0      JSR   LOOKVOL          ; OTHERWISE LOOK ON ALL DEVICES.
C69A:B0 4C   C6E8             BCS   SRITZ            ; CAN'T FIND IT.
C69C:A9 00         ROOT1      LDA   #0               ; ZERO OUT DIRECTORY TEMPS
C69E:A0 2A                    LDY   #42              ; (DECIMAL)
C6A0:99 B4 DB      CLRDSP     STA   D.DEV,Y
C6A3:88                       DEY
C6A4:10 FA   C6A0             BPL   CLRDSP
C6A6:A0 10                    LDY   #VCBDEV          ; SET UP DEVICE NUMBER
C6A8:B1 B6                    LDA   (VCBPTR),Y
C6AA:85 35                    STA   DEVNUM
C6AC:8D B4 DB                 STA   D.DEV            ; FOR FUTURE REFERENCE
C6AF:C8                       INY
C6B0:B1 B6                    LDA   (VCBPTR),Y       ; GET CURRENT STATUS OF THIS VOLUME
C6B2:8D 9F DB                 STA   V.STATUS
C6B5:A0 16                    LDY   #VCBROOT         ; GET BLOCK ADDRESS OF ROOT DIRECTORY TOO.
C6B7:B1 B6                    LDA   (VCBPTR),Y
C6B9:85 C6                    STA   BLOKNML
C6BB:8D B5 DB                 STA   D.HEAD           ; PRESERVE AS HEADER
C6BE:C8                       INY
C6BF:B1 B6                    LDA   (VCBPTR),Y
C6C1:85 C7                    STA   BLOKNMH
C6C3:8D B6 DB                 STA   D.HEAD+1
C6C6:20 58 CC                 JSR   RDGBUF           ; GO READ IN ROOT
C6C9:90 0B   C6D6             BCC   ROOT2            ; BRANCH IF NO ERROR
C6CB:48                       PHA                    ; SAVE ERROR CODE
C6CC:A0 11                    LDY   #VCBSTAT         ; CHECK THIS BUGGER FOR AN OPEN FILE.
C6CE:B1 B6                    LDA   (VCBPTR),Y
C6D0:0A                       ASL   A                ; (SHIFT OPEN STATUS INTO CARRY)
C6D1:68                       PLA                    ; GET ERROR CODE AGAIN
C6D2:B0 31   C705             BCS   ROOTERR          ;  BRANCH IF ERROR NEEDS TO BE REPORTED
C6D4:D0 C1   C697             BNE   ROOT0            ; OTHERWISE, LOOK ELSEWHERE (BRANCH ALWAYS).
C6D6:              *
C6D6:20 06 C7      ROOT2      JSR   CHKROOT          ; VERIFY ROOT NAME
C6D9:F0 0E   C6E9             BEQ   ROOT3            ; BRANCH IF MATCHED.
C6DB:A0 11                    LDY   #VCBSTAT         ; TEST FOR OPEN FILES ON THIS VOLUME BEFORE
C6DD:B1 B6                    LDA   (VCBPTR),Y       ;  LOOKING FOR IT ELSEWHERE.
C6DF:10 B6   C697             BPL   ROOT0
C6E1:20 2F DD                 JSR   USRREQ           ;  REQUEST USER MOUNT VOLUME
C6E4:90 B6   C69C             BCC   ROOT1            ;  USER SAID S/HE DID-- CHECK IT
C6E6:A9 00                    LDA   #VNFERR          ; REPORT VOLUME NOT FOUND ERR IF REFUSE TO INSERT
C6E8:60            SRITZ      RTS
C6E9:              *
```

```
C6E9:A0 0F         ROOT3      LDY   #$F              ; (NOTE: X CONTAINS THE LENGTH OF THE ROOT NAME)
C6EB:B9 1B 12      ROOTINFO   LDA   GBUF+HCRDT+3,Y   ; SAVE HEADER INFO.
C6EE:99 9F DB                 STA   V.STATUS,Y
C6F1:88                       DEY
C6F2:D0 F7   C6EB             BNE   ROOTINFO         ; LOOP TIL ALL 15 BYTES MOVED
C6F4:AD A9 DB                 LDA   H.FCNT
C6F7:85 09                    STA   ENTCNTL
C6F9:AD AA DB                 LDA   H.FCNT+1
C6FC:85 0A                    STA   ENTCNTH
C6FE:8A                       TXA                    ; NOW THAT ROOT IS IDENTIFIED, ADJUST
C6FF:38                       SEC                    ;  PATH NAME POINTER TO NEXT NAME IN THE PATH
C700:65 B0                    ADC   PATHNML
C702:85 B0                    STA   PATHNML
C704:18                       CLC                    ; INDICATE NO ERROR
C705:60            ROOTERR    RTS
C706:              *
C706:              *
C706:A0 00         CHKROOT    LDY   #0               ; GET LENGTH OF NAME
C708:B1 B0                    LDA   (PATHNML),Y
C70A:A8                       TAY
C70B:AA                       TAX                    ; SAVE IN X FOR LATTER ADJUSTMENT TO PATH POINTER
C70C:4D 04 12                 EOR   GBUF+4
C70F:29 0F                    AND   #$F              ; DOES PATHNAME HAVE SAME LENGTH AS DIRECTORY NAME?
C711:D0 0A   C71D             BNE   NOTROOT          ; BRANCH IF NOT
C713:B1 B0         CKROOT1    LDA   (PATHNML),Y      ; COMPARE CHARACTER BY CHARACTER
C715:D9 04 12                 CMP   GBUF+4,Y
C718:D0 03   C71D             BNE   NOTROOT
C71A:88                       DEY
C71B:D0 F6   C713             BNE   CKROOT1          ; LOOP UNTIL ALL CHARACTERS MATCH
C71D:60            NOTROOT    RTS
C71E:              *
```

```
C71E:A9 11         FINDVOL    LDA   #VCB/256         ; SEARCH VCB FOR VOLUME NAME
C720:85 B7                    STA   VCBPTR+1
C722:A9 00                    LDA   #0
C724:8D B4 DB                 STA   D.DEV
C727:85 B6                    STA   VCBPTR
C729:48            FNDVOL1    PHA                    ; SAVE LAST SEARCH POSITION
C72A:AA                       TAX
C72B:A0 00                    LDY   #0               ; (INDEX TO PATHNAME POINTER)
C72D:BD 00 11                 LDA   VCB,X            ; GET LENGTH OF VOLUME NAME TO COMPARE
C730:F0 29   C75B             BEQ   NXTVCB           ; BRANCH IF VCB ENTRY IS EMPTY
C732:D1 B0                    CMP   (PATHNML),Y      ; ARE NAMES OF SAME LENGTH?
C734:D0 25   C75B             BNE   NXTVCB           ; NO, INDEX NEXT VCB
C736:18                       CLC                    ; SCAN NAME BACKWARDS
C737:A8                       TAY
C738:8A                       TXA
C739:7D 00 11                 ADC   VCB,X
C73C:AA                       TAX                    ; NOW BOTH INDEXES POINT TO LAST CHARACTER OF THE NAMES TO
COMPARE
C73D:B1 B0         VOLNAM     LDA   (PATHNML),Y
C73F:DD 00 11                 CMP   VCB,X
C742:D0 17   C75B             BNE   NXTVCB
C744:CA                       DEX
C745:88                       DEY
C746:D0 F5   C73D             BNE   VOLNAM           ; CHECK ALL CHARACTERS
C748:68                       PLA                    ; SINCE A MATCH IS FOUND
C749:85 B6                    STA   VCBPTR           ;  SET UP INDEX TO VCB ENTRY
C74B:AA                       TAX
C74C:BD 1F 11                 LDA   VCB+VCBSWAP,X    ;  BRANCH IF
C74F:F0 08   C759             BEQ   FOUNDVOL         ;  VOLUME NOT SWAPPED
C751:20 51 DC                 JSR   SWAPIN           ;  IF USER REALLY WANTS IT, THEN BRING IN IF SWAPPED
C754:90 03   C759             BCC   FOUNDVOL         ;  BRANCH IF SUCCESS
C756:A9 00                    LDA   #XIOERROR        ;  USER REFUSES TO MOUNT
C758:60                       RTS
C759:18            FOUNDVOL   CLC                    ; INDICATE VOLUME FOUND
C75A:60                       RTS
C75B:              *
C75B:68            NXTVCB     PLA                    ; GET CURRENT INDEX AGAIN.
C75C:18                       CLC
C75D:69 20                    ADC   #VCBSIZE         ; VCB ENTRY LENGTH.
C75F:90 C8   C729             BCC   FNDVOL1          ; BRANCH IF THER IS ANOTHER TO CHECK
C761:60                       RTS                    ; RETURN WITH CARRY SET TO SHOW FAILURE.
```

```
C762:              *
C762:              *
C762:A2 0C         LOOKVOL    LDX   #12              ; (1) COUNT+(12)DEVICE LIST
C764:BD 00 00      LOOKVOL1   LDA   BLKDLST,X        ;  EXTRN
C767:9D E3 DB                 STA   SCRTCH,X         ;  MY CHANGEABLE COPY
C76A:CA                       DEX
C76B:10 F7   C764             BPL   LOOKVOL1         ;  WORK BACKWARDS SO
C76D:85 36                    STA   TOTDEVS          ;  ENTRY ZERO IS TOTAL DEVICES LISTED
C76F:E8                       INX                    ;  MAKE XREG = ZERO
C770:E8            LOKDEV1    INX
C771:8E E3 DB                 STX   SCRTCH
C774:BD E3 DB                 LDA   SCRTCH,X
C777:CD B4 DB                 CMP   D.DEV
C77A:F0 5A   C7D6             BEQ   NXTDEV           ; DON'T LOOK AGAIN ON A DRIVE THAT HAS BEEN CHECKED
C77C:85 35                    STA   DEVNUM           ; CHECK FOR DEVICE ALREADY LOGGED IN A VCB
C77E:20 48 C8                 JSR   DEVVCB           ; (CARRY CLEAR IF IT'S THERE)
C781:90 2F   C7B2             BCC   LOKVOL1
C783:A9 00                    LDA   #0               ; FIND A FREE VCB TO LOG THIS GUY IN
C785:AA            ENTVCB     TAX                    ; INDEX TO NEXT VCB ENTRY
C786:BD 00 11                 LDA   VCB,X
C789:F0 1F   C7AA             BEQ   FREEVCB          ; FOUND A FREE SPOT.
C78B:8A                       TXA                    ; NOW INDEX TO NEXT, AND KEEP LOOKIN
C78C:18                       CLC
C78D:69 20                    ADC   #VCBSIZE         ; (EACH VCB ENTRY IS 32 BYTES)
C78F:90 F4   C785             BCC   ENTVCB           ; BRANCH IF MORE TO FIND
C791:A9 00                    LDA   #0
C793:        C793  ENTVCB2    EQU   *                ; SEE IF WE CAN REPLACE A DEVICE
C793:AA                       TAX
C794:BD 11 11                 LDA   VCB+VCBSTAT,X    ; VCB HAS FILES OPEN?
C797:F0 11   C7AA             BEQ   FREEVCB          ; NO, USE IT!
C799:8A                       TXA
C79A:18                       CLC
C79B:69 20                    ADC   #VCBSIZE         ; SEARCH NEXT VCB ENTRY
C79D:90 F4   C793             BCC   ENTVCB2
C79F:60                       RTS                    ; FAILED TO FIND A FREE VCB ENTRY
C7A0:              *
C7A0:A0 00         CHKVLOG    LDY   #0               ; MAKE SURE VOLUME WAS ACTUALLY LOGGED IN
C7A2:B1 B6                    LDA   (VCBPTR),Y
C7A4:D0 B3   C759             BNE   FOUNDVOL         ; AH, MADE IT...
C7A6:A9 00                    LDA   #DUPVOL          ; WELL, NOT QUITE, THIS VOLUME CAN'T BE LOGGED
C7A8:38                       SEC
C7A9:60                       RTS
```

```
C7AA:              *
C7AA:86 B6         FREEVCB    STX   VCBPTR           ; NOW THIS IS THE POINTER TO A FREE VCB
C7AC:A9 02                    LDA   #2               ; ROOT DIRECTORIES ALWAYS AT BLOCK 2
C7AE:A2 00                    LDX   #0
C7B0:F0 0E   C7C0             BEQ   GETROOT          ; BRANCH ALWAYS
C7B2:A0 11         LOKVOL1    LDY   #VCBSTAT         ; MAKE SURE NO FILES ARE ACTIVE ON
C7B4:B1 B6                    LDA   (VCBPTR),Y       ;  THE VOLUME BEFORE LOGGING IT IN.
C7B6:30 28   C7E0             BMI   SNSWIT           ;  BRANCH IF FILES ACTIVE
C7B8:A0 17                    LDY   #VCBROOT+1       ; GET ADDRESS OF ROOT DIRECTORY
C7BA:B1 B6                    LDA   (VCBPTR),Y       ; HIGH FIRST.
C7BC:AA                       TAX
C7BD:88                       DEY                    ; THEN LOW.
C7BE:B1 B6                    LDA   (VCBPTR),Y
C7C0:20 1E C9      GETROOT    JSR   GETROT0
C7C3:90 07   C7CC             BCC   LOKVOL2          ; BRANCH IF SUCCESSFULLY READ.
C7C5:A9 00                    LDA   #0               ; OTHERWISE, TAKE THIS DEVICE OUT OF VCB
C7C7:A8                       TAY
C7C8:91 B6                    STA   (VCBPTR),Y       ; (VOLUME 'OFF LINE')
C7CA:F0 0A   C7D6             BEQ   NXTDEV           ; BRANCH ALWAYS
C7CC:              *
C7CC:20 8F C8      LOKVOL2    JSR   LOGVCB           ; GO UPDATE VCB TO INCLUDE CURRENT VOLUME INFO
C7CF:B0 05   C7D6             BCS   NXTDEV           ;  IF NOT A SOS DISKETTE, SKIP TO NEXT DEVICE
C7D1:20 06 C7                 JSR   CHKROOT          ; GO COMPARE TO SEE IF WE FOUND WHAT WE'RE
C7D4:F0 CA   C7A0             BEQ   CHKVLOG          ;  LOOKING FOR...
C7D6:              *
C7D6:AE E3 DB      NXTDEV     LDX   SCRTCH           ; LOOK AT OTHER DEVICES?
C7D9:E4 36                    CPX   TOTDEVS
C7DB:90 93   C770             BCC   LOKDEV1          ; YES.
C7DD:A9 00                    LDA   #VNFERR          ; REPORT VOLUME NOT FOUND.
C7DF:60                       RTS
C7E0:              *
C7E0:        C7E0  SNSWIT     EQU   *                ;  SENSE DSWITCH
C7E0:A0 10                    LDY   #VCBDEV
C7E2:B1 B6                    LDA   (VCBPTR),Y
C7E4:85 35                    STA   DEVNUM           ;  MAKE SURE DEVICE NUMBER IS CURRENT
C7E6:20 87 D5                 JSR   TWRPROT1         ;  USES DEVNUM
C7E9:AD BB D5                 LDA   DSWGLOB          ;  DISK SWITCH GLOBAL
C7EC:F0 E8   C7D6             BEQ   NXTDEV           ;  BRANCH IF NO DISK SWITCH
C7EE:20 0A C9                 JSR   VERFYVOL         ;  COMPARES VCBPTR VS. DEVNUM CONTENTS
C7F1:90 E3   C7D6             BCC   NXTDEV           ;  BRANCH IF DISK HAS NOT BEEN SWITCHED
C7F3:20 06 C7                 JSR   CHKROOT          ;  COMPARES PATHNML VS. GBUF
C7F6:D0 DE   C7D6             BNE   NXTDEV           ;  IGNORE IF NOT WHAT WE ARE LOOKING FOR
C7F8:A2 00                    LDX   #0               ;  LOOK FOR FREE
C7FA:20 02 C8                 JSR   SNSWIT1
C7FD:B0 D7   C7D6             BCS   NXTDEV           ;  ANY ERRORS LOGGING IN THE NEW VOLUME
C7FF:4C A0 C7                 JMP   CHKVLOG          ;  MAKE SURE THE NEW VOLUME IS LOGGED
C802:BD 00 11      SNSWIT1    LDA   VCB,X            ;  VCB ENTRY
C805:F0 08   C80F             BEQ   SNSWIT2          ;  BRANCH IF FOUND
C807:8A                       TXA
C808:18                       CLC
C809:69 20                    ADC   #VCBSIZE         ;  LOOK AT NEXT VCB AREA
C80B:AA                       TAX
C80C:90 F4   C802             BCC   SNSWIT1
C80E:60                       RTS                    ;  CAN'T BE LOGGED IN!
C80F:A9 00         SNSWIT2    LDA   #0
C811:85 3C                    STA   DUPLFLAG         ;  TURN OFF DUPLICATE VOLUME FLAG
```

```
C813:86 B6                    STX   VCBPTR
C815:20 9A C8                 JSR   LOGVCB1         ;  PARTIALLY LOG IN THE NEW VOLUME
C818:B0 2B   C845             BCS   NONSOS          ;  CS MEANS NONSOS ERROR
C81A:A5 3C                    LDA   DUPLFLAG        ;  WAS IT A DUPLICATE VOLUME?
C81C:D0 23   C841             BNE   SNSWIT6         ;  BRANCH IF YES
C81E:A0 1F                    LDY   #VCBSWAP        ;  BY MAKING SWAP BYTE NON ZERO
C820:A9 01                    LDA   #1
C822:91 B6                    STA   (VCBPTR),Y      ;  SO SWAPOUT WON'T AFFECT
C824:A5 35                    LDA   DEVNUM          ;  A REG PASSES DEVNUM TO SWAPOUT
C826:20 F6 DB                 JSR   SWAPOUT         ;  OLD ACTIVE MOUNT MUST BE SWAPPED
C829:90 03   C82E             BCC   SNSWIT3
C82B:A9 00                    LDA   #XIOERROR       ;  USER REFUSED TO REPLACE OLD VOLUME
C82D:60                       RTS
C82E:A0 1F        SNSWIT3     LDY   #VCBSWAP        ;  NOW LOG IN THE NEW ALL THE WAY
C830:A9 00                    LDA   #0
C832:91 B6                    STA   (VCBPTR),Y
C834:20 0A C9     SNSWIT4     JSR   VERFYVOL        ;  DON'T BOTHER TO ASK IF NEW VOLUME IS ALREADY MOUNTED
C837:90 07   C840             BCC   SNSWIT5         ;  BRANCH IF NEW VOLUME ON LINE
C839:20 2F DD                 JSR   USRREQ          ;  ASK USER TO REMOUNT NEW VOLUME
C83C:90 F6   C834             BCC   SNSWIT4         ;  USER SAYS THEY DID: CHECK IT OUT
C83E:A9 00                    LDA   #VNFERR
C840:60           SNSWIT5     RTS
C841:A9 00        SNSWIT6     LDA   #DUPVOL
C843:38                       SEC
C844:60                       RTS
```

```
C845:              *
C845:A9 00         NONSOS     LDA   #NOTSOS          ; TELL EM IT'S NOT A SOS DISK (COULD BE PASCAL)
C847:60                       RTS                    ;  CARRY SHOULD ALREADY BE SET
C848:              *
C848:              *
C848:A9 00         DEVVCB     LDA   #0               ; SCAN VCB FOR DEVICE SPECIFIED IN 'DEVNUM'
C84A:AA            DVCB1      TAX                    ; FIRST TEST FOR VALID VCB.
C84B:BD 00 11                 LDA   VCB,X
C84E:F0 0C   C85C             BEQ   DVCB2
C850:BD 1F 11                 LDA   VCB+VCBSWAP,X    ;  SWAPPED VOLUMES DON'T COUNT
C853:D0 07   C85C             BNE   DVCB2            ;  AS LOGGED IN
C855:BD 10 11                 LDA   VCB+VCBDEV,X     ; GET DEVICE NUMBER
C858:C5 35                    CMP   DEVNUM           ; TEST AGAINST REQUESTED DEVICE
C85A:F0 26   C882             BEQ   FOUNDEV          ; YES, SET UP A POINTER TO IT
C85C:8A            DVCB2      TXA                    ; BUMP TO NEXT VCB
C85D:18                       CLC
C85E:69 20                    ADC   #VCBSIZE
C860:90 E8   C84A             BCC   DVCB1            ; BRANCH IF MORE TO LOOK AT.
C862:60                       RTS                    ; RETURN CARRY SET TO INDICATE NOT FOUND
C863:              *
C863:A6 B6         TSTDUPVOL  LDX   VCBPTR           ; PRESERVE CURRENT ADDR OF FREE VCB
C865:A9 00                    LDA   #0               ; LOOK FOR A CURRENTLY LOGGED ON VOLUME OF THE SAME NAME.
C867:85 B6         TSDUPV1    STA   VCBPTR
C869:20 F2 C8                 JSR   CMPVCB
C86C:B0 0D   C87B             BCS   TSDUPV2          ; BRANCH IF NO MATCH.
C86E:A0 11                    LDY   #VCBSTAT         ; TEST FOR ANY OPEN FILES.
C870:B1 B6                    LDA   (VCBPTR),Y
C872:30 12   C886             BMI   FOUNDDUP         ; TELL THE SUCKER HE CAN'T LOOK AT THIS VOLUME!
C874:A9 00                    LDA   #0               ; TAKE DUPLICATE OFF LINE IF NO OPEN FILES.
C876:A8                       TAY
C877:91 B6                    STA   (VCBPTR),Y
C879:F0 07   C882             BEQ   NODUPVOL         ; RETURN THAT ALL IS OK TO LOG IN NEW.
C87B:A5 B6         TSDUPV2    LDA   VCBPTR
C87D:18                       CLC
C87E:69 20                    ADC   #VCBSIZE         ; BUMP TO NEXT ENTRY.
C880:90 E5   C867             BCC   TSDUPV1
C882:        C882  NODUPVOL   EQU   *
C882:18            FOUNDEV    CLC
C883:86 B6         FNDDUP1    STX   VCBPTR
C885:60                       RTS
C886:              *
C886:85 3C         FOUNDDUP   STA   DUPLFLAG         ; A DUPLICATE HAS BEEN DETECTED.
C888:38                       SEC                    ; INDICATE ERROR
C889:A5 B6                    LDA   VCBPTR           ;  SAVE ADDRESS OF DUPLICATE
C88B:85 3E                    STA   VCBENTRY
C88D:B0 F4   C883             BCS   FNDDUP1          ; BRANCH ALWAYS TAKEN
```

```
C88F:              *
C88F:              *
C88F:A0 00         LOGVCB     LDY   #VCBNML          ; IS THIS A PREVIOUSLY LOGGED IN VOLUME
C891:B1 B6                    LDA   (VCBPTR),Y       ; (ACC=0?)
C893:F0 05   C89A             BEQ   LOGVCB1          ; NO, GO AHEAD AND PREPARE VCB.
C895:20 F2 C8                 JSR   CMPVCB           ; DOES VCB MATCH VOLUME READ?
C898:90 54   C8EE             BCC   VCBLOGD          ; YES, DON'T DISTURB IT.
C89A:A9 00         LOGVCB1    LDA   #0               ; ZERO OUT VCB ENTRY
C89C:A0 1F                    LDY   #VCBSIZE-1
C89E:91 B6         ZERVCB     STA   (VCBPTR),Y
C8A0:88                       DEY
C8A1:10 FB   C89E             BPL   ZERVCB
C8A3:20 65 C4                 JSR   TSTSOS           ; MAKE SURE IT'S A SOS DISKETTE.
C8A6:B0 46   C8EE             BCS   VCBLOGD          ; IF NOT, RETURN CARRY SET.
C8A8:20 63 C8                 JSR   TSTDUPVOL        ; FIND OUT IF A DUPLICATE WITH OPEN FILES ALREADY EXISTS
C8AB:B0 42   C8EF             BCS   NOTLOG0
C8AD:AD 04 12                 LDA   GBUF+4           ; MOVE VOLUME NAME TO VCB
C8B0:29 0F                    AND   #$F              ; STRIP ROOT MARKER
C8B2:A8                       TAY
C8B3:48                       PHA
C8B4:B9 04 12      MOVOLNM    LDA   GBUF+4,Y
C8B7:91 B6                    STA   (VCBPTR),Y
C8B9:88                       DEY
C8BA:D0 F8   C8B4             BNE   MOVOLNM
C8BC:68                       PLA                    ; GET LENGTH AGAIN
C8BD:91 B6                    STA   (VCBPTR),Y       ; SAVE THAT TOO.
C8BF:A0 10                    LDY   #VCBDEV          ; SAVE DEVICE NUMBER ALSO.
C8C1:A5 35                    LDA   DEVNUM
C8C3:91 B6                    STA   (VCBPTR),Y
C8C5:20 F8 CB                 JSR   CLEARBMS         ;  MARKS THIS DEVICES OLD BITMAPS AS INVALID (A REG PASSED)
C8C8:AD 29 12                 LDA   GBUF+VTBLK+4     ; AND TOTOL NUMBER OF BLOCKS ON THIS UNIT,
C8CB:A0 12                    LDY   #VCBTBLK
C8CD:91 B6                    STA   (VCBPTR),Y
C8CF:AD 2A 12                 LDA   GBUF+VTBLK+5
C8D2:C8                       INY
C8D3:91 B6                    STA   (VCBPTR),Y
C8D5:A0 16                    LDY   #VCBROOT
C8D7:A5 C6                    LDA   BLOKNML          ; AND ADDRESS OF ROOT DIRECTORY
C8D9:91 B6                    STA   (VCBPTR),Y
C8DB:C8                       INY
C8DC:A5 C7                    LDA   BLOKNMH
C8DE:91 B6                    STA   (VCBPTR),Y
C8E0:A0 1A                    LDY   #VCBDMAP
C8E2:AD 27 12                 LDA   GBUF+VBMAP+4     ; AND LASTLY, THE ADDRESS
C8E5:91 B6                    STA   (VCBPTR),Y       ;  OF THE FIRST BITMAP
C8E7:AD 28 12                 LDA   GBUF+VBMAP+5
C8EA:C8                       INY
C8EB:91 B6                    STA   (VCBPTR),Y
C8ED:18                       CLC                    ; INDICATE THAT IT WAS LOGGED IF POSIBLE.
C8EE:60            VCBLOGD    RTS
C8EF:4C 29 C9      NOTLOG0    JMP   NOTLOG1
```

```
C8F2:AD 04 12      CMPVCB     LDA   GBUF+4           ; COMPARE VOLUME NAME IN VCB
C8F5:29 0F                    AND   #$F
C8F7:A0 00                    LDY   #VCBNML          ;  WITH NAME IN DIRECTORY
C8F9:D1 B6                    CMP   (VCBPTR),Y       ; ARE THEY SAME LENGTH
C8FB:D0 2A   C927             BNE   NOTSAME
C8FD:A8                       TAY
C8FE:B9 04 12      VCBCMP1    LDA   GBUF+4,Y
C901:D1 B6                    CMP   (VCBPTR),Y
C903:D0 22   C927             BNE   NOTSAME
C905:88                       DEY
C906:D0 F6   C8FE             BNE   VCBCMP1
C908:18                       CLC                    ; INDICATE MATCH.
C909:60                       RTS
C90A:              *
C90A:A2 00         VERFYVOL   LDX   #0               ; READ IN ROOT DIRECTORY HEADER.
C90C:A9 02                    LDA   #2
C90E:20 1E C9                 JSR   GETROT0
C911:B0 08   C91B             BCS   NOVRFY1          ; PASS BACK WHATEVER OTHER ERROR OCCURS.
C913:20 F2 C8                 JSR   CMPVCB           ; TEST ROOT WITH VOLUME NAME IN VCB.
C916:90 02   C91A             BCC   NOVRFY           ;  BRANCH IF ROOT MATCHES VCB
C918:A9 00                    LDA   #0               ;  OTHERWISE, PASS BACK FOREIGN VOLUME ERROR (SOS OR UCSD)
C91A:60            NOVRFY     RTS                    ; RETURN RESULTS IN CARRY.
C91B:A9 00         NOVRFY1    LDA   #VNFERR          ;  NOTHING IN DRIVE
C91D:60                       RTS
C91E:              *
C91E:85 C6         GETROT0    STA   BLOKNML
C920:86 C7                    STX   BLOKNMH          ; STORE ADDRESS AND READ IN ROOT
C922:20 58 CC                 JSR   RDGBUF
C925:90 01   C928             BCC   RETROT2          ; BRANCH IF SUCCESSFULLY READ.
C927:        C927  NOTSAME    EQU   *
C927:38                       SEC                    ; INDICATE ERROR
C928:60            RETROT2    RTS
C929:              *
C929:A6 B6         NOTLOG1    LDX   VCBPTR           ;  LOAD THE VCB ADDRESS
C92B:A5 3E                    LDA   VCBENTRY         ;  OF THE DUPLICATE VOLUME
C92D:85 B6                    STA   VCBPTR
C92F:86 3E                    STX   VCBENTRY         ;  AND SAVE THE FREE VCB SPACE ADDR
C931:A0 10                    LDY   #VCBDEV          ;  IS DUPLICATE ON SAME DEVICE?
C933:A5 35                    LDA   DEVNUM
C935:D1 B6                    CMP   (VCBPTR),Y
C937:D0 0D   C946             BNE   NOTLOG2          ;  BRANCH IF NOT
C939:20 51 DC                 JSR   SWAPIN           ;  SWAP IN IF NECESSARY
C93C:A9 00                    LDA   #0
C93E:85 3C                    STA   DUPLFLAG         ;  NO MORE DUPLICATE VOLUME STATUS
C940:A5 B6                    LDA   VCBPTR           ;  MAKE CHKROOT WORK IN A MOMENT
C942:85 B0                    STA   PATHNML          ;  THIS IS INCREDIBLY GROSS
C944:              ;  BUT IS A RESULT OF MAKING VOLUME A SPECIAL
C944:              ;  CASE OF SEARCHING ALL DEVICES FOR
C944:              ;  A KNOWN VOLUME
C944:18                       CLC
C945:60                       RTS
C946:A5 3E         NOTLOG2    LDA   VCBENTRY         ;  REACH HERE IF REAL DUPLICATE VOLUME
C948:85 B6                    STA   VCBPTR           ;  RESOTRE FREE VCB PTR
C94A:18                       CLC
C94B:60                       RTS                    ;  DUPLICATE VOLUME PRETENDS TO BE NO ERROR
```

```
C94C:              692 *
C94C:A0 15         693 TSFRBLK    LDY   #VCBTFRE+1
C94E:B1 B6         694            LDA   (VCBPTR),Y       ; FIND OUT IF ENOUGH FREE BLOCKS
C950:88            695            DEY                    ; ARE AVAILABLE TO ACCOMODATE REQEST.
C951:11 B6         696            ORA   (VCBPTR),Y       ; BUT FIRST FIND OUT IF WE GOT A PROPER COUNT FOR THIS VOLUME.
C953:D0 5C   C9B1  697            BNE   CMPFREB          ; BRANCH IF COUNT IS NON-ZERO
C955:88            698            DEY                    ; IF ZERO, THEN COUNT MUST BE TAKEN
C956:B1 B6         699            LDA   (VCBPTR),Y       ;  GET HIGH TOTAL BLKS
C958:AA            700            TAX                    ;  SAVE IT
C959:88            701            DEY                    ;  GET LOW
C95A:B1 B6         702            LDA   (VCBPTR),Y       ;  TOTAL BLKS
C95C:D0 01   C95F  703            BNE   TSFR01
C95E:CA            704            DEX                    ;  ADJUST FOR BITMAP BLOCK BOUNDARY
C95F:8A            705 TSFR01     TXA
C960:4A            706            LSR   A                ; DIVIDE BY 16. THE RESULT IS THE NUMBER
C961:4A            707            LSR   A                ;  OF BIT MAPS TO BE SEARCHED.
C962:4A            708            LSR   A
C963:4A            709            LSR   A
C964:85 0D         710            STA   BMCNT            ; SAVE IT.
C966:A9 00         711            LDA   #0               ; START COUNT AT ZERO.
C968:8D E3 DB      712            STA   SCRTCH
C96B:8D E4 DB      713            STA   SCRTCH+1
C96E:A9 FF         714            LDA   #$FF             ; MARK 'FIRST FREE' TEMP AS UNKNOWN
C970:85 0C         715            STA   NOFREE
C972:A0 10         716            LDY   #VCBDEV          ; MAKE SURE BIT MAP IS UP TO DATE
C974:B1 B6         717            LDA   (VCBPTR),Y       ; GET DEVICE NUMBER
C976:AA            718            TAX                    ; PASS TO 'UPBMAP' IN X
C977:20 E4 CB      719            JSR   UPBMAP           ; (NOTHING HAPPENS IF IT DON'T HAFTA.)
C97A:B0 46   C9C2  720            BCS   TFBERR           ; BRANCH IF WE GOT TROUBLE,
C97C:A0 1A         721            LDY   #VCBDMAP         ; GET ADDRESS OF FIRST BIT MAP.
C97E:B1 B6         722            LDA   (VCBPTR),Y
C980:85 C6         723            STA   BLOKNML
C982:C8            724            INY                    ; (FOR HIGH ADDRESS)
C983:B1 B6         725            LDA   (VCBPTR),Y
C985:85 C7         726            STA   BLOKNMH
C987:20 58 CC      727 BMAPRD     JSR   RDGBUF           ; USE G(ENERAL)BUFF(ER) FOR TEMPORARY
C98A:B0 36   C9C2  728            BCS   TFBERR           ;  SPACE TO COUNT FREE BLOCKS (BITS)
C98C:20 C3 C9      729            JSR   COUNT            ; GO COUNT EM
C98F:C6 0D         730            DEC   BMCNT            ; WAS THAT THE LAST BIT MAP?
C991:30 09   C99C  731            BMI   CHGVCB           ; IF SO, GO CHANGE FCB TO AVOID DOING THIS AGAIN!
C993:E6 C6         732            INC   BLOKNML          ; NOTE: THE ORGANIZATION OF THE BIT MAPS
C995:D0 F0   C987  733            BNE   BMAPRD           ;  ARE CONTIGUOUS FOR SOS VERSION 0
C997:E6 C7         734            INC   BLOKNMH          ;  IF SOME OTHER ORGANIZATION IS IMPLEMENTED, THIS CODE
C999:4C 87 C9      735            JMP   BMAPRD           ;  MUST BE CHANGED!
```

```
C99C:              *
C99C:A0 1C         CHGVCB     LDY   #VCBCMAP         ; MARK WHICH BLOCK HAD FIRST FREE SPACE
C99E:A5 0C                    LDA   NOFREE
C9A0:30 1D   C9BF             BMI   DSKFULL          ; BRANCH IF NO FREE SPACE WAS FOUND.
C9A2:91 B6                    STA   (VCBPTR),Y
C9A4:A0 15                    LDY   #VCBTFRE+1       ; UPDATE THE FREE COUNT.
C9A6:AD E4 DB                 LDA   SCRTCH+1         ; GET HIGH COUNT BYTE
C9A9:91 B6                    STA   (VCBPTR),Y       ; UPDATE VOLUME CONTROL BLOCK.
C9AB:88                       DEY
C9AC:AD E3 DB                 LDA   SCRTCH
C9AF:91 B6                    STA   (VCBPTR),Y       ; AND LOW BYTE TOO...
C9B1:B1 B6         CMPFREB    LDA   (VCBPTR),Y       ; COMPARE TOTAL AVAILABLE
C9B3:38                       SEC
C9B4:E5 04                    SBC   REQL             ;  FREE BLOCKS ON THIS VOLUME.
C9B6:C8                       INY
C9B7:B1 B6                    LDA   (VCBPTR),Y
C9B9:E5 05                    SBC   REQH
C9BB:90 02   C9BF             BCC   DSKFULL
C9BD:18                       CLC
C9BE:60                       RTS
C9BF:A9 00         DSKFULL    LDA   #OVRERR
C9C1:38                       SEC
C9C2:60            TFBERR     RTS
```

```
C9C3:             *
C9C3:A0 00        COUNT      LDY   #0                ; BEGIN AT THE BEGINNING.
C9C5:B9 00 12     FRCONT     LDA   GBUF,Y            ; GET BIT PATTERN
C9C8:F0 03   C9CD            BEQ   FRCNT1            ; DON'T BOTHER COUNTING NOTHIN'
C9CA:20 F5 C9                JSR   CNTFREE
C9CD:B9 00 13     FRCNT1     LDA   GBUF+$100,Y       ; DO BOTH PAGES WITH SAME LOOP
C9D0:F0 03   C9D5            BEQ   FRCNT2
C9D2:20 F5 C9                JSR   CNTFREE
C9D5:C8           FRCNT2     INY
C9D6:D0 ED   C9C5            BNE   FRCONT            ; LOOP TILL ALL 512 BYTES COUNTED
C9D8:24 0C                   BIT   NOFREE            ; HAS FIRST BLOCK WITH FREE SPACE BEEN FOUND YET?
C9DA:10 18   C9F4            BPL   FRCNT3            ; BRANCH IF IT HAS.
C9DC:AD E3 DB                LDA   SCRTCH            ; TEST TO SEE IF ANY BLOCKS WERE COUNTED
C9DF:0D E4 DB                ORA   SCRTCH+1
C9E2:F0 10   C9F4            BEQ   FRCNT3            ; BRANCH IF NONE COUNTED.
C9E4:A0 13                   LDY   #VCBTBLK+1
C9E6:B1 B6                   LDA   (VCBPTR),Y        ; SHOW THIS MAP IS FIRST WITH FREE SPACE
C9E8:38                      SEC                     ;  CORRECT FOR EXACT MULTIPLES OF $1000
C9E9:E9 01                   SBC   #$01
C9EB:4A                      LSR   A
C9EC:4A                      LSR   A
C9ED:4A                      LSR   A
C9EE:4A                      LSR   A
C9EF:38                      SEC                     ; SUBTRACT COUNTDOWN FROM TOTAL BIT MAPS
C9F0:E5 0D                   SBC   BMCNT
C9F2:85 0C                   STA   NOFREE
C9F4:60           FRCNT3     RTS
C9F5:             *
C9F5:0A           CNTFREE    ASL   A                 ; COUNT THE NUMBER OF BITS IN THIS BYTE.
C9F6:90 08   CA00            BCC   CFREE1
C9F8:EE E3 DB                INC   SCRTCH
C9FB:D0 03   CA00            BNE   CFREE1
C9FD:EE E4 DB                INC   SCRTCH+1
CA00:AA           CFREE1     TAX
CA01:D0 F2   C9F5            BNE   CNTFREE           ; LOOP UNTIL ALL BITS COUNTED.
CA03:60                      RTS
CA04:                        CHN   ALLOC
CA04:             *
CA04:86 0D        DEALLOC    STX   BMCNT             ; SAVE HIGH ORDER ADDRESS OF BLOCK TO BE FREED.
CA06:48                      PHA                     ; SAVE IT
CA07:A6 B6                   LDX   VCBPTR            ; WHILE THE BITMAP
CA09:BD 13 11                LDA   VCB+VCBTBLK+1,X   ; DISK ADDRESS IS CHECKED
CA0C:C5 0D                   CMP   BMCNT             ; TO SEE IF IT MAKES SENSE
CA0E:68                      PLA                     ; RESTORE
CA0F:90 51   CA62            BCC   DEALERR1          ; BRANCH IF IMPOSSIBLE
CA11:AA                      TAX
CA12:29 07                   AND   #$7               ; GET THE BIT TO BE OR-ED IN.
CA14:A8                      TAY
CA15:B9 66 CA                LDA   WHICHBIT,Y        ; (SHIFTING TAKES 7 BYTES, BUT IS SLOWER)
CA18:85 0C                   STA   NOFREE            ; SAVE BIT PATTERN
CA1A:8A                      TXA                     ; GET LOW BLOCK ADDRESS AGAIN.
CA1B:46 0D                   LSR   BMCNT
CA1D:6A                      ROR   A                 ; GET POINTER TO BYTE IN BITMAP THAT REPRESENTS
CA1E:46 0D                   LSR   BMCNT             ; THE BLOCK ADDRESS.
CA20:6A                      ROR   A
CA21:46 0D                   LSR   BMCNT
```

```
CA23:6A                         ROR   A
CA24:85 17                      STA   BMPTR            ; SAVE POINTER.
CA26:46 0D                      LSR   BMCNT            ; NOW TRANSFER BIT WHICH SPECIFIES WHICH PAGE OF BITMAP.
CA28:26 19                      ROL   HALF
CA2A:A6 1A                      LDX   BMTAB            ; (THIS POINTS TO THE TABLE FOR THE BITMAP BUFFER USED).
CA2C:B5 21                      LDA   BMACMAP,X        ; WHAT IS THE CURRENT MAP
CA2E:C5 0D                      CMP   BMCNT            ; IS IN CORE BIT MAP THE ONE WE WANT?
CA30:F0 14   CA46               BEQ   DEALL1           ; BRANCH IF IN-CORE IS CORRECT.
CA32:20 65 D7                   JSR   BMAPUP           ; PUT CURRENT MAP AWAY.
CA35:B0 2A   CA61               BCS   DEALERR          ; PASS BACK ANY ERROR.
CA37:A5 0D                      LDA   BMCNT            ; GET DESIRED MAP NUMBER.
CA39:A0 1C                      LDY   #VCBCMAP
CA3B:91 B6                      STA   (VCBPTR),Y       ; AND MAKE IT CURRENT.
CA3D:A6 1A                      LDX   BMTAB
CA3F:B5 1D                      LDA   BMADEV,X
CA41:20 10 CC                   JSR   GTBMAP           ; READ IT INTO THE BUFFER,
CA44:B0 1B   CA61               BCS   DEALERR
CA46:A4 17          DEALL1      LDY   BMPTR            ; INDEX TO BYTE.
CA48:46 19                      LSR   HALF
CA4A:90 02   CA4E               BCC   DEALL2           ; BRANCH IF ON PAGE ONE OF BITMAP.
CA4C:E6 B9                      INC   BMADR+1
CA4E:A5 0C          DEALL2      LDA   NOFREE           ; THE INDIVIDUAL BIT.
CA50:11 B8                      ORA   (BMADR),Y
CA52:91 B8                      STA   (BMADR),Y
CA54:90 02   CA58               BCC   DEALL3           ; BRANCH IF ADDRESS IS PROPER
CA56:C6 B9                      DEC   BMADR+1
CA58:A6 1A          DEALL3      LDX   BMTAB            ; MARK BITMAP AS MODIFIED.
CA5A:A9 80                      LDA   #$80
CA5C:15 1C                      ORA   BMASTAT,X
CA5E:95 1C                      STA   BMASTAT,X
CA60:18                         CLC
CA61:60             DEALERR     RTS
CA62:A9 00          DEALERR1    LDA   #BITMAPADR       ; BIT MAP BLOCK NUMBER IMPOSSIBLE
CA64:38                         SEC                    ; SAY BIT MAP DISK ADDRESS WRONG (PROBABLY DATA MASQUERADING AS
INDEX BLOCK)
CA65:60                         RTS
CA66:               *
CA66:80 40 20 10    WHICHBIT    DFB   $80,$40,$20,$10
CA6A:08 04 02 01                DFB   8,4,2,1
CA6E:               *
CA6E:               *
```

```
CA6E:               *
CA6E:A9 00          ALCIDXS    LDA   #0               ; ALLOCATION OF THE INDEXES ALWAYS FILLS IN
CA70:85 0E                     STA   SAPTR            ; STARTING AT THE BEGINNING OF THE BLOCK.
CA72:20 9C CA                  JSR   ALC1BLK          ; THIS GETS FIRST INDEX AND SETS UP A
CA75:B0 1E   CA95              BCS   ERRALC1          ; POINTER TO THE FREE BLOCKS (TO AVOID
CA77:A4 0E          ALIDX1     LDY   SAPTR            ; SCANNING THE WHOLE BLOCK EVERY TIME).
CA79:91 B2                     STA   (TINDX),Y        ; SAVE INDEX BLOCK ADDRESS (LOW)
CA7B:E6 B3                     INC   TINDX+1
CA7D:AD E4 DB                  LDA   SCRTCH+1         ; GET HIGH BYTE OF ADDRESS
CA80:91 B2                     STA   (TINDX),Y        ; (AND SAVE IT)
CA82:C6 B3                     DEC   TINDX+1
CA84:C6 04                     DEC   REQL             ; HAS REQUEST BEEN SATIFIED?
CA86:F0 13   CA9B              BEQ   ALDXEND          ; (CARRY IS CLEAR)
CA88:E6 0E                     INC   SAPTR            ; BUMP INDEX POINTER
CA8A:A4 17                     LDY   BMPTR            ; GET INDEX POINTER TO LAST ACCESSED BIT GROUP
CA8C:A5 19                     LDA   HALF             ; WHICH HALF OF MAP? (BOTH BMPTR & HALF SET UP BY 'ALC1BLK')
CA8E:D0 06   CA96              BNE   SECNDHAF
CA90:20 A5 CA                  JSR   GETBITS1         ; GET NEXT FREE BLOCK ADDRESS.
CA93:90 E2   CA77              BCC   ALIDX1           ; BRANCH IF NO ERROR
CA95:60             ERRALC1    RTS
CA96:               *
CA96:20 B2 CA       SECNDHAF   JSR   GETBITS2         ; GET NEXT FREE BLOCK ADDRESS FROM SECOND HALF OF BIT MAP
CA99:90 DC   CA77              BCC   ALIDX1           ; BRANCH IF NO ERROR.
CA9B:60             ALDXEND    RTS                    ; RETURN STATUS (CARRY SET INDICATES ERROR)
CA9C:               *
CA9C:               *
CA9C:20 7F CB       ALC1BLK    JSR   FNDBMAP          ; GET ADDRESS OF BIT MAP IN 'BMADR'
CA9F:B0 F4   CA95              BCS   ERRALC1          ; BRANCH IF ERROR ENCOUNTERED
CAA1:A0 00          SRCHFRE    LDY   #0               ; START SEARCH AT BEGINNING OF BIT MAP BLOCK
CAA3:84 19                     STY   HALF             ; INDICATE WHICH HALF (PAGE) WE'RE SEARCHING.
CAA5:B1 B8          GETBITS1   LDA   (BMADR),Y
CAA7:D0 1A   CAC3              BNE   BITFOUND         ; FREE BLOCKS ARE INDICATED BY 'ON' BITS
CAA9:C8                        INY
CAAA:D0 F9   CAA5              BNE   GETBITS1         ; CHECK ALL OF 'EM IN FIRST PAGE.
CAAC:E6 B9                     INC   BMADR+1          ; BUMP HIGH ADDRESS OF CURRENT BITMAP
CAAE:E6 19                     INC   HALF             ; INDICATE SEARCH HAS PROGRESSED TO PAGE 2
CAB0:E6 18                     INC   BASVAL           ; BASE VALUE= BASE ADDRESS/2048
CAB2:B1 B8          GETBITS2   LDA   (BMADR),Y        ; SEARCH SECOND HALF FOR FREE BLOCK
CAB4:D0 0D   CAC3              BNE   BITFOUND
CAB6:C8                        INY
CAB7:D0 F9   CAB2              BNE   GETBITS2
CAB9:C6 B9                     DEC   BMADR+1          ; RESET BIT MAP ADDRESS TO BEGINNING.
CABB:E6 18                     INC   BASVAL           ; ADD 2048 OFFSET FOR NEXT PAGE
CABD:20 57 CB                  JSR   NXTBMAP          ; GET NEXT BITMAP (IF IT EXISTS) AND UPDATE VCB.
CAC0:90 DF   CAA1              BCC   SRCHFRE          ; BRANCH IF NO ERROR ENCOUNTERED.
CAC2:60                        RTS                    ; RETURN ERROR.
```

```
CAC3:               *
CAC3:84 17          BITFOUND   STY   BMPTR            ; SAVE INDX POINTER TO VALID BIT GROUP
CAC5:A5 18                     LDA   BASVAL           ; SET UP FOR BLOCK ADDRESS CALCULATION
CAC7:8D E4 DB                  STA   SCRTCH+1
CACA:98                        TYA                    ; GET ADDRESS OF BIT PATTERN
CACB:0A                        ASL   A                ; MULTIPLY THIS AND BASVAL BY 8
CACC:2E E4 DB                  ROL   SCRTCH+1
CACF:0A                        ASL   A
CAD0:2E E4 DB                  ROL   SCRTCH+1
CAD3:0A                        ASL   A
CAD4:2E E4 DB                  ROL   SCRTCH+1
CAD7:AA                        TAX                    ; NOW X= LOW ADDRESS WITHIN 7 OF ACTUAL ADDRESS.
CAD8:B1 B8                     LDA   (BMADR),Y        ; GET BIT PATTERN AGAIN
CADA:38                        SEC                    ; MARK RIGHT END OF BYTE.
CADB:2A             ADCALC     ROL   A                ; FIND LEFT MOST 'ON' BIT
CADC:B0 03   CAE1              BCS   BOUNCE           ; BRANCH IF FOUND.
CADE:E8                        INX                    ; ADJUST LOW ADDRESS
CADF:D0 FA   CADB              BNE   ADCALC           ; BRANCH ALWAYS
CAE1:4A             BOUNCE     LSR   A                ; RESTORE ALL BUT LEFT MOST BIT TO ORIGINAL POSITION
CAE2:90 FD   CAE1              BCC   BOUNCE           ; LOOP UNTIL MARK (SET ABOVE) MOVES INTO CARRY
CAE4:91 B8                     STA   (BMADR),Y        ; UPDATE BITMAP TO SHOW ALLOCATED BLOCK IN USE.
CAE6:8E E3 DB                  STX   SCRTCH           ; SAVE LOW ADDRESS.
CAE9:A6 1A                     LDX   BMTAB            ; UPDATE BIT MAP BUFFER STATUS
CAEB:A9 80                     LDA   #$80             ; INDICATE MAP HAS BEEN MODIFIED
CAED:15 1C                     ORA   BMASTAT,X        ; (X IS EITHER 0 OR 6 FOR
CAEF:95 1C                     STA   BMASTAT,X        ; BUFFER 'A' OR 'B' RESPECTIVELY.)
CAF1:A0 14                     LDY   #VCBTFRE         ; SUBTRACT 1 FROM TOTAL FREE
CAF3:B1 B6                     LDA   (VCBPTR),Y       ; BLOCKS IN VCB TO ACCOUNT FOR NEWLY
CAF5:E9 01                     SBC   #1               ; ALLOCATED BLOCK (CARRY IS SET FROM 'BOUNCE')
CAF7:91 B6                     STA   (VCBPTR),Y
CAF9:B0 07   CB02              BCS   RET1BLK          ; BRANCH IF HI FREE COUNT DOESN'T NEED ADJUSTMENT.
CAFB:C8                        INY
CAFC:B1 B6                     LDA   (VCBPTR),Y       ; ADJUST HIGH COUNT.
CAFE:E9 00                     SBC   #0               ; (CARRY IS CLEAR, SO ACC=ACC-1)
CB00:91 B6                     STA   (VCBPTR),Y
CB02:18             RET1BLK    CLC                    ; INDICATE NO ERROR ENCOUNTERED
CB03:AD E3 DB                  LDA   SCRTCH           ; GET ADDRESS LOW IN ACC.
CB06:AC E4 DB                  LDY   SCRTCH+1         ; AND HIGH ADDRESS IN Y
CB09:60                        RTS                    ; RETURN ADDRESS OF NEWLY ALLOCATED BLOCK.
CB0A:               *
```

```
CB0A:              *
CB0A:A0 10         GTTINDX    LDY   #VCBDEV          ; GET DEVICE NUMBER SO WE DON'T
CB0C:A2 00                    LDX   #0               ; ANTICPATE USING BUFFER 'A'.
CB0E:B1 B6                    LDA   (VCBPTR),Y       ; USE THE BUFFER USED BY IT!
CB10:C5 1D                    CMP   BMADEV           ; IS IT IN BUFFER 'A'?
CB12:F0 0A   CB1E             BEQ   FREEBE           ; IF SO, FREE 'B'!
CB14:C5 23                    CMP   BMBDEV           ; IF NOT, IS IT IN 'B'?
CB16:F0 08   CB20             BEQ   FREEA            ; IF SO, FREE UP BUFFER 'A'
CB18:20 7F CB                 JSR   FNDBMAP          ; OTHERWISE, FORCE ALLOCATION FOR ONE OF THE BUFFERS
CB1B:90 ED   CB0A             BCC   GTTINDX          ; NOW TRY AGAIN.
CB1D:60                       RTS                    ; RETURN ERROR.
CB1E:              *
CB1E:A2 06         FREEBE     LDX   #BMTABSZ         ; DE-ALLOCATE BUFFER IF NECESSARY
CB20:86 0C         FREEA      STX   NOFREE           ; SAVE WHICH BUFFER WE'RE LOOKIN AT.
CB22:B4 1C                    LDY   BMASTAT,X        ; DO WE NEED TO WRITE BUFFER TO FREE IT?
CB24:10 0D   CB33             BPL   USEBUF           ; NO, THEN USE IT.
CB26:86 3D                    STX   ZPGTEMP          ; SAVE BM BUFFER ID FOR A BIT
CB28:20 4F CC                 JSR   WRTBMAP          ; WRITE BM TO OWNING UNIT
CB2B:B0 29   CB56             BCS   SOMERR1          ; RETURN ANY ERROR (W/O RELEASING BM)
CB2D:A6 3D                    LDX   ZPGTEMP          ; FETCH THE BM BUFFER ID
CB2F:A9 00                    LDA   #0
CB31:95 1C                    STA   BMASTAT,X        ; AND MARK BM BUFFER AS FREE
CB33:A6 0C         USEBUF     LDX   NOFREE           ; GET INDEX TO BUFFER INFO
CB35:A9 00                    LDA   #0               ; MARK STATUS OF BUFFER AS FREE.
CB37:95 1D                    STA   BMADEV,X         ; (DEVICE 0 IS NOT ANY DEVICE)
CB39:85 B2                    STA   TINDX
CB3B:85 B8                    STA   BMADR
CB3D:BD 1E 00                 LDA   BMAMADR,X        ; GET MEMORY ADDRESS OF FREE BUFFER.
CB40:85 B3                    STA   TINDX+1
CB42:8A                       TXA                    ; SET UP PROPER HI ADDRESS OF BIT MAP TOO...
CB43:49 06                    EOR   #BMTABSZ         ; SELECT ALTERNATE BIT MAP TABLE.
CB45:85 1A                    STA   BMTAB            ; (TO INDICATE WHICH IS BITMAP)
CB47:AA                       TAX
CB48:BD 1E 00                 LDA   BMAMADR,X        ; GET HIGH ADDRESS OF BIT MAP.
CB4B:85 B9                    STA   BMADR+1
CB4D:A5 1B                    LDA   BMBUFBNK         ; AND BANK PAIR NUMBER.
CB4F:8D B3 14                 STA   SSTIDXH
CB52:8D B9 14                 STA   SISBMADR
CB55:18                       CLC                    ; INDICATE NO ERRORS
CB56:60            SOMERR1    RTS
CB57:              *
```

```
CB57:A0 13         NXTBMAP    LDY   #VCBTBLK+1       ; BEFORE BUMPING TO NEXT MAP,
CB59:B1 B6                    LDA   (VCBPTR),Y       ; CHECK TO BE SURE THERE IS
CB5B:4A                       LSR   A                ; INDEED A NEXT MAP!
CB5C:4A                       LSR   A
CB5D:4A                       LSR   A
CB5E:4A                       LSR   A
CB5F:A0 1C                    LDY   #VCBCMAP
CB61:D1 B6                    CMP   (VCBPTR),Y       ; ARE THERE MORE MAPS?
CB63:F0 51   CBB6             BEQ   NOMORBIT         ; BRANCH IF NO MORE TO LOOK AT.
CB65:B1 B6                    LDA   (VCBPTR),Y       ; ADD 1 TO CURRENT MAP
CB67:18                       CLC
CB68:69 01                    ADC   #1
CB6A:91 B6                    STA   (VCBPTR),Y
CB6C:A0 10                    LDY   #VCBDEV
CB6E:B1 B6                    LDA   (VCBPTR),Y
CB70:AA                       TAX                    ; GO WRITE OUT LAST MAP IF NECESSARY
CB71:20 E4 CB                 JSR   UPBMAP
CB74:4C 7F CB                 JMP   FNDBMAP          ; READ NEXT BIT MAP INTO BUFFER
CB77:              *
CB77:A2 00         GETA.BUF   LDX   #0
CB79:F0 0E   CB89             BEQ   FRESHMAP
CB7B:              *
CB7B:A2 06         GETB.BUF   LDX   #BMTABSZ
CB7D:D0 0A   CB89             BNE   FRESHMAP         ; BRANCH ALWAYS
CB7F:              *
CB7F:              *
CB7F:A0 10         FNDBMAP    LDY   #VCBDEV          ; GET DEVICE NUMBER
CB81:B1 B6                    LDA   (VCBPTR),Y
CB83:A2 00                    LDX   #0               ; START WITH MAP 'A'
CB85:D5 1D         FNDMAP1    CMP   BMADEV,X
CB87:D0 0C   CB95             BNE   TRYMAP2
CB89:86 1A         FRESHMAP   STX   BMTAB            ; SAVE POINTER TO BIT MAP INFO TABLE
CB8B:B4 1C                    LDY   BMASTAT,X        ; IS THIS ONE ALREADY MODIFIED?
CB8D:30 0E   CB9D             BMI   BMFOUND          ; YES, RETURN POINTER IN 'BMADR'
CB8F:20 10 CC                 JSR   GTBMAP           ; OTHERWISE READ IN FRESH BIT MAP
CB92:90 09   CB9D             BCC   BMFOUND          ; BRANCH IF SUCCESSFUL.
CB94:60                       RTS                    ; OTHERWISE, RETURN ERROR.
CB95:              *
CB95:CA            TRYMAP2    DEX                    ; WAS LAST FAILURE MAP 'A'
CB96:10 22   CBBA             BPL   FRBMBUF          ; NO, MUST FREE UP ONE OF THE BUFFERS
CB98:A2 06                    LDX   #BMTABSZ         ; TRY BIT MAP BUFFER 'B'.
CB9A:4C 85 CB                 JMP   FNDMAP1
```

```
CB9D:               *
CB9D:A6 1A          BMFOUND    LDX   BMTAB            ; WHICH TABLE?
CB9F:A0 1C                     LDY   #VCBCMAP
CBA1:B1 B6                     LDA   (VCBPTR),Y
CBA3:0A                        ASL   A
CBA4:85 18                     STA   BASVAL
CBA6:BD 1E 00                  LDA   BMAMADR,X        ; GET HIGH ADDRESS
CBA9:85 B9                     STA   BMADR+1
CBAB:A5 1B                     LDA   BMBUFBNK         ; GET BANK NUMBER OF BUFFER BIT MAP BUFFERS
CBAD:8D B9 14                  STA   SISBMADR
CBB0:A9 00                     LDA   #0               ; BUFFERS ALWAYS FALL ON A PAGE BOUNDARY
CBB2:85 B8                     STA   BMADR
CBB4:18                        CLC                    ; INDICATE ALL IS VALID AND GOOD!
CBB5:60                        RTS
CBB6:               *
CBB6:A9 00          NOMORBIT   LDA   #OVRERR          ; INDICATE REQUEST CAN'T BE FILLED.
CBB8:38                        SEC                    ; INDICATE ERROR
CBB9:60                        RTS
CBBA:               *
CBBA:38             FRBMBUF    SEC
CBBB:A6 1A                     LDX   BMTAB            ; FIND OUT WHICH WAS LAST USED.
CBBD:F0 05   CBC4              BEQ   CHKBMB           ; IF 'A' WAS USED CHECK 'B' FIRST
CBBF:18                        CLC                    ; INDICATE 'A' IS CHECKED FIRST
CBC0:24 1C                     BIT   BMASTAT          ; IS BUFFER 'A' FREE (UNMODIFIED)?
CBC2:10 B3   CB77              BPL   GETA.BUF         ; YES, USE IT.
CBC4:24 22          CHKBMB     BIT   BMBSTAT          ; IS BUFFER 'B' FREE?
CBC6:90 06   CBCE              BCC   FREBUF1          ; BRANCH IF BOTH ARE USED
CBC8:10 B1   CB7B              BPL   GETB.BUF         ; YES...
CBCA:24 1C                     BIT   BMASTAT          ; (CHECK 'A')
CBCC:10 A9   CB77              BPL   GETA.BUF
CBCE:A2 00          FREBUF1    LDX   #0
CBD0:90 02   CBD4              BCC   FREBUFA          ; BRANCH IF BUFFER 'A' HAS LEAST PRIORITY.
CBD2:A2 06                     LDX   #BMTABSZ
CBD4:86 3D          FREBUFA    STX   ZPGTEMP          ; SAVE BM BUFF ID FOR A BIT
CBD6:20 4F CC                  JSR   WRTBMAP          ; XREG PASSES BM BUFF ID
CBD9:B0 08   CBE3              BCS   NOGO             ; ERROR ENCOUNTERED ON WRITING
CBDB:A6 3D                     LDX   ZPGTEMP          ; FETCH BM BUFF ID
CBDD:A9 00                     LDA   #0
CBDF:95 1C                     STA   BMASTAT,X        ; AND MARK BM BUFFER AS FREE
CBE1:90 9C   CB7F              BCC   FNDBMAP          ; LOOK AGAIN FOR FRRE BIT MAP BUFFER SPACE
CBE3:60             NOGO       RTS                    ; RETURN ERROR ON WRITING BM
CBE4:               *
CBE4:E4 1D          UPBMAP     CPX   BMADEV           ; UPDATE BIT MAP OF DEVICE X
CBE6:D0 06   CBEE              BNE   UPBM1
CBE8:18                        CLC                    ; FREE BUFFER 'A' IF NEEDED.
CBE9:24 1C                     BIT   BMASTAT
CBEB:30 E1   CBCE              BMI   FREBUF1          ; (CARRY CLEAR FOR BUFFER 'A')
CBED:60                        RTS
```

```
CBEE:              *
CBEE:E4 23         UPBM1      CPX   BMBDEV
CBF0:D0 04   CBF6             BNE   NOUPDAT          ; DON'T UPDATE IF NOT NECESSARY.
CBF2:24 22                    BIT   BMBSTAT
CBF4:30 D8   CBCE             BMI   FREBUF1          ; (CARRY IS SET)
CBF6:18            NOUPDAT    CLC
CBF7:60                       RTS                    ; RETURN 'NO ERROR'
CBF8:              *
CBF8:        CBF8  CLEARBMS   EQU   *                ; MAKE SURE ALL BIT MAPS ASSOCIATED
CBF8:              * WITH A DEVICE ARE MARKED INVALID
CBF8:              * IF A NEW VOLUME IS LOGGED IN ON IT.
CBF8:              * INPUT ARG: A REG = DEVNUM
CBF8:              * X REG PRESERVED
CBF8:A0 00                    LDY   #0
CBFA:C5 1D                    CMP   BMADEV
CBFC:D0 07   CC05             BNE   CLRBM1           ; BRANCH IF BIT MAP A NOT OWNED
CBFE:24 1C                    BIT   BMASTAT
CC00:30 02   CC04             BMI   CLRBM2           ; BRANCH IF BITMAP A BUSY
CC02:84 1D                    STY   BMADEV           ; ELSE, CLEAR IT
CC04:60            CLRBM2     RTS                    ; NEED ONLY CLEAR ONE
CC05:C5 23         CLRBM1     CMP   BMBDEV           ; BIT MAP B?
CC07:D0 FB   CC04             BNE   CLRBM2           ; BRANCH IF BIT MAP B NOT OWNED BY DEVNUM
CC09:24 22                    BIT   BMBSTAT
CC0B:30 F7   CC04             BMI   CLRBM2           ; BRANCH IF BITMAP B BUSY
CC0D:84 23                    STY   BMBDEV           ; ELSE CLEAR IT
CC0F:60                       RTS                    ; AND RETURN TO CALLER (NO ERRORS)
CC10:              *
CC10:95 1D         GTBMAP     STA   BMADEV,X         ; SAVE ACC AS CURRENT DEVICE FOR BUFFER
CC12:BD 1E 00                 LDA   BMAMADR,X        ; GET HIGH ORDER ADDRESS OF BUFFER
CC15:85 B9                    STA   BMADR+1          ; SELECTED BY X
CC17:A5 1B                    LDA   BMBUFBNK         ; AND GET BANK PAIR NUMBER
CC19:8D B9 14                 STA   SISBMADR         ; OF BOTH BIT MAP BUFFERS 'A' AND 'B'
CC1C:A0 1C                    LDY   #VCBCMAP         ; GET LOWEST MAP NUMBER WITH FREE BLOCKS IN IT.
CC1E:B1 B6                    LDA   (VCBPTR),Y
CC20:95 21                    STA   BMACMAP,X        ; ASSOCIATE THE OFFSET WITH THE BITMAP CONTROL BLOCK
CC22:18                       CLC
CC23:A0 1A                    LDY   #VCBDMAP         ; ADD THIS NUMBER TO THE BASE
CC25:71 B6                    ADC   (VCBPTR),Y       ; ADDRESS OF FIRST BIT MAP
CC27:95 1F                    STA   BMADADR,X        ; SAVE LOW ADDRESS OF BIT MAP TO BE USED.
CC29:C8                       INY                    ; NOW GET HIGH DISK ADDRESS OF MAP
CC2A:B1 B6                    LDA   (VCBPTR),Y       ; ADD TO THIS THE STATE OF THE CARRY
CC2C:69 00                    ADC   #0
CC2E:95 20                    STA   BMADADR+1,X      ; SAVE HIGH DISK ADDRESS TOO.
CC30:              ; DROP INTO 'RDBMAP'
CC30:              *
```

```
CC30:              *
CC30:A9 00                    LDA   #RDCMD           ; (X CONTAINS AN INDEX TO DETERMINE WHICH BUFFER)
CC32:85 C0         DOBMAP     STA   DHPCMD           ; SAVE DEVICE COMMAND
CC34:A5 35                    LDA   DEVNUM           ; FIX THE 'BIT MAP TRASH BUG'
CC36:48                       PHA                    ; BY NOT MUNGING DEVNUM
CC37:B5 1D                    LDA   BMADEV,X         ; GET DEVICE NUMBER.
CC39:85 35                    STA   DEVNUM
CC3B:B5 1F                    LDA   BMADADR,X        ; AND MAP'S DISK ADDRESS
CC3D:85 C6                    STA   BLOKNML
CC3F:B5 20                    LDA   BMADADR+1,X
CC41:85 C7                    STA   BLOKNMH
CC43:BD 1E 00                 LDA   BMAMADR,X        ; LASTLY GET THE ADDRESS OF THE BUFFER
CC46:A6 1B                    LDX   BMBUFBNK         ; AND BANK NUMBER.
CC48:20 6A CC                 JSR   DOBITMAP         ; (NOTE: LOW ADDRESS IS FIXED TO ZERO AS THIS IS A BUFFER)
CC4B:68                       PLA                    ; RESTORE
CC4C:85 35                    STA   DEVNUM           ; THE DEVNUM WE CAME IN WITH!
CC4E:60                       RTS
CC4F:              *
CC4F:A9 01         WRTBMAP    LDA   #WRTCMD          ; WRITE BIT MAP POINTED TO BY X
CC51:4C 32 CC                 JMP   DOBMAP
CC54:              *
CC54:A9 01         WRTGBUF    LDA   #WRTCMD          ; SET CALL FOR WRITE.
CC56:D0 02   CC5A             BNE   SVGCMD           ; BRANCH ALWAYS.
CC58:A9 00         RDGBUF     LDA   #RDCMD           ; SET CALL FOR READ.
CC5A:85 C0         SVGCMD     STA   DHPCMD           ; PASSED TO DEVICE HANDLER.
CC5C:A5 C6                    LDA   BLOKNML          ; SAVE CURRENT
CC5E:8D 76 CC                 STA   TTLINK           ;   GBUF BLOCK
CC61:A5 C7                    LDA   BLOKNMH          ; ADDRESS
CC63:8D 77 CC                 STA   TTLINK+1         ; FOR DIRECTORY EXTEND
CC66:A9 12                    LDA   #GBUF/256        ; GET HIGH ADDRESS OF GENERAL BUFFER
CC68:A2 00                    LDX   #0               ; TO FORCE ACCESS TO NON BANK MEMORY.
CC6A:        CC6A  DOBITMAP   EQU   *
CC6A:85 C3         DOIDX      STA   DBUFPH
CC6C:8E C3 14                 STX   SISBPH           ; SELECT BANK
CC6F:A9 00                    LDA   #0               ; GENERAL PURPOSE BUFFERS ALWAYS
CC71:85 C2                    STA   DBUFPL           ; START ON A PAGE BOUNDARY.
CC73:4C 25 CF                 JMP   FILEIO2          ; END VIA DEVICE DISPATCHER.
CC76:              *
CC76:        0002  TTLINK     DS    2                ; GBUF CURRENT ADDRESS
CC78:              *
CC78:A9 01         WRTINDX    LDA   #WRTCMD
CC7A:A6 02                    LDX   IDXADRL          ; GET BLOCK ADDRESS OF INDEX BLOCK
CC7C:A4 03                    LDY   IDXADRH
CC7E:85 C0         DOFRST     STA   DHPCMD           ; (ENTRY USED BY RD/WRTDFRST)
CC80:86 C6                    STX   BLOKNML
CC82:84 C7                    STY   BLOKNMH
CC84:A5 B3                    LDA   TINDX+1          ; HIGH RAM ADDRESS OF INDEX BLOCK
CC86:AE B3 14                 LDX   SSTIDXH          ; AND BANK NUMBER.
CC89:4C 6A CC                 JMP   DOIDX            ; AND GO DO REQUESTED OPERATION.
CC8C:              *
CC8C:A9 01         WRTDFRST   LDA   #WRTCMD          ; WRITE FILE'S FIRST BLOCK (USED
CC8E:D0 02   CC92             BNE   FADDR            ; BY CREATE, SO ADDRESS IN 'D.' STUFF).
CC90:A9 00         RDFRST     LDA   #RDCMD
CC92:AE CB DB      FADDR      LDX   DFIL+D.FRST      ; (BUFFER ADDRESS IS IN 'TINDX')
CC95:AC CC DB                 LDY   DFIL+D.FRST+1
CC98:4C 7E CC                 JMP   DOFRST
```

```
CC9B:              385 *
CC9B:              386            CHN   POSN.OPEN
```

```
CC9B:A0 12         GETMARK    LDY   #FCBMARK         ; MOVE CURRENT POSITION MARKER TO
CC9D:B1 BA         GMARK1     LDA   (FCBPTR),Y       ; USER'S 4 BYTE BUFFER POINTED TO BY
CC9F:48                       PHA                    ; C.MRKPTR IN SOS ZPAGE
CCA0:C8                       INY
CCA1:C0 15                    CPY   #FCBMARK+3       ; USE STACK AS TEMPORARY STORAGE FOR THREE BYTE
CCA3:D0 F8   CC9D             BNE   GMARK1           ; POSITION VALUE.
CCA5:A9 00                    LDA   #0               ; THE FOURTH (HIGHEST ORDER) BYTE IS ALWAYS ZERO.
CCA7:A0 03                    LDY   #3
CCA9:48                       PHA
CCAA:68            MOVMRK     PLA
CCAB:91 A2                    STA   (C.MRKPTR),Y     ; MOVE TO USER'S SPACE
CCAD:88                       DEY                    ; IS THERE ANOTHER TO PULL FROM STACK?
CCAE:10 FA   CCAA             BPL   MOVMRK           ; YES, GET NEXT LOWER BYTE FROM STACK.
CCB0:18                       CLC                    ; INDICATE NO ERROR.
CCB1:60                       RTS
CCB2:              *
CCB2:20 CD CC      SETMARK    JSR   ADJMARK          ; MAKE ADJUSTMENTS TO REQUESTED MARK ACCORDING TO BASE.
CCB5:90 01   CCB8             BCC   SMARK1           ; BRANCH IF ADJUSTMENT WAS VALID.
CCB7:60                       RTS
CCB8:A2 02         SMARK1     LDX   #2               ; NOW COMPARE END OF FILE WITH NEW
CCBA:A0 17                    LDY   #FCBEOF+2        ; POSITION TO BE SURE IT'S WITHIN
CCBC:B5 2A         CMPEOF     LDA   TPOSLL,X         ; THE BOUNDS OF CURRENTLY DEFINED
CCBE:D1 BA                    CMP   (FCBPTR),Y       ; LIMITS.
CCC0:90 47   CD09             BCC   CKSAMBLK         ; BRANCH IF MARK<EOF
CCC2:D0 06   CCCA             BNE   ERRMEOF          ; RETURN ERROR IF MARK>= EOF
CCC4:88                       DEY
CCC5:CA                       DEX
CCC6:10 F4   CCBC             BPL   CMPEOF
CCC8:30 3F   CD09             BMI   CKSAMBLK         ; BRANCH ALWAYS
CCCA:A9 00         ERRMEOF    LDA   #POSNERR         ; TELL USER MARK IS OUT OF RANGE.
CCCC:60                       RTS                    ; (CARRY IS SET TO INDICATE ERROR)
CCCD:              *
CCCD:A5 A6         ADJMARK    LDA   C.MARK+3         ; MAKE SURE FOURTH BYTE OF DISPLACE IS ZIP
CCCF:D0 29   CCFA             BNE   ERRPOSN          ; BRANCH TO ERR IF NOT
CCD1:A2 FD                    LDX   #$FD             ; ANTICIPATE OTHER THAN BASE OF ZERO
CCD3:A0 12                    LDY   #FCBMARK         ; FURTHER ASSUME IT'S A BASE OFFSET FROM CURRENT POSITION
CCD5:A5 A2                    LDA   C.BASE           ; NOW FIND OUT WHAT IT REALLY IS.
CCD7:4A                       LSR   A                ; (CARRY SET=SUBTRACT, NON ZERO REMAINDER= OFFSET FROM EOF)
CCD8:B0 10   CCEA             BCS   SUBMARK
CCDA:F0 22   CCFE             BEQ   ADJMRK           ; BRANCH IF MARK IS FROM BEGINNING OF FILE
CCDC:B1 BA         ADDPOSN    LDA   (FCBPTR),Y       ; ADD USER QUANTITY TO CURRENT
CCDE:75 A6                    ADC   C.MARK+3,X       ; POSITION TO FORM NEW POSITION.
CCE0:95 2D                    STA   >TPOSLL-$FD,X    ; (NOTE: ZERO PAGE REFERENCE WRAPS AROUND IN Z-PAGE)
CCE2:C8                       INY
CCE3:E8                       INX
CCE4:D0 F6   CCDC             BNE   ADDPOSN          ; ADD ALL THREE BYTES
CCE6:B0 12   CCFA             BCS   ERRPOSN          ; BRANCH IF OVERFLOW
CCE8:F0 1D   CD07             BEQ   ADJMRK1          ; BRANCH ALWAYS
CCEA:              *
```

```
CCEA:D0 02   CCEE   SUBMARK    BNE   SUBPOSN          ; BRANCH IF IT'S AN OFFSET FROM CURRENT POSITION
CCEC:A0 15                     LDY   #FCBEOF          ; OTHERWISE ASSUME OFFSET FROM END OF FILE.
CCEE:B1 BA          SUBPOSN    LDA   (FCBPTR),Y       ; SUBTRACT USER QUANTITY TO FORM
CCF0:F5 A6                     SBC   C.MARK+3,X       ; NEW POSITION. IF FINAL
CCF2:95 2D                     STA   >TPOSLL-$FD,X    ; RESULT IS L.T. ZERO, THEN REPORT
CCF4:C8                        INY                    ; POSITION ERROR...
CCF5:E8                        INX
CCF6:D0 F6   CCEE              BNE   SUBPOSN
CCF8:B0 0D   CD07              BCS   ADJMRK1          ; BRANCH IF LEGAL POSITION CALCULATED.
CCFA:A9 00          ERRPOSN    LDA   #POSNERR
CCFC:38                        SEC                    ; INDICATE ERROR
CCFD:60                        RTS
CCFE:               *
CCFE:A2 02          ADJMRK     LDX   #2               ; FIRST SET UP POSITION TEMPS USED
CD00:B5 A3          ADJMRK0    LDA   C.MARK,X         ; BY BOTH POSITION ROUTINES
CD02:95 2A                     STA   TPOSLL,X
CD04:CA                        DEX
CD05:10 F9   CD00              BPL   ADJMRK0
CD07:18             ADJMRK1    CLC                    ; NO ERRORS
CD08:60                        RTS
CD09:               *
CD09:               *
CD09:        CD09   RDPOSN     EQU   *
CD09:        CD09   CKSAMBLK   EQU   *
CD09:A0 13                     LDY   #FCBMARK+1       ; FIRST TEST TO SEE IF NEW POSITION IS
CD0B:B1 BA                     LDA   (FCBPTR),Y       ; WITHIN THE SAME (CURRENT) DATA BLOCK.
CD0D:29 FE                     AND   #$FE
CD0F:8D E3 DB                  STA   SCRTCH
CD12:C8                        INY                    ; BUMP TO ACCESS HIGHEST ORDER ADDRESS BYTE
CD13:A5 2B                     LDA   TPOSLH           ; GET MIDDLE BYTE OF NEW POSITION
CD15:38                        SEC
CD16:ED E3 DB                  SBC   SCRTCH
CD19:8D E3 DB                  STA   SCRTCH
CD1C:90 0D   CD2B              BCC   TYPMARK          ; BRANCH IF POSSIBLY L.T. CURRENT POSITION
CD1E:C9 02                     CMP   #2               ; MUST BE WITHIN 512 BYTES OF BEGINNING OF CURRENT
CD20:B0 09   CD2B              BCS   TYPMARK
CD22:A5 2C                     LDA   TPOSHI           ; NOW MAKE SURE WERE TALKIN ABOUT
CD24:D1 BA                     CMP   (FCBPTR),Y       ; THE SAME 64K CHUNK!
CD26:D0 03   CD2B              BNE   TYPMARK          ; BRANCH IF WE AREN'T.
CD28:4C 54 CE                  JMP   SVMARK           ; IF WE IS, ADJUST FCB AND POSPTR AND RETURN.
CD2B:               *
CD2B:A0 07          TYPMARK    LDY   #FCBSTYP         ; NOW FIND OUT WHICH TYPE
CD2D:B1 BA                     LDA   (FCBPTR),Y       ; OF FILE WE'RE POSITIONING ON.
CD2F:F0 22   CD53              BEQ   FERRTYP          ; THERE IS NO SUCH TYPE AS ZERO, BRANCH NEVER!
CD31:C9 04                     CMP   #4               ; IS IT A TREE CLASS FILE?
CD33:90 03   CD38              BCC   CHKDSKSW         ; YES, GO POSITION
CD35:4C 84 CE                  JMP   DIRMARK          ; NO, TEST FOR DIRECTORY TYPE.
CD38:               *
CD38:        CD38   CHKDSKSW   EQU   *                ; MAKE SURE S/HE HASN'T MOVED THE VOLUME
CD38:A0 01                     LDY   #FCBDEVN
CD3A:B1 BA                     LDA   (FCBPTR),Y
CD3C:85 35                     STA   DEVNUM           ; MAKE SURE DEVICE NUMBER PARM IS CURRENT
CD3E:20 87 D5                  JSR   TWRPROT1         ; PASSES DEVNUM (CHECK DISK SWITCH)
CD41:AD BB D5                  LDA   DSWGLOB          ; DISK SWITCH GLOBAL
CD44:F0 15   CD5B              BEQ   TREPOS           ; BRANCH IF NONE DETECTED
CD46:20 0A C9       CHKDSKS1   JSR   VERFYVOL         ; MATCHES VCBPTR VS. DEVNUM
```

```
CD49:90 10   CD5B                BCC   TREPOS           ; BRANCH IF DISK HASN'T SWITCHED
CD4B:20 2F DD                    JSR   USRREQ           ; POLITELY ASK USER TO MOUNT
CD4E:90 F6   CD46                BCC   CHKDSKS1         ; SAID HE DID, CHECK AGAIN
CD50:A9 00                       LDA   #VNFERR          ; REFUSES TO MOUNT
CD52:60                          RTS
CD53:                *
CD53:A0 00               FERRTYP LDY   #FCBREFN         ; CLEAR ILLEGALLY TYPED FCB ENTRY
CD55:91 BA                       STA   (FCBPTR),Y
CD57:A9 00                       LDA   #BADREFNUM       ; TELL EM THERE IS NO SUCH FILE
CD59:38                          SEC
CD5A:60                          RTS
CD5B:                *
```

```
CD5B:A0 07         TREPOS     LDY   #FCBSTYP         ; USE STORAGE TYPE AS NUMBER
CD5D:B1 BA                    LDA   (FCBPTR),Y       ; OF LEVELS (SINCE 1=SEED, 2=SAPLING, AND 3=TREE)
CD5F:85 07                    STA   LEVELS
CD61:A0 08                    LDY   #FCBSTAT         ; SINCE IT'S A DIFFERENT DATA
CD63:B1 BA                    LDA   (FCBPTR),Y       ; BLOCK, MUST NOT FORGET PREVIOUS DATA.
CD65:29 40                    AND   #DATMOD          ; THEREFORE, SEE IF PREVIOUS DATA WAS MODIFIED
CD67:F0 05   CD6E             BEQ   POSNEW1          ; THEN DISK MUST BE UPDATED.
CD69:20 84 CF                 JSR   WFCBDAT          ; GO WRITE CURRENT DATA BLOCK.
CD6C:B0 61   CDCF             BCS   POSERR           ; RETURN ANY ERROR ENCOUNTERED.
CD6E:              *
CD6E:A0 14         POSNEW1    LDY   #FCBMARK+2       ; TEST TO SEE IF CURRENT
CD70:B1 BA                    LDA   (FCBPTR),Y       ; INDEX BLOCK IS GOING TO BE USABLE...
CD72:29 FE                    AND   #$FE             ; OR IN OTHER WORDS-
CD74:8D E3 DB                 STA   SCRTCH           ; IS NEW POSITION WITHIN 128K OF THE BEGINNING
CD77:A5 2C                    LDA   TPOSHI           ; OF CURRENT SAPLING LEVEL CHUNK.
CD79:38                       SEC
CD7A:ED E3 DB                 SBC   SCRTCH
CD7D:90 1C   CD9B             BCC   POSNEW2          ; BRANCH IF A NEW INDEX BLOCK IS ALSO NEEDED
CD7F:C9 02                    CMP   #2               ; NEW POSITION IS > THAN BEGINING OF OLD. IS IT WITHIN 128K?
CD81:B0 18   CD9B             BCS   POSNEW2          ; BRANCH IF NOT.
CD83:A6 07                    LDX   LEVELS           ; IS THE FILE WE'RE DEALING WITH A SEED?
CD85:CA                       DEX
CD86:D0 75   CDFD             BNE   DATLEVEL         ; NO, USE CURRENT INDEXES.
CD88:A5 2B         TSTINY     LDA   TPOSLH           ; IS NEW POSITION UNDER 512?
CD8A:4A                       LSR   A
CD8B:05 2C                    ORA   TPOSHI
CD8D:D0 5C   CDEB             BNE   NOIDXDAT         ; NO, MARK BOTH DATA AND INDEX BLOCK AS UN-ALLOCATED.
CD8F:A0 0C                    LDY   #FCBFRST
CD91:B1 BA                    LDA   (FCBPTR),Y       ; FIRST BLOCK IS ONLY BLOCK AND IT'S DATA!
CD93:85 C6                    STA   BLOKNML
CD95:C8                       INY
CD96:B1 BA                    LDA   (FCBPTR),Y       ; (HIGH BLOCK ADDRESS)
CD98:4C 4A CE                 JMP   RNEWPOS          ; GO READ IN BLOCK AND SET APPROPRIATE STATUSES.
CD9B:              *
```

```
CD9B:A0 08         POSNEW2    LDY   #FCBSTAT         ; GOTA CHECK TO SEE IF PREVIOUS
CD9D:B1 BA                    LDA   (FCBPTR),Y       ; INDEX BLOCK WAS MODIFIED.
CD9F:29 80                    AND   #IDXMOD
CDA1:F0 05   CDA8             BEQ   POSNIDX          ; READ IN OVER IT IF CURRENT IS UP TO DATE.
CDA3:20 94 CF                 JSR   WFCBIDX          ; GO UPDATE INDEX ON DISK (BLOCK ADDR IN FCB)
CDA6:B0 27   CDCF             BCS   POSERR
CDA8:A6 07         POSNIDX    LDX   LEVELS           ; BEFORE READING IN TOP INDEX, CHECK TO BE SURE
CDAA:E0 03                    CPX   #3               ; THAT THERE IS A TOP INDEX...
CDAC:F0 23   CDD1             BEQ   POSINDEX         ; BRANCH IF FILE IS FULL BLOWN TREE.
CDAE:A5 2C                    LDA   TPOSHI           ; IS NEW POSITION WITHIN RANGE OF A
CDB0:4A                       LSR   A                ; SAPLING FILE (L.T. 128K)?
CDB1:08                       PHP                    ; ANTICIPATE NO GOOD.
CDB2:A9 07                    LDA   #TOPALC+IDXALC+DATALC ; (TO INDICATE NO LEVEL IS ALLOCATED FOR NEW POSITION.)
CDB4:28                       PLP                    ; Z FLAG TELLS ALL...
CDB5:D0 5D   CE14             BNE   NODATA           ; GO MARK 'EM ALL DUMMY.
CDB7:20 7B CE                 JSR   CLRSTATS         ; GO CLEAR STATUS BITS 0,1,2 (INDEX/DATA ALLOC STATUS).
CDBA:CA                       DEX                    ; (UNAFFECTED SINCE LOADED ABOVE) CHECK FOR SEED
CDBB:F0 CB   CD88             BEQ   TSTINY           ; IF SEED, CHECK FOR POSITION L.T. 512...
CDBD:20 F0 CE                 JSR   RFCBFST          ; GO GET ONLY INDEX BLOCK
CDC0:B0 0D   CDCF             BCS   POSERR           ; BRANCH IF ERROR
CDC2:A0 0E                    LDY   #FCBIDXB         ; SAVE NEWLY LOADED INDEX BLOCK'S ADDRESS
CDC4:A5 C6                    LDA   BLOKNML
CDC6:91 BA                    STA   (FCBPTR),Y
CDC8:C8                       INY
CDC9:A5 C7                    LDA   BLOKNMH
CDCB:91 BA                    STA   (FCBPTR),Y
CDCD:90 2E   CDFD             BCC   DATLEVEL         ; BRANCH ALWAYS...
CDCF:38            POSERR     SEC
CDD0:60                       RTS
CDD1:              *
CDD1:20 7B CE      POSINDEX   JSR   CLRSTATS         ; CLEAR ALL ALLOCATION REQUIREMENTS FOR PREVIOUS POSITION
CDD4:20 F0 CE                 JSR   RFCBFST          ; GET HIGHEST LEVEL INDEX BLOCK.
CDD7:B0 F6   CDCF             BCS   POSERR
CDD9:A5 2C                    LDA   TPOSHI           ; THEN TEST FOR A SAP LEVEL INDEX BLOCK
CDDB:4A                       LSR   A
CDDC:A8                       TAY
CDDD:B1 B2                    LDA   (TINDX),Y
CDDF:E6 B3                    INC   TINDX+1
CDE1:D1 B2                    CMP   (TINDX),Y        ; (BOTH HI AND LO WILL BE ZERO IF NO INDEX EXISTS)
CDE3:D0 0B   CDF0             BNE   SAPLEVEL
CDE5:C9 00                    CMP   #0               ; ARE BOTH BYTES ZERO?
CDE7:D0 07   CDF0             BNE   SAPLEVEL
CDE9:C6 B3                    DEC   TINDX+1          ; DON'T LEAVE WRONG POINTERS LAYING AROUND!
CDEB:A9 03         NOIDXDAT   LDA   #IDXALC+DATALC   ; SHOW NEITHER INDEX OR DATA BLOCK ALLOCATED.
CDED:4C 14 CE                 JMP   NODATA
CDF0:              *
```

```
CDF0:85 C6         SAPLEVEL   STA   BLOKNML          ; READ IN NEXT LOWER INDEX BLOCK
CDF2:B1 B2                    LDA   (TINDX),Y        ; (HI ADDRESS)
CDF4:85 C7                    STA   BLOKNMH
CDF6:C6 B3                    DEC   TINDX+1
CDF8:20 D8 CE                 JSR   RFCBIDX          ; READ IN SAPLING LEVEL
CDFB:B0 D2   CDCF             BCS   POSERR
CDFD:A5 2C         DATLEVEL   LDA   TPOSHI           ; NOW GET BLOCK ADDRESS OF DATA BLOCK
CDFF:4A                       LSR   A
CE00:A5 2B                    LDA   TPOSLH           ; ( IF THERE IS ONE )
CE02:6A                       ROR   A
CE03:A8                       TAY
CE04:B1 B2                    LDA   (TINDX),Y        ; DATA BLOCK ADDRESS LOW
CE06:E6 B3                    INC   TINDX+1
CE08:D1 B2                    CMP   (TINDX),Y
CE0A:D0 38   CE44             BNE   POSNEW3
CE0C:C9 00                    CMP   #0
CE0E:D0 34   CE44             BNE   POSNEW3
CE10:A9 01                    LDA   #DATALC          ; SHOW DATA BLOCK AS NEVER BEEN ALLOCATED
CE12:C6 B3                    DEC   TINDX+1
CE14:              *
CE14:A0 08         NODATA     LDY   #FCBSTAT
CE16:11 BA                    ORA   (FCBPTR),Y       ; SET STATUS TO SHOW WHATS MISSIN'
CE18:91 BA                    STA   (FCBPTR),Y
CE1A:4A                       LSR   A                ; THROW AWAY BIT THAT SAYS DATA BLOCK UN-ALLOCATED
CE1B:4A                       LSR   A                ; CUZ WE KNOW THAT. CARRY NOW INDICATES IF INDEX BLOCK
CE1C:20 32 CE                 JSR   ZIPDATA          ; ALSO IS INVALID AND NEEDS TO BE ZEROED (CARRY UNDISTURBED)
CE1F:90 33   CE54             BCC   SVMARK           ; BRANCH IF INDEX BLOCK DOESN'T NEED ZIPPIN.
CE21:91 B2         ZIPIDX     STA   (TINDX),Y
CE23:C8                       INY
CE24:D0 FB   CE21             BNE   ZIPIDX
CE26:E6 B3                    INC   TINDX+1
CE28:91 B2         ZPIDX1     STA   (TINDX),Y
CE2A:C8                       INY
CE2B:D0 FB   CE28             BNE   ZPIDX1
CE2D:C6 B3                    DEC   TINDX+1          ; RESTORE PROPER ADDRESS
CE2F:4C 54 CE                 JMP   SVMARK
CE32:              *
CE32:A9 00         ZIPDATA    LDA   #0               ; ALSO IS INVALID AND NEEDS TO BE ZEROED.
CE34:A8                       TAY
CE35:91 BC         ZIPDAT0    STA   (DATPTR),Y       ; ZERO OUT DATA AREA
CE37:C8                       INY
CE38:D0 FB   CE35             BNE   ZIPDAT0
CE3A:E6 BD                    INC   DATPTR+1
CE3C:91 BC         ZPDAT1     STA   (DATPTR),Y
CE3E:C8                       INY
CE3F:D0 FB   CE3C             BNE   ZPDAT1
CE41:C6 BD                    DEC   DATPTR+1
CE43:60                       RTS
CE44:              *
```

```
CE44:              *
CE44:85 C6         POSNEW3    STA   BLOKNML          ; GET DATA BLOCK OF NEW POSITION
CE46:B1 B2                    LDA   (TINDX),Y        ; (HI ADDRESS)
CE48:C6 B3                    DEC   TINDX+1
CE4A:85 C7         RNEWPOS    STA   BLOKNMH
CE4C:20 CA CE                 JSR   RFCBDAT
CE4F:B0 28   CE79             BCS   PRITZ            ; RETURN ANY ERROR
CE51:20 7B CE                 JSR   CLRSTATS         ; SHOW WHOLE CHAIN IS ALLOCATED
CE54:A0 14         SVMARK     LDY   #FCBMARK+2       ; UPDATE POSITION IN FILE CONTROL BLOCK
CE56:A2 02                    LDX   #2
CE58:B1 BA         SVMRK1     LDA   (FCBPTR),Y       ; REMEMBER OLDMARK IN CASE
CE5A:99 E1 DB                 STA   OLDMARK-FCBMARK,Y ; CALLING ROUTINE FAILS LATER
CE5D:B5 2A                    LDA   TPOSLL,X
CE5F:91 BA                    STA   (FCBPTR),Y
CE61:88                       DEY
CE62:CA                       DEX                    ; MOVE 3 BYTE POSITION MARKER
CE63:10 F3   CE58             BPL   SVMRK1
CE65:              *
CE65:18                       CLC                    ; LAST, BUT NOT LEAST, SET UP
CE66:A5 BC                    LDA   DATPTR           ; INDIRECT ADDRESS TO BUFFER PAGE POINTED
CE68:85 BE                    STA   POSPTR           ; TO BY THE CURRENT POSITION MARKER.
CE6A:A5 2B                    LDA   TPOSLH
CE6C:29 01                    AND   #1
CE6E:65 BD                    ADC   DATPTR+1
CE70:85 BF                    STA   POSPTR+1
CE72:AD BD 14                 LDA   SISDATP
CE75:8D BF 14                 STA   SISPOSP          ; SISTER PAGE BYTE ALSO.
CE78:60                       RTS                    ; CARRY SHOULD ALWAYS BE CLEAR
CE79:38            PRITZ      SEC                    ; RANDOM ERROR
CE7A:60                       RTS                    ; RETURN
CE7B:              *
CE7B:              *
CE7B:A0 08         CLRSTATS   LDY   #FCBSTAT         ; CLEAR ALLOCATION STATES FOR DATA BLOCK
CE7D:B1 BA                    LDA   (FCBPTR),Y       ; AND BOTH LEVELS OF INDEXES.
CE7F:29 F8                    AND   #$FF-TOPALC-IDXALC-DATALC
CE81:91 BA                    STA   (FCBPTR),Y       ; THIS SAYS THAT EITHER THEY EXIST CURRENTLY
CE83:60                       RTS                    ; OR THAT THEY'RE UNNECESSARY FOR CURRENT POSITION.
CE84:              *
```

```
CE84:              *
CE84:C9 0D         DIRMARK    CMP   #DIRTYP          ; IS IT A DIRECTORY?
CE86:F0 05   CE8D             BEQ   DIRPOS           ; YES...
CE88:A9 00                    LDA   #CPTERR          ; NO, THERE IS A COMPATABLITY PROBLEM-
CE8A:20 00 00                 JSR   SYSERR           ; THE DAMN THING SHOULD OF NEVER BEEN OPENED!
CE8D:              *
CE8D:AD E3 DB      DIRPOS     LDA   SCRTCH           ; RECOVER RESULTS OF PREVIOUS SUBTRACTION.
CE90:4A                       LSR   A                ; USE DIFFERENCE AS COUNTER AS TO HOW MANY
CE91:85 0B                    STA   CNTENT           ; BLOCKS MUST BE READ TO GET TO NEW POSITION.
CE93:A0 13                    LDY   #FCBMARK+1       ; TEST FOR POSITION DIRECTION.
CE95:B1 BA                    LDA   (FCBPTR),Y
CE97:C5 2B                    CMP   TPOSLH           ; CARRY INDICATES DIRECTION...
CE99:90 0D   CEA8             BCC   DIRFWRD          ; IF SET, POSITION FORWARD.
CE9B:A0 00         DIRVRSE    LDY   #0               ; OTHERWISE, READ DIRECTORY FILE IN REVERSE ORDER.
CE9D:20 B5 CE                 JSR   DIRPOS1          ; READ PREVIOUS BLOCK.
CEA0:B0 22   CEC4             BCS   DRPOSERR         ; BRANCH IF ANYTHING GOES WRONG.
CEA2:E6 0B                    INC   CNTENT           ; COUNT UP TO 128
CEA4:10 F5   CE9B             BPL   DIRVRSE          ; LOOP IF THERE IS MORE BLOCKS TO PASS OVER.
CEA6:30 AC   CE54             BMI   SVMARK           ; BRANCH ALWAYS.
CEA8:              *
CEA8:A0 02         DIRFWRD    LDY   #2               ; POSITION IS FORWARD FROM CURRENT POSITION.
CEAA:20 B5 CE                 JSR   DIRPOS1          ; READ NEXT DIRECTORY BLOCK.
CEAD:B0 15   CEC4             BCS   DRPOSERR
CEAF:C6 0B                    DEC   CNTENT
CEB1:D0 F5   CEA8             BNE   DIRFWRD          ; LOOP IF POSITION NOT FOUND IN THIS BLOCK.
CEB3:F0 9F   CE54             BEQ   SVMARK           ; BRANCH ALWAYS.
CEB5:              *
CEB5:B1 BC         DIRPOS1    LDA   (DATPTR),Y       ; GET LINK ADDRESS OF PREVIOUS OR
CEB7:85 C6                    STA   BLOKNML          ; NEXT DIRECTORY BLOCK.
CEB9:C8                       INY                    ; BUT FIRST BE SURE THERE IS A LINK.
CEBA:D1 BC                    CMP   (DATPTR),Y
CEBC:D0 08   CEC6             BNE   DIRPOS2          ; BRANCH IF CERTAIN LINK EXISTS
CEBE:C9 00                    CMP   #0               ; ARE BOTHE LINK BYTES 0?
CEC0:D0 04   CEC6             BNE   DIRPOS2          ; NOPE, JUST HAPPEN TO BE THE SAME VALUE.
CEC2:A9 00                    LDA   #EOFERR          ; SOMETHING IS WRONG WITH THIS DIRECTORY FILE!
CEC4:38            DRPOSERR   SEC                    ; INDICATE ERROR
CEC5:60                       RTS
CEC6:              *
CEC6:B1 BC         DIRPOS2    LDA   (DATPTR),Y       ; (HIGH ORDER BLOCK ADDRESS)
CEC8:85 C7                    STA   BLOKNMH
CECA:              * DROP INTO 'RFCBDAT' (READ FILE'S DATA BLOCK)
CECA:              *
CECA:              * NOTE: FOR DIRECTORY POSITIONING NO OPTIMIZATION HAS BEEN
CECA:              * DONE SINCE DIRECTORY FILES WILL ALMOST ALWAYS BE LESS
CECA:              * THAN 6 BLOCKS. IF MORE SPEED IS REQUIRED OR DIRECTORY
CECA:              * TYPE FILES ARE TO BE USED FOR OTHER PURPOSES REQUIRING
CECA:              * MORE BLOCKS, THEN THE RECOMMENDED METHOD IS TO CALL
CECA:              * 'RFCBDAT' FOR THE FIRST BLOCK AND GO DIRECTLY TO
CECA:              * DEVICE (VIA JMP (IOUNITL)) HANDLER FOR SUBSEQUENT
CECA:              * ACCESSES.
CECA:              * ALSO NOTE THAT NO CHECKING IS DONE FOR READ/WRITE
CECA:              * ENABLE SINCE A DIRECTORY FILE CAN ONLY BE OPENED
CECA:              * FOR READ ACCESS.
CECA:              *
```

```
CECA:              *
CECA:A9 00         RFCBDAT    LDA   #RDCMD           ; SET READ COMMAND.
CECC:85 C0                    STA   DHPCMD
CECE:A2 BC                    LDX   #DATPTR          ; USE X TO POINT AT ADDRESS OF DATA BUFFER
CED0:20 0E CF                 JSR   FILEIO1          ; GO DO FILE INPUT.
CED3:A0 10                    LDY   #FCBDATB         ; SAVE BLOCK NUMBER JUST READ IN FCB.
CED5:90 0E   CEE5             BCC   FCBLOKNM         ; BRANCH IF NO ERRORS HAPPENED.
CED7:60                       RTS                    ; RETURN ERROR
CED8:              *
CED8:A9 00         RFCBIDX    LDA   #RDCMD           ; PREPARE TO READ IN INDEX BLOCK.
CEDA:85 C0                    STA   DHPCMD
CEDC:A2 B2                    LDX   #TINDX           ; POINT AT ADDRESS OF CURRENT INDEX BUFFER
CEDE:20 0E CF                 JSR   FILEIO1          ; GO READ INDEX BLOCK.
CEE1:B0 0C   CEEF             BCS   RDFCBERR         ; REPORT ERROR
CEE3:A0 0E                    LDY   #FCBIDXB         ; SAVE BLOCK ADDRESS OF THIS INDEX IN FCB.
CEE5:A5 C6         FCBLOKNM   LDA   BLOKNML
CEE7:91 BA                    STA   (FCBPTR),Y
CEE9:C8                       INY
CEEA:A5 C7                    LDA   BLOKNMH
CEEC:91 BA                    STA   (FCBPTR),Y
CEEE:18                       CLC
CEEF:60            RDFCBERR   RTS
CEF0:              *
CEF0:A2 B2         RFCBFST    LDX   #TINDX           ; POINT AT ADDRESS OF INDEX BUFFER
CEF2:A0 0C                    LDY   #FCBFRST         ; AND BLOCK ADDRESS OF FIRST FILE BLOCK IN FCB
CEF4:A9 00                    LDA   #RDCMD           ; AND LASTLY, MAKE IT A READ!
CEF6:              * DROP INTO DOFILEIO
CEF6:              *
CEF6:85 C0         DOFILEIO   STA   DHPCMD           ; SAVE COMMAND.
CEF8:B1 BA                    LDA   (FCBPTR),Y       ; GET DISK BLOCK ADDRESS FROM FCB.
CEFA:85 C6                    STA   BLOKNML
CEFC:C8                       INY                    ; BLOCK ZERO NOT LEGAL.
CEFD:D1 BA                    CMP   (FCBPTR),Y
CEFF:D0 09   CF0A             BNE   FILEIO
CF01:C9 00                    CMP   #0               ; ARE BOTH BYTES ZERO?
CF03:D0 05   CF0A             BNE   FILEIO           ; NO, CONTINUE WITH REQUEST.
CF05:A9 00                    LDA   #ALCERR          ; OTHERWISE REPORT ALLOCATION ERROR.
CF07:20 00 00                 JSR   SYSDEATH         ; NEVER RETURNS...
CF0A:              *
```

```
CF0A:B1 BA         FILEIO     LDA   (FCBPTR),Y       ; GET HIGH ADDRESS OF DISK BLOCK
CF0C:85 C7                    STA   BLOKNMH
CF0E:B5 00         FILEIO1    LDA   0,X              ; GET MEMORY ADDRESS OF BUFFER FROM
CF10:85 C2                    STA   DBUFPL           ; S.O.S. ZERO PAGE POINTED TO BY
CF12:20 BD D5                 JSR   WRAPADJ          ;GO ADJUST FOR BANK CROSSING <SRS 82.162>
CF15:B5 01                    LDA   1,X
CF17:85 C3                    STA   DBUFPH           ; SET HI BYTE
CF19:BD 01 14                 LDA   SISTER+1,X       ; AND BANK PAIR BYTE. <SRS 82.162>
CF1C:8D C3 14                 STA   SISBPH
CF1F:A0 01                    LDY   #FCBDEVN
CF21:B1 BA                    LDA   (FCBPTR),Y       ; OF COURSE HAVING THE DEVICE NUMBER
CF23:85 35                    STA   DEVNUM           ; WOULD MAKE THE WHOLE OPERATION MORE MEANINGFUL...
CF25:A9 02         FILEIO2    LDA   #2               ; ALSO, SET UP BYTE COUNT TO 512 AND
CF27:85 C5                    STA   RQCNTH           ; SET 'BYTES READ' POINTER TO
CF29:85 34                    STA   IOACCESS         ; (INTERUPT! SET TO INDICATE REG CALL MADE TO DEV HANDLER.
RETURN INTERUPT!)
CF2B:A9 67                    LDA   #>TRASH          ; A PLACE TO THROW BYTES READ AWAY
CF2D:85 C8                    STA   BRDPTR
CF2F:A9 CF                    LDA   #<TRASH          ; LOCALLY DEFINED
CF31:85 C9                    STA   BRDPTR+1
CF33:A9 00                    LDA   #0               ; SO THAT IT DOESN'T MESS UP ANY OTHER DATA.
CF35:85 C4                    STA   RQCNTL
CF37:8D C9 14                 STA   SSBRDPH          ; ('BYTES READ' IS THROWN AWAY)
CF3A:A5 35         RPEATIO1   LDA   DEVNUM           ; TRANSFER THE DEVICE NUMBER FOR DISPATCHER TO CONVERT TO UNIT
NUMBER.
CF3C:85 C1                    STA   UNITNUM
CF3E:A0 09         RPEATIO0   LDY   #$9              ; PREPARE TO SAVE DEVICE PARMS
CF40:B9 C0 00      SAVPRMS    LDA   DEVICE,Y         ; MOVE FROM Z PAGE
CF43:99 69 CF                 STA   RPTBLOK,Y        ; TO MY OWN SPACE
CF46:88                       DEY                    ; FROM $C9 THROUGH $C0
CF47:10 F7   CF40             BPL   SAVPRMS
CF49:        CF49  DMGRGO     EQU   *                ; CALL EXTERNAL DEVICE MANAGER
CF49:A9 00                    LDA   #0
CF4B:8D 00 00                 STA   SERR             ; CLEAR GLOBAL ERROR VALUE
CF4E:20 00 00                 JSR   DMGR             ; CALL THE DRIVER
CF51:90 05   CF58             BCC   RRITZ            ; RTS IF NO ERRORS
CF53:C9 00                    CMP   #XDISKSW         ; DISKSWITCH ITERATES
CF55:F0 02   CF59             BEQ   RPEATIO2         ; BRANCH IF DISK SWITCH AND REPEAT I/O REQUEST
CF57:38                       SEC                    ; REPORT ERROR
CF58:60            RRITZ      RTS
CF59:A0 09         RPEATIO2   LDY   #$9              ; LENGTH OF PARM BLOCK
CF5B:B9 69 CF      GETPRMS    LDA   RPTBLOK,Y
CF5E:99 C0 00                 STA   DEVICE,Y         ; RESTORE POSSIBLY DISTURBED PARM BLOCK
CF61:88                       DEY
CF62:10 F7   CF5B             BPL   GETPRMS
CF64:4C 49 CF                 JMP   DMGRGO           ; AND TRY THE I/O AGAIN
CF67:              *
CF67:              *
CF67:        0002  TRASH      DS    2                ; ONLY USED TO PUT BYTES READ TO SLEEP
CF69:        000A  RPTBLOK    DS    10               ; DMGR PARM SAVE BLOCK
CF73:              *
CF73:              *
CF73:A0 01         WFCBFST    LDY   #FCBDEVN         ; FETCH THE
CF75:B1 BA                    LDA   (FCBPTR),Y       ;  DEVICE NUMBER
CF77:AA                       TAX                    ;    AND UPDATE
CF78:20 E4 CB                 JSR   UPBMAP           ;      ITS BITMAP
CF7B:A2 B2                    LDX   #TINDX           ; POINT AT ADDRESS OF INDEX BLOCK
CF7D:A0 0C                    LDY   #FCBFRST         ; AND THE DISK ADDRESS OF FILE'S FIRST BLOCK IN FCB
```

```
CF7F:A9 01                    LDA   #WRTCMD          ; LASTLY, MAKE IT A WRITE REQUEST.
CF81:4C F6 CE                 JMP   DOFILEIO         ; AND GO DO IT!
CF84:              *
CF84:A2 BC         WFCBDAT    LDX   #DATPTR
CF86:A0 10                    LDY   #FCBDATB         ; POINT AT MEMORY ADDRESS WITH X AND DISK ADDRESS WITH Y.
CF88:A9 01                    LDA   #WRTCMD          ; WRITE DATA BLOCK.
CF8A:20 F6 CE                 JSR   DOFILEIO
CF8D:B0 20   CFAF             BCS   FILIOERR         ; REPORT ANY ERRORS
CF8F:A9 BF                    LDA   #$FF-DATMOD      ; MARK DATA STATUS AS CURRENT.
CF91:4C A9 CF                 JMP   FCBUPDAT
CF94:              *
CF94:A0 01         WFCBIDX    LDY   #FCBDEVN         ; MAKE SURE
CF96:B1 BA                    LDA   (FCBPTR),Y       ;  THE BITMAP
CF98:AA                       TAX                    ;    FOR THIS DEVICE ("X")
CF99:20 E4 CB                 JSR   UPBMAP           ;      IS UPDATED
CF9C:A2 B2                    LDX   #TINDX           ; POINT AT ADDRESS OF INDEX BUFFER
CF9E:A0 0E                    LDY   #FCBIDXB         ; AND BLOCK ADDRESS OF THAT INDEX BLOCK.
CFA0:A9 01                    LDA   #WRTCMD
CFA2:20 F6 CE                 JSR   DOFILEIO         ; GO WRITE OUT INDEX BLOCK.
CFA5:B0 08   CFAF             BCS   FILIOERR         ; REPORT ANY ERRORS
CFA7:A9 7F                    LDA   #$FF-IDXMOD      ; MARK INDEX STATUS AS CURRENT.
CFA9:A0 08         FCBUPDAT   LDY   #FCBSTAT         ; CHANGE STATUS BYTE TO
CFAB:31 BA                    AND   (FCBPTR),Y       ; REFLECT SUCCESSFUL DISK FILE UPDATE.
CFAD:91 BA                    STA   (FCBPTR),Y       ; (CARRY IS UNAFFECTED)
CFAF:60            FILIOERR   RTS
CFB0:              *
CFB0:              *
```

```
CFB0:20 80 C4      OPEN       JSR   FINDFILE         ; FIRST OF ALL LOOK UP THE FILE...
CFB3:90 04   CFB9             BCC   OPEN0
CFB5:C9 00                    CMP   #BADPATH         ; IS AN ATTEMPT TO OPEN A ROOT DIRECTORY?
CFB7:D0 07   CFC0             BNE   ERROPN           ; NO, PASS BACK ERROR
CFB9:              *
CFB9:20 F6 D0      OPEN0      JSR   TSTOPEN          ; FIND OUT IF ANY OTHER FILES ARE WRITING
CFBC:90 04   CFC2             BCC   OPEN1            ; TO THIS SAME FILE. (BRANCH IF NOT)
CFBE:A9 00         ERRBUSY    LDA   #FILBUSY         ; REPORT SHARED ACCESS NOT ALLOWED.
CFC0:38            ERROPN     SEC
CFC1:60                       RTS                    ; RETURN ERROR.
CFC2:              *
CFC2:A5 BC         OPEN1      LDA   DATPTR           ; GET ADDRESS OF FIRST FREE FCB FOUND
CFC4:85 BA                    STA   FCBPTR           ; DURING TEST OPEN SEQUENCE AND USE
CFC6:A5 BD                    LDA   DATPTR+1         ; IT AS FILE CONTROL AREA. IF HIGH BYTE OF
CFC8:85 BB                    STA   FCBPTR+1         ; POINTER IS ZERO, THEN NO FCB
CFCA:D0 04   CFD0             BNE   ASGNFCB          ; IS AVAILABLE FOR USE.
CFCC:A9 00                    LDA   #FCBFULL         ; REPORT FCB FULL ERROR.
CFCE:38                       SEC
CFCF:60                       RTS
CFD0:              *
CFD0:A0 1F         ASGNFCB    LDY   #$1F             ; ASSIGN FCB, BUT FIRST
CFD2:A9 00                    LDA   #0               ; CLEAN OUT ANY OLD RUBBISH LEFT AROUND...
CFD4:91 BA         CLRFCB     STA   (FCBPTR),Y
CFD6:88                       DEY
CFD7:10 FB   CFD4             BPL   CLRFCB
CFD9:A0 06                    LDY   #FCBENTN         ; NOW BEGIN CLAIM BY MOVING IN FILE
CFDB:B9 B3 DB      FCBOWNR    LDA   D.DEV-1,Y        ; OWNERSHIP INFORMATION.
CFDE:91 BA                    STA   (FCBPTR),Y       ; NOTE: THIS CODE DEPENDS UPON THE DEFINED
CFE0:88                       DEY                    ; ORDER OF BOTH THE FCB AND DIRECTORY ENTRY
CFE1:D0 F8   CFDB             BNE   FCBOWNR          ; BUFFER (D.). BEWARE OF CHANGES!!! *************
CFE3:AD BA DB                 LDA   DFIL+D.STOR      ; GET STORAGE TYPE.
CFE6:4A                       LSR   A                ; STRIP OFF FILE NAME LENGTH.
CFE7:4A                       LSR   A
CFE8:4A                       LSR   A                ; (BY DIVIDING BY 16)
CFE9:4A                       LSR   A
CFEA:AA                       TAX                    ; SAVE IN X FOR LATER TYPE COMPARISON
CFEB:A0 07                    LDY   #FCBSTYP
CFED:91 BA                    STA   (FCBPTR),Y       ; SAVE STORAGE TYPE.
CFEF:A5 A7                    LDA   C.OPLSTLN        ; IS THERE AN OPEN LIST?
CFF1:F0 11   D004             BEQ   DEFOPEN          ; NO, USE DEFAULT REQUST ACCESS...
CFF3:A0 00                    LDY   #0               ; YES, FIND OUT WHAT ACCESS IS REQUESTED.
CFF5:B1 A5                    LDA   (C.OPLIST),Y     ; IF REQ-ACCESS IS ZERO, THEN
CFF7:F0 0B   D004             BEQ   DEFOPEN          ; USE DEFAULTS...
CFF9:2D D8 DB                 AND   DFIL+D.ATTR      ; CHECK REQUEST AGAINST ATTRIBUTES.
CFFC:D1 A5                    CMP   (C.OPLIST),Y     ; WERE ALL ACCESS REQUESTS SATISFIED?
CFFE:F0 09   D009             BEQ   SVATTRB          ; YES, SAVE ATTRIBUTES.
D000:A9 00                    LDA   #ACCSERR         ; REPORT ACCESS REQUEST CAN'T BE MET.
D002:38                       SEC
D003:60                       RTS
```

```
D004:AD D8 DB      DEFOPEN    LDA   DFIL+D.ATTR      ; GET FILES ATTRIBUTES AND
D007:29 03                    AND   #READEN+WRITEN   ; USE IT AS A DEFAULT ACCESS REQUEST.
D009:A0 09         SVATTRB    LDY   #FCBATTR
D00B:E0 0D                    CPX   #DIRTYP          ; IF DIRECTORY, DON'T ALLOW WRITE ENABLE
D00D:D0 02   D011             BNE   SVATTR1
D00F:29 01                    AND   #READEN
D011:91 BA         SVATTR1    STA   (FCBPTR),Y
D013:29 02                    AND   #WRITEN          ; CHECK FOR WRITE ENABLED REQUESTED.
D015:F0 04   D01B             BEQ   OPEN2            ; BRANCH IF READ ONLY OPEN.
D017:A5 08                    LDA   TOTENT           ; OTHERWISE, BE SURE NO ONE ELSE IS READING SAME
D019:D0 A3   CFBE             BNE   ERRBUSY          ; FILE (SET UP BY TSTOPEN).
D01B:AD D7 DB      OPEN2      LDA   DFIL+D.COMP      ; OH, BY THE WAY... IS THIS FILE
D01E:F0 04   D024             BEQ   OPEN3            ; COMPATABLE WITH VERSION 0000? ***************
D020:A9 00         ERRCMPAT   LDA   #CPTERR          ; REPORT FILE IS INCOMPATABLE!
D022:38                       SEC
D023:60                       RTS
D024:              *
D024:E0 04         OPEN3      CPX   #TRETYP+1        ; IS IT A TREE TYPE FILE?
D026:90 04   D02C             BCC   OPEN4            ; TEST FOR FURTHER COMPATABLITY. IT MUST
D028:E0 0D                    CPX   #DIRTYP          ; BE EITHER A TREE OR A DIRECTORY.
D02A:D0 F4   D020             BNE   ERRCMPAT         ; REPORT INCOMPATABLE.
D02C:A0 0C         OPEN4      LDY   #FCBFRST         ; MOVE ADDRESS OF FIRST BLOCK OF FILE
D02E:AD CB DB                 LDA   DFIL+D.FRST      ; INTO FCB. NO CHECKING IS DONE FOR VALIDITY.
D031:91 BA                    STA   (FCBPTR),Y
D033:85 C6                    STA   BLOKNML
D035:C8                       INY
D036:AD CC DB                 LDA   DFIL+D.FRST+1
D039:91 BA                    STA   (FCBPTR),Y       ; NOTE: THE FCB HAS NOT BEEN OFFICIALLY
D03B:85 C7                    STA   BLOKNMH          ; CLAIMED YET. TO DO THIS, THE FIRST BYTE
D03D:A0 15                    LDY   #FCBEOF          ; MUST CONTAIN A VALID REFERENCE NUMBER.
D03F:B9 BA DB      EOFCBMV    LDA   DFIL+D.EOF-FCBEOF,Y ; MOVE CURRENT END OF FILE
D042:91 BA                    STA   (FCBPTR),Y       ; TO FCB.
D044:C8                       INY
D045:C0 18                    CPY   #FCBEOF+3
D047:D0 F6   D03F             BNE   EOFCBMV
D049:AD CD DB                 LDA   DFIL+D.USAGE
D04C:91 BA                    STA   (FCBPTR),Y       ; AND CURRENT BLOCK COUNT OF FILE.
D04E:C8                       INY
D04F:AD CE DB                 LDA   DFIL+D.USAGE+1
D052:91 BA                    STA   (FCBPTR),Y
D054:A5 A7                    LDA   C.OPLSTLN        ; NOW THAT WE'VE COME THIS FAR, FIND
D056:F0 28   D080             BEQ   DEFBUFR          ; OUT WHICH TYPE OF BUFFER AND ALLOCATE IT!
D058:C9 01                    CMP   #1               ; WAS IT ONLY TO SET ATTRIBUTES?
D05A:F0 24   D080             BEQ   DEFBUFR
D05C:C9 04                    CMP   #4               ; IS A FULL ADDRESS INCLUDED?
D05E:F0 04   D064             BEQ   UBUFSPEC
D060:A9 00                    LDA   #BADLSTCNT
D062:38                       SEC
D063:60                       RTS
D064:              *
```

```
D064:A0 01         UBUFSPEC   LDY   #1               ; (INDEX TO 'PAGECNT' OF OPEN LIST)
D066:B1 A5                    LDA   (C.OPLIST),Y     ; IS USER SPECIFING THE BUFFER?
D068:F0 16   D080             BEQ   DEFBUFR          ; NO, USE DEFAULT BUFFER (DYNAMIC)
D06A:E0 04                    CPX   #TRETYP+1        ; IF TREE TYPE FILE, THEN AT LEAS 4 PAGES ARE NEEDED.
D06C:90 08   D076             BCC   ONEKTST          ; BRANCH IF TREE TYPE.
D06E:C9 02                    CMP   #2               ; DID USER GIVE AT LEAST 2 PAGES FOR DIRECTORY TYPE?
D070:B0 08   D07A             BCS   FIXDBUF          ; YES, LOG IT WITH BUFFER MANAGER
D072:A9 00         ERRBTS     LDA   #BTSERR          ; REPORT NOT ENOUGH BUFFER SPACE.
D074:38                       SEC
D075:60                       RTS
D076:              *
D076:C9 04         ONEKTST    CMP   #4               ; IS THERE AT LEAST ONE KILOBYTE BUFFER FOR TREES?
D078:90 F8   D072             BCC   ERRBTS           ; NO, THEN TO HELL WITH IT!.
D07A:20 00 00      FIXDBUF    JSR   REQFXBUF         ; CALL BOB AND ASK FOR HIM TO FIX IT...
D07D:90 0E   D08D             BCC   FCBUFFER         ; GO SAVE BUFFER NUMBER.
D07F:60            ERROPN1    RTS                    ; RETURN ANY ERROR ENCOUNTERED.
D080:              *
D080:A9 04         DEFBUFR    LDA   #4               ; ASSUME TREE FILE (4 PAGES REQUIRED)
D082:E0 04                    CPX   #TRETYP+1
D084:90 02   D088             BCC   BUFREQST         ; BRANCH IF IT IS A TREE.
D086:A9 02                    LDA   #2               ; OTHERWIZE, WE JUST NEED TWO PAGES.
D088:20 00 00      BUFREQST   JSR   REQBUF           ; CALL BOB TO ALLOCATE A DYNAMIC BUFFER.
D08B:B0 F2   D07F             BCS   ERROPN1          ; REPORT ANY ERRORS.
D08D:A0 0B         FCBUFFER   LDY   #FCBBUFN         ; SAVE BUFFER NUMBER AND THEN
D08F:91 BA                    STA   (FCBPTR),Y       ; FIND OUT WHERE IT IS.
D091:20 A4 BE                 JSR   GTBUFFRS         ; HAVE BOB RETURN ADDRESS IN DATA & INDEX POINTERS.
D094:B0 2F   D0C5             BCS   ERROPEN2         ; IF ERROR, FREE BUFFER BEFOR RETURNING.
D096:A0 00                    LDY   #FCBREFN         ; NOW CLAIM FCB FOR THIS FILE.
D098:A5 0B                    LDA   CNTENT           ; THIS WAS SET UP BY 'TSTOPEN'.............
D09A:91 BA                    STA   (FCBPTR),Y
D09C:A0 1B                    LDY   #FCBLEVL         ; MARK LEVEL
D09E:AD 00 00                 LDA   LEVEL            ; AT WHICH
D0A1:91 BA                    STA   (FCBPTR),Y       ; FILE WAS OPENED
D0A3:A0 07                    LDY   #FCBSTYP         ; GET STORAGE TYPE AGAIN.
D0A5:B1 BA                    LDA   (FCBPTR),Y       ; FILE MUST BE POSITIONED TO BEGINNING.
D0A7:C9 04                    CMP   #TRETYP+1        ; IS IT A TREE FILE?
D0A9:B0 2B   D0D6             BCS   OPNDIR           ; NO, ASSUME IT'S A DIRECTORY.
D0AB:A9 FF                    LDA   #$FF             ; FOOL THE POSITION ROUTINE INTO GIVING
D0AD:A0 12                    LDY   #FCBMARK         ; A VALID POSITION WITH PRELOADED DATA, ETC.
D0AF:91 BA         OPNPOS     STA   (FCBPTR),Y
D0B1:C8                       INY
D0B2:C0 15                    CPY   #FCBMARK+3
D0B4:D0 F9   D0AF             BNE   OPNPOS
D0B6:A0 02                    LDY   #2               ; SET DESIRED POSITION TO ZERO.
D0B8:A9 00                    LDA   #0
D0BA:99 2A 00      OPNPOS1    STA   TPOSLL,Y
D0BD:88                       DEY
D0BE:10 FA   D0BA             BPL   OPNPOS1
D0C0:20 09 CD                 JSR   RDPOSN           ; LET TREE POSITION ROUTINE DO THE REST.
D0C3:90 16   D0DB             BCC   OPENDONE         ; BRANCH IF SUCCESSFUL.
D0C5:              *
```

```
D0C5:48            ERROPEN2   PHA                    ; SAVE ERROR CODE.
D0C6:A0 0B                    LDY   #FCBBUFN         ; SINCE ERROR WAS ENCOUNTERED BEFORE FILE
D0C8:B1 BA                    LDA   (FCBPTR),Y       ; WAS SUCCESSFULLY OPENED, THEN
D0CA:20 00 00                 JSR   RELBUF           ; IT'S NECESSARY TO FREE THE BUFFER AND
D0CD:A0 00                    LDY   #FCBREFN         ; FILE CONTROL BLOCK.
D0CF:A9 00                    LDA   #0
D0D1:91 BA                    STA   (FCBPTR),Y
D0D3:68                       PLA
D0D4:38                       SEC
D0D5:60                       RTS
D0D6:              *
D0D6:20 CA CE      OPNDIR     JSR   RFCBDAT          ; READ IN FIRST BLOCK OF DIRECTORY FILE.
D0D9:B0 EA   D0C5             BCS   ERROPEN2         ; RETURN ANY ERROR AFTER FREEING BUFFER & FCB
D0DB:A0 1E         OPENDONE   LDY   #VCBOPNC         ; INCREMENT OPEN COUNT FOR THIS
D0DD:B1 B6                    LDA   (VCBPTR),Y       ; VOLUME. ALSO MARK STATUS.
D0DF:18                       CLC
D0E0:69 01                    ADC   #1
D0E2:91 B6                    STA   (VCBPTR),Y
D0E4:A0 11                    LDY   #VCBSTAT         ; HI BIT INDICATES VOLUME BUSY
D0E6:B1 B6                    LDA   (VCBPTR),Y
D0E8:09 80                    ORA   #$80
D0EA:91 B6                    STA   (VCBPTR),Y       ; DOESN'T MATTER HOW MANY, JUST BE SURE IT'S SET.
D0EC:A0 00                    LDY   #FCBREFN         ; PASS USER HIS REFERENCE NUMBER
D0EE:B1 BA                    LDA   (FCBPTR),Y
D0F0:A0 00                    LDY   #0
D0F2:91 A3                    STA   (C.OUTREF),Y
D0F4:18                       CLC
D0F5:60                       RTS
D0F6:              *
```

```
D0F6:              *
D0F6:AD 28 00      TSTOPEN    LDA   FCBADDRH         ; TEST FOR SHARED ACCESS FILES WITH WRITE ENABLED.
D0F9:85 BB                    STA   FCBPTR+1
D0FB:A5 29                    LDA   FCBANKNM
D0FD:8D BB 14                 STA   SISFCBP
D100:A9 00                    LDA   #0
D102:85 BD                    STA   DATPTR+1         ; MARK AS NO FREE FOUND.
D104:85 0B                    STA   CNTENT
D106:85 08                    STA   TOTENT           ; ALSO, INIT COUNT OF MATCHING FILES
D108:85 BA         TSTOPN1    STA   FCBPTR           ; SAVE NEW LOW ORDER ADDRESS
D10A:A6 BD                    LDX   DATPTR+1         ; FIND OUT IF A FREE SPOT HAS BEEN FOUND YET.
D10C:D0 02   D110             BNE   TSTOPN2          ; YES, DON'T INCREMENT REFNUM (CNTENT).
D10E:E6 0B                    INC   CNTENT           ; BUMP REFNUM
D110:A0 00         TSTOPN2    LDY   #FCBREFN         ; TEST FOR IN USE FCB
D112:B1 BA                    LDA   (FCBPTR),Y       ; (NON ZERO)
D114:D0 0E   D124             BNE   CHKACTV          ; THIS FCB IS IN USE, COPARE OWNERSHIP.
D116:8A                       TXA                    ; TEST AGAIN FOR FREE FCB
D117:D0 29   D142             BNE   TSNXFCB          ; BRANCH IF A FREE SPOT HAS ALREADY BEEN FOUND.
D119:A5 BA                    LDA   FCBPTR           ; TRANSFER CURRENT POINTER SO IT MAY BE
D11B:85 BC                    STA   DATPTR           ; USED AS A FREE FCB BY OPEN.
D11D:A5 BB                    LDA   FCBPTR+1         ; HIGH BYTE ALWAYS NON ZERO.
D11F:85 BD                    STA   DATPTR+1
D121:4C 42 D1                 JMP   TSNXFCB
D124:              *
D124:        D124  CHKACTV    EQU   *                ; IF MATCHING FILE IS SWAPPED, IT DOESNT COUNT
D124:A0 1A                    LDY   #FCBSWAP
D126:B1 BA                    LDA   (FCBPTR),Y
D128:D0 18   D142             BNE   TSNXFCB          ; BRANCH IF SWAPPED
D12A:A0 06                    LDY   #FCBENTN         ; NOTE: THIS CODE DEPENDS ON THE
D12C:B1 BA         WHOWNS     LDA   (FCBPTR),Y       ; DEFINED ORDER OF FCB AND DIRECTORY
D12E:D9 B3 DB                 CMP   D.DEV-1,Y        ; *****************************
D131:D0 0F   D142             BNE   TSNXFCB          ; BRANCH IF THIS ONE HAS A DIFFERENT OWNER.
D133:88                       DEY
D134:D0 F6   D12C             BNE   WHOWNS
D136:E6 08                    INC   TOTENT           ; REPORT THIS ONE AS A CO-OWNER.
D138:A0 09                    LDY   #FCBATTR         ; NOW FIND OUT IF THIS ONE WANTS TO WRITE.
D13A:B1 BA                    LDA   (FCBPTR),Y
D13C:29 02                    AND   #WRITEN          ; IF WRITE IS NOT ENABLED THEN CONTINUE.
D13E:F0 02   D142             BEQ   TSNXFCB
D140:38                       SEC                    ; OTHERWISE, JUST SET THE CARRY TO SHOW
D141:60                       RTS                    ; THAT THE FILE CAN'T BE SHARED.
D142:              *
D142:A5 BA         TSNXFCB    LDA   FCBPTR           ; CALCULATE NEXT FCB AREA (+$20)
D144:18                       CLC
D145:69 20                    ADC   #$20
D147:90 BF   D108             BCC   TSTOPN1          ; LOOP IF NO PAGE CROSS.
D149:A6 BB                    LDX   FCBPTR+1
D14B:E6 BB                    INC   FCBPTR+1
D14D:EC 28 00                 CPX   FCBADDRH         ; HAVE WE LOOKED AT BOTH PAGES?
D150:F0 B6   D108             BEQ   TSTOPN1          ; NOPE, LOOK AT PAGE TWO.
D152:18                       CLC                    ; INDICATE NO FILES THAT SHARE HAVE WRITE ENABLED,
D153:60                       RTS
D154:              *
D154:                         CHN   READ.WRITE
```

```
D154:18            READ       CLC                    ; FIRST DETERMINE IF REQESTED
D155:A0 09                    LDY   #FCBATTR         ; READ IS LEGAL
D157:B1 BA                    LDA   (FCBPTR),Y
D159:29 01                    AND   #READEN          ; IS READ ENABLED?
D15B:D0 04   D161             BNE   READ1            ; YES, CONTINUE...
D15D:A9 00                    LDA   #ACCSERR         ; REPORT ILLEGAL ACCESS.
D15F:38                       SEC
D160:60                       RTS
D161:              *
D161:A0 12         READ1      LDY   #FCBMARK         ; GET CURRENT MARK INTO 'TPOS' AND
D163:B1 BA                    LDA   (FCBPTR),Y       ; DETERMINE IF RESULTING POSITION
D165:85 2A                    STA   TPOSLL           ; EXCEEDS CURRENT END OF FILE.
D167:65 A4                    ADC   C.BYTES
D169:8D E3 DB                 STA   SCRTCH
D16C:C8                       INY
D16D:B1 BA                    LDA   (FCBPTR),Y
D16F:85 2B                    STA   TPOSLH
D171:65 A5                    ADC   C.BYTES+1        ; (THIS WAS DONE STRAIT-LINE SINCE
D173:8D E4 DB                 STA   SCRTCH+1         ; WE'RE ADDING A TWO BYTE TO A THREE
D176:C8                       INY                    ; BYTE QUANTITY)
D177:B1 BA                    LDA   (FCBPTR),Y
D179:85 2C                    STA   TPOSHI
D17B:69 00                    ADC   #0               ; ADD IN REMAINING CARRY.
D17D:8D E5 DB                 STA   SCRTCH+2
D180:A0 17                    LDY   #FCBEOF+2        ; NOW TEST EOF AGAINST POSITION GENERATED
D182:B9 CE DB      EOFTEST    LDA   SCRTCH-FCBEOF,Y
D185:D1 BA                    CMP   (FCBPTR),Y       ; IS NEW POSITION > EOF?
D187:90 1F   D1A8             BCC   READ2            ; NO, PROCEED.
D189:D0 05   D190             BNE   ADJSTCNT         ; YES, ADJUST 'C.BYTES' REQUEST
D18B:88                       DEY
D18C:C0 14                    CPY   #FCBEOF-1        ; HAVE WE COMPARED ALL TREE BYTES?
D18E:D0 F2   D182             BNE   EOFTEST          ; NO, TEST NEXT LOWEST.
D190:        D190  ADJSTCNT   EQU   *                ; ADJUST REQUEST TO READ UP TO (BUT
D190:A0 15                    LDY   #FCBEOF          ; NOT INCLUDING) END OF FILE.
D192:B1 BA                    LDA   (FCBPTR),Y       ; RESULT= (EOF-1)-POSITION
D194:E5 2A                    SBC   TPOSLL
D196:85 A4                    STA   C.BYTES
D198:C8                       INY
D199:B1 BA                    LDA   (FCBPTR),Y
D19B:E5 2B                    SBC   TPOSLH
D19D:85 A5                    STA   C.BYTES+1
D19F:05 A4                    ORA   C.BYTES          ; IF BOTH BYTES ARE ZERO, REPORT EOF ERROR.
D1A1:D0 05   D1A8             BNE   READ2
D1A3:A9 00                    LDA   #EOFERR
D1A5:20 00 00                 JSR   SYSERR
D1A8:A5 A4         READ2      LDA   C.BYTES
D1AA:85 2D                    STA   RWREQL
D1AC:D0 09   D1B7             BNE   READ3            ; BRANCH IF READ REQUEST DEFINITELY NON-ZERO.
D1AE:C5 A5                    CMP   C.BYTES+1
D1B0:D0 05   D1B7             BNE   READ3            ; BRANCH IF READ REQUEST<>ZERO
D1B2:85 2E                    STA   RWREQH
D1B4:4C 6B D2      GORDDNE    JMP   READONE          ; DO NOTHING.
```

```
D1B7:               *
D1B7:A5 A5          READ3      LDA   C.BYTES+1
D1B9:85 2E                     STA   RWREQH
D1BB:A5 A2                     LDA   C.OUTBUF         ; MOVE POINTER TO USERS BUFFER TO BFM
D1BD:85 B0                     STA   USRBUF           ; Z-PAGE AREA.
D1BF:A2 A2                     LDX   #C.OUTBUF        ; <SRS 82.162>
D1C1:20 BD D5                  JSR   WRAPADJ          ; ADJUST FOR BANK CROSSING. <SRS 82.162>
D1C4:85 B1                     STA   USRBUF+1
D1C6:8C B1 14                  STY   SISUSRBF         ; SAVE VALID USER BUFFER ADDRESS (THAT WILL NOT CROSS BANKS)
D1C9:A0 07                     LDY   #FCBSTYP         ; NOW FIND OUT IF IT'S A TREE READ OR OTHER.
D1CB:B1 BA                     LDA   (FCBPTR),Y
D1CD:C9 04                     CMP   #TRETYP+1
D1CF:90 03   D1D4              BCC   TREAD            ; BRANCH IF A TREE FILE.
D1D1:4C 1E D3                  JMP   DREAD            ; OTHEWISE ASSUME IT'S A DIRECTORY.
D1D4:               *
D1D4:20 09 CD       TREAD      JSR   RDPOSN           ; GET DATA POINTER SET UP.
D1D7:90 03   D1DC              BCC   TREAD0           ; REPORT ANY ERRORS
D1D9:4C 64 D2                  JMP   ERRFIX1
D1DC:20 7E D2       TREAD0     JSR   PREPRW           ; TEST FOR NEWLINE, SETS UP FOR PARTIAL READ.
D1DF:20 A2 D2                  JSR   READPART         ; MOVE CURRENT DATA BUFFER CONTENTS TO USER AREA
D1E2:70 D0   D1B4              BVS   GORDDNE          ; BRANCH IF REQUEST IS SATISFIED.
D1E4:B0 EE   D1D4              BCS   TREAD            ; CARRY SET INDICATES NEWLINE IS SET.
D1E6:A5 2E                     LDA   RWREQH           ; FIND OUT HOW MANY BLOCKS ARE TO BE READ
D1E8:4A                        LSR   A                ; IF LESS THAN TWO, THEN DO IT THE SLOW WAY.
D1E9:F0 E9   D1D4              BEQ   TREAD
D1EB:85 2F                     STA   BULKCNT          ; SAVE BULK BLOCK COUNT.
D1ED:A0 08                     LDY   #FCBSTAT         ; MAKE SURE CURRENT DATA AREA
D1EF:B1 BA                     LDA   (FCBPTR),Y       ; DOESN'T NEED TO BE WRITTEN BEFORE
D1F1:29 40                     AND   #DATMOD          ; RESETTING POINTER TO READ DIRECTLY INTO
D1F3:D0 DF   D1D4              BNE   TREAD            ; USER'S AREA. BRANCH IF DATA NEED TO BE WRITTEN
D1F5:85 34                     STA   IOACCESS         ; TO FORCE FIRST CALL THRU ALL DEVICE HANDLER CHECKING.
D1F7:A5 B0                     LDA   USRBUF           ; MAKE THE DATA BUFFER THE USER'S SPACE.
D1F9:85 BC                     STA   DATPTR
D1FB:A5 B1                     LDA   USRBUF+1
D1FD:85 BD                     STA   DATPTR+1
D1FF:AD B1 14                  LDA   SISUSRBF
D202:8D BD 14                  STA   SISDATP
D205:               *
```

```
D205:20 09 CD      RDFAST     JSR   RDPOSN           ; GET NEXT BLOCK DIRECTLY INTO USER SPACE.
D208:B0 55   D25F             BCS   ERRFIX           ; BRANCH ON ANY ERROR.
D20A:E6 BD         RDFAST0    INC   DATPTR+1         ; BUMP ALL POINTERS BY 512 (ONE BLOCK)
D20C:E6 BD                    INC   DATPTR+1
D20E:C6 2E                    DEC   RWREQH
D210:C6 2E                    DEC   RWREQH
D212:E6 2B                    INC   TPOSLH
D214:E6 2B                    INC   TPOSLH
D216:D0 07   D21F             BNE   RDFAST1          ; BRANCH IF POSITION DOES NOT GET TO A 64K BOUNDARY.
D218:E6 2C                    INC   TPOSHI           ; OTHERWISE, MUST CHECK FOR A 128K BOUNDARY
D21A:A5 2C                    LDA   TPOSHI           ; SET CARRY IF MOD 128K HAS BEEN REACHED
D21C:49 01                    EOR   #1
D21E:4A                       LSR   A
D21F:C6 2F         RDFAST1    DEC   BULKCNT          ; HAVE WE READ ALL WE CAN FAST?
D221:D0 0B   D22E             BNE   RDFAST2          ; BRANCH IF MORE TO READ.
D223:20 07 D3                 JSR   FXDATPTR         ; GO FIX UP DATA POINTER TO SOS BUFFER.
D226:A5 2D                    LDA   RWREQL           ; TEST FOR END OF READ.
D228:05 2E                    ORA   RWREQH           ; ARE BOTH ZERO?
D22A:F0 3F   D26B             BEQ   READONE
D22C:D0 A6   D1D4             BNE   TREAD            ; NO, READ LAST PARTIAL BLOCK.
D22E:              *
D22E:B0 D5   D205  RDFAST2    BCS   RDFAST
D230:A5 2C                    LDA   TPOSHI           ; GET INDEX TO NEXT BLOCK ADDRESS
D232:4A                       LSR   A
D233:A5 2B                    LDA   TPOSLH
D235:6A                       ROR   A
D236:A8                       TAY                    ; INDEX TO ADDRESS IS INT(POS/512)
D237:B1 B2                    LDA   (TINDX),Y        ; GET LOW ADDRESS
D239:85 C6                    STA   BLOKNML
D23B:E6 B3                    INC   TINDX+1
D23D:D1 B2                    CMP   (TINDX),Y        ; ARE BOTH HI AND LOW ADDRESS THE SAME?
D23F:D0 08   D249             BNE   REALRD           ; NO, IT'S A REAL BLOCK ADDRESS.
D241:C9 00                    CMP   #0               ; ARE BOTH BYTES ZERO?
D243:D0 04   D249             BNE   REALRD           ; NOPE -- MUST BE REAL DATA
D245:85 34                    STA   IOACCESS         ; DON'T DO REPEATIO JUST AFTER SPARSE
D247:F0 03   D24C             BEQ   NOSTUF           ; BRANCH ALWAYS (CARRY SET)
D249:B1 B2         REALRD     LDA   (TINDX),Y        ; GET HIGH ADDRESS BYTE
D24B:18                       CLC
D24C:C6 B3         NOSTUF     DEC   TINDX+1
D24E:B0 B5   D205             BCS   RDFAST           ; BRANCH IF NO BLOCK TO READ
D250:85 C7                    STA   BLOKNMH
D252:A5 34                    LDA   IOACCESS         ; HAS FIRST CALL GONE TO DEVICE YET?
D254:F0 AF   D205             BEQ   RDFAST           ; NOPE, GO THRU NORMAL ROUTE...
D256:A5 BD                    LDA   DATPTR+1         ; RESET HI BUFFER ADDRESS FOR DEVICE HANDLER
D258:85 C3                    STA   DBUFPH
D25A:20 BD C2                 JSR   REPEATIO
D25D:90 AB   D20A             BCC   RDFAST0          ; BRANCH IF NO ERRORS.
```

```
D25F:48             ERRFIX      PHA                    ; SAVE ERROR CODE
D260:20 07 D3                   JSR   FXDATPTR         ; GO RESTORE DATA POINTERS, ETC...
D263:68                         PLA
D264:48             ERRFIX1     PHA                    ; SAVE ERROR CODE
D265:20 6B D2                   JSR   READONE          ; PASS BACK NUMBER OF BYTES ACTUALLY READ.
D268:68                         PLA
D269:38                         SEC                    ; REPORT ERROR
D26A:60                         RTS
D26B:               *
D26B:A0 00          READONE     LDY   #0               ; RETURN TOTAL NUMBER OF BYTES ACTUALLY READ
D26D:38                         SEC                    ; THIS IS DERIVED FROM C.BYTES-RWREQ
D26E:A5 A4                      LDA   C.BYTES
D270:E5 2D                      SBC   RWREQL
D272:91 A6                      STA   (C.OUTCNT),Y
D274:C8                         INY
D275:A5 A5                      LDA   C.BYTES+1
D277:E5 2E                      SBC   RWREQH
D279:91 A6                      STA   (C.OUTCNT),Y
D27B:4C 09 CD                   JMP   RDPOSN           ; LEAVE WITH VALID POSITION IN FCB.
D27E:               *
D27E:38             PREPRW      SEC                    ; ADJUST POINTER TO USER'S BUFFER TO
D27F:A5 B0                      LDA   USRBUF           ; MAKE THE TRANSFER
D281:E5 2A                      SBC   TPOSLL
D283:85 B0                      STA   USRBUF
D285:B0 02   D289               BCS   PREPRW1          ; BRANCH IF NO ADJUSTMENT TO HI ADDR. NEEDED.
D287:C6 B1                      DEC   USRBUF+1         ; NOTE: SARA ALLOWS INDIRECT FROM $101 UP
D289:A0 09          PREPRW1     LDY   #FCBATTR         ; AS LONG AS ACTUAL RESULTING ADDRESS IS >=$200
D28B:B1 BA                      LDA   (FCBPTR),Y       ; TEST FOR NEW LINE ENABLED
D28D:29 10                      AND   #NLINEN          ; SET CARRY IF IT IS.
D28F:18                         CLC
D290:F0 07   D299               BEQ   NONEWLIN         ; BRANCH IF NEWLINE IS NOT ENABLED
D292:38                         SEC
D293:A0 0A                      LDY   #FCBNEWL
D295:B1 BA                      LDA   (FCBPTR),Y       ; MOVE NEWLINE CHARACTER TO MORE
D297:85 30                      STA   NLCHAR           ; ACCESSABLE SPOT.
D299:A4 2A          NONEWLIN    LDY   TPOSLL           ; GET INDEX TO FIRST DATA
D29B:A5 BC                      LDA   DATPTR           ; RESET LOW ORDER OF POSPTR TO BEGINNING OF PAGE.
D29D:85 BE                      STA   POSPTR
D29F:A6 2D                      LDX   RWREQL           ; AND LASTLY GET LOW ORDER COUNT OF REQUESTED BYTES.
D2A1:60                         RTS                    ; RETURN STATUSES...
D2A2:               *
D2A2:8A             READPART    TXA
D2A3:D0 06   D2AB               BNE   RDPART0          ; BRANCH IF REQUEST IS NOT A EVEN PAGES
D2A5:A5 2E                      LDA   RWREQH           ; A CALL OF ZERO BYTES SHOULD NEVER GET HERE!
D2A7:F0 47   D2F0               BEQ   SETRDNE          ; BRANCH IF NOTHIN' TO DO.
D2A9:C6 2E                      DEC   RWREQH
D2AB:CA             RDPART0     DEX
D2AC:B1 BE          RDPART      LDA   (POSPTR),Y       ; MOVE DATA TO USER'S BUFFER
D2AE:91 B0                      STA   (USRBUF),Y       ; ONE BYTE AT A TIME.
D2B0:8A                         TXA                    ; NOTE: THIS ROUTINE IS CODED TO BE
D2B1:F0 19   D2CC               BEQ   ENDRQCHK         ; FASTEST WHEN NEWLINE IS DISABLED.
D2B3:B0 2A   D2DF   RDPART1     BCS   TSTNEWL          ; BRANCH IF NEW LINE NEEDS TO BE TESTED.
D2B5:CA             RDPART2     DEX
D2B6:C8                         INY                    ; PAGE CROSSED?
D2B7:D0 F3   D2AC               BNE   RDPART           ; NO. MOVE NEXT BYTE.
D2B9:A5 BF                      LDA   POSPTR+1         ; TEST FOR END OF BUFFER
```

```
D2BB:E6 B1                    INC   USRBUF+1         ; BUT FIRST ADJUST USER BUFFER POINTER
D2BD:E6 2B                    INC   TPOSLH           ; AND POSITION.
D2BF:D0 02   D2C3             BNE   RDPART3
D2C1:E6 2C                    INC   TPOSHI
D2C3:E6 BF         RDPART3    INC   POSPTR+1         ; AND SOS BUFFER HIGH ADDRESS.
D2C5:45 BD                    EOR   DATPTR+1         ; (CARRY HAS BEEN CLEVERLY UNDISTURBED.)
D2C7:F0 E3   D2AC             BEQ   RDPART           ; BRANCH IF MORE TO READ IN BUFFER.
D2C9:B8                       CLV                    ; INDICATE NOT FINISHED.
D2CA:50 27   D2F3             BVC   RDPRTDNE         ; BRANCH ALWAYS.
D2CC:              *
D2CC:A5 2E         ENDRQCHK   LDA   RWREQH
D2CE:F0 15   D2E5             BEQ   RDRQDNE          ; BRANCH IF REQEST SATISFIED.
D2D0:C8                       INY                    ; DONE WITH THIS BLOCK OF DATA?
D2D1:D0 06   D2D9             BNE   ENDRCHK1         ; NO, ADJUST HIGH BYTE OF REQUEST.
D2D3:A5 BF                    LDA   POSPTR+1         ; MAYBE- CHECK FOR END OF BLOCK BUFFER.
D2D5:45 BD                    EOR   DATPTR+1         ; (DON'T DISTURB CARRY)
D2D7:D0 02   D2DB             BNE   ENDRCHK2         ; BRANCH IF HI COUNT CAN BE DEALT WITH NEXT TIME.
D2D9:C6 2E         ENDRCHK1   DEC   RWREQH
D2DB:88            ENDRCHK2   DEY                    ; RESTORE PROPER VALUE TO 'Y'
D2DC:4C B3 D2                 JMP   RDPART1
D2DF:              *
D2DF:B1 BE         TSTNEWL    LDA   (POSPTR),Y       ; GET LAST BYTE TRANSFERED AGAIN.
D2E1:45 30                    EOR   NLCHAR           ; HAVE WE MATCHED NEWLINE CHARACTER?
D2E3:D0 D0   D2B5             BNE   RDPART2          ; NO, READ NEXT.
D2E5:C8            RDRQDNE    INY                    ; ADJUST POSITION.
D2E6:D0 08   D2F0             BNE   SETRDNE
D2E8:E6 B1                    INC   USRBUF+1         ; BUMP POINTERS.
D2EA:E6 2B                    INC   TPOSLH
D2EC:D0 02   D2F0             BNE   SETRDNE
D2EE:E6 2C                    INC   TPOSHI
D2F0:2C 06 D3      SETRDNE    BIT   SETVFLG          ; (SET V FLAG)
D2F3:84 2A         RDPRTDNE   STY   TPOSLL           ; SAVE LOW POSITION
D2F5:70 01   D2F8             BVS   RDONE1
D2F7:E8                       INX                    ; LEAVE REQUEST AS +1 FOR NEXT CALL
D2F8:86 2D         RDONE1     STX   RWREQL           ; AND REMAINDER OF REQUEST COUNT.
D2FA:08                       PHP                    ; SAVE STATUSES
D2FB:18                       CLC                    ; ADJUST USER'S LOW BUFFER ADDRESS
D2FC:98                       TYA
D2FD:65 B0                    ADC   USRBUF
D2FF:85 B0                    STA   USRBUF
D301:90 02   D305             BCC   RDPART4
D303:E6 B1                    INC   USRBUF+1         ; ADJUST HI ADDRESS AS NEEDED.
D305:28            RDPART4    PLP                    ; RESTORE RETURN STATUSES
D306:60            SETVFLG    RTS                    ; (THIS BYTE <$60> IS USED TO SET V FLAG)
D307:              *
D307:A5 BC         FXDATPTR   LDA   DATPTR           ; PUT CURRENT USER BUFFER
D309:85 B0                    STA   USRBUF           ; ADDRESS BACK TO NORMAL
D30B:A5 BD                    LDA   DATPTR+1
D30D:85 B1                    STA   USRBUF+1         ; BANK PAIR BYTE SHOULD BE MOVED ALSO.
D30F:AD BD 14                 LDA   SISDATP
D312:8D B1 14                 STA   SISUSRBF
D315:A0 0B                    LDY   #FCBBUFN         ; RESTORE BUFFER ADDRESS
D317:B1 BA                    LDA   (FCBPTR),Y
D319:A2 BC                    LDX   #DATPTR
D31B:4C 00 00                 JMP   GETBUFADR        ; END VIA CALL TO BOB'S CODE.
D31E:              *
```

```
D31E:              *
D31E:              * READ DIRECTORY FILE...
D31E:              *
D31E:20 09 CD      DREAD      JSR   RDPOSN
D321:B0 32   D355             BCS   ERRDRD           ; PASS BACK ANY ERRORS
D323:20 7E D2                 JSR   PREPRW           ; PREPARE FOR TRANSFER.
D326:20 A2 D2                 JSR   READPART         ; MOVE DATA TO USER'S BUFFER
D329:50 F3   D31E             BVC   DREAD            ; REPEAT UNTIL REQUEST IS SATISFIED.
D32B:20 6B D2                 JSR   READONE          ; UPDATE FCB AS TO NEW POSITION.
D32E:90 23   D353             BCC   DREDONE          ; BRANCH IF ALL IS WELL.
D330:C9 00                    CMP   #EOFERR          ; WAS LAST READ TO END OF FILE?
D332:38                       SEC                    ; ANTICIPATE SOME OTHER PROBLEM
D333:D0 1F   D354             BNE   DREDERR          ; BRANCH IF NOT EOF ERROR.
D335:20 54 CE                 JSR   SVMARK
D338:20 32 CE                 JSR   ZIPDATA          ; CLEAR OUT DATA BLOCK.
D33B:A0 11                    LDY   #FCBDATB+1       ; PROVIDE DUMMY BACK POINTER FOR FUTURE RE-POSITION
D33D:B1 BA                    LDA   (FCBPTR),Y       ; GET HI BYTE OF LAST BLOCK.
D33F:48                       PHA
D340:88                       DEY
D341:B1 BA                    LDA   (FCBPTR),Y       ; AND LOW BYTE.
D343:48                       PHA
D344:A9 00                    LDA   #0               ; NOW MARK CURRENT BLOCK AS IMPOSIBLE.
D346:91 BA                    STA   (FCBPTR),Y
D348:C8                       INY
D349:91 BA                    STA   (FCBPTR),Y
D34B:A8                       TAY                    ; NOW MOVE LAST BLOCK ADDRESS TO DATA BUFFER AS BACK POINTER.
D34C:68                       PLA
D34D:91 BC                    STA   (DATPTR),Y
D34F:68                       PLA
D350:C8                       INY
D351:91 BC                    STA   (DATPTR),Y
D353:18            DREDONE    CLC                    ; INDICATE NO ERROR
D354:60            DREDERR    RTS
D355:              *
D355:4C 64 D2      ERRDRD     JMP   ERRFIX1          ; REPORT HOW MUCH WE COULD TRANSFER BEFORE ERROR.
D358:              *
```

```
D358:18            WRITE      CLC                     ; FIRST DETERMINE IF REQESTED
D359:A0 09                    LDY   #FCBATTR          ; WRITE IS LEGAL
D35B:B1 BA                    LDA   (FCBPTR),Y
D35D:29 02                    AND   #WRITEN           ; IS WRITE ENABLED?
D35F:D0 04   D365             BNE   WRITE1            ; YES, CONTINUE...
D361:A9 00         ERRACCS    LDA   #ACCSERR          ; REPORT ILLEGAL ACCESS.
D363:38                       SEC
D364:60            WPERROR    RTS
D365:              *
D365:20 78 D5      WRITE1     JSR   TSTWPROT          ; OTHERWISE, MAKE SURE DEVICE IS NOT WRITE PROTECTED.
D368:B0 FA   D364             BCS   WPERROR           ; REPORT WRITE PROTECTED AND ABORT OPERATION.
D36A:              *
D36A:A0 12                    LDY   #FCBMARK          ; GET CURRENT MARK INTO 'TPOS' AND
D36C:B1 BA                    LDA   (FCBPTR),Y        ; DETERMINE IF RESULTING POSITION
D36E:85 2A                    STA   TPOSLL            ; EXCEEDS CURRENT END OF FILE.
D370:65 A4                    ADC   C.BYTES
D372:8D E3 DB                 STA   SCRTCH
D375:C8                       INY
D376:B1 BA                    LDA   (FCBPTR),Y
D378:85 2B                    STA   TPOSLH
D37A:65 A5                    ADC   C.BYTES+1         ; (THIS WAS DONE STRAIGHT-LINE SINCE
D37C:8D E4 DB                 STA   SCRTCH+1          ; WE'RE ADDING A TWO BYTE TO A THREE
D37F:C8                       INY                     ; BYTE QUANTITY)
D380:B1 BA                    LDA   (FCBPTR),Y
D382:85 2C                    STA   TPOSHI
D384:69 00                    ADC   #0                ; ADD IN REMAINING CARRY.
D386:8D E5 DB                 STA   SCRTCH+2
D389:A0 17                    LDY   #FCBEOF+2         ; NOW TEST EOF AGAINST POSITION GENERATED
D38B:B9 CE DB      WEOFTST    LDA   SCRTCH-FCBEOF,Y
D38E:D1 BA                    CMP   (FCBPTR),Y        ; IS NEW POSITION > EOF?
D390:90 19   D3AB             BCC   WRITE2            ; NO, PROCEED.
D392:D0 05   D399             BNE   WADJEOF           ; YES, ADJUST END OF FILE
D394:88                       DEY
D395:C0 14                    CPY   #FCBEOF-1         ; HAVE WE COMPARED ALL TREE BYTES?
D397:D0 F2   D38B             BNE   WEOFTST           ; NO, TEST NEXT LOWEST.
D399:18            WADJEOF    CLC                     ; ADJUST REQUEST TO WRITE UP TO (BUT
D39A:A0 15                    LDY   #FCBEOF           ; NOT INCLUDING) END OF FILE.
D39C:B1 BA         WRTADJEOF  LDA   (FCBPTR),Y        ; SAVE OLD EOF IN CASE OF LATER ERROR
D39E:99 DB DB                 STA   OLDEOF-FCBEOF,Y
D3A1:B9 CE DB                 LDA   SCRTCH-FCBEOF,Y   ; RESULT=EOF
D3A4:              *
D3A4:91 BA                    STA   (FCBPTR),Y
D3A6:C8                       INY
D3A7:C0 18                    CPY   #FCBEOF+3
D3A9:D0 F1   D39C             BNE   WRTADJEOF
D3AB:A5 A4         WRITE2     LDA   C.BYTES
D3AD:85 2D                    STA   RWREQL
D3AF:D0 09   D3BA             BNE   WRITE3            ; BRANCH IF WRITE REQUEST DEFINITELY NON-ZERO.
D3B1:C5 A5                    CMP   C.BYTES+1
D3B3:D0 05   D3BA             BNE   WRITE3            ; BRANCH IF WRITE REQUEST<>ZERO
D3B5:85 2E                    STA   RWREQH
D3B7:4C 63 D4                 JMP   WRITDONE          ; DO NOTHING.
D3BA:              *
```

```
D3BA:A5 A5         WRITE3     LDA   C.BYTES+1
D3BC:85 2E                    STA   RWREQH
D3BE:A5 A2                    LDA   C.OUTBUF         ; MOVE POINTER TO USERS BUFFER TO BFM
D3C0:85 B0                    STA   USRBUF           ; Z-PAGE AREA.
D3C2:A5 A3                    LDA   C.OUTBUF+1
D3C4:85 B1                    STA   USRBUF+1         ; (SO IT MAY BE ADJUSTED WITHOUT LOOSING
D3C6:AD A3 14                 LDA   SISOUTBF         ; ORIGINAL ADDRESS.)
D3C9:8D B1 14                 STA   SISUSRBF
D3CC:A0 07                    LDY   #FCBSTYP         ; NOW FIND OUT IF IT'S A TREE WRITE OR OTHER.
D3CE:B1 BA                    LDA   (FCBPTR),Y
D3D0:C9 04                    CMP   #TRETYP+1
D3D2:90 03   D3D7             BCC   TWRITE           ; BRANCH IF A TREE FILE.
D3D4:4C 61 D3                 JMP   ERRACCS          ; OTHEWISE RETURN AN ACCESS ERROR!
D3D7:20 09 CD      TWRITE     JSR   RDPOSN           ; READ BLOCK WE'RE
D3DA:B0 24   D400             BCS   WRITERROR
D3DC:A0 08                    LDY   #FCBSTAT
D3DE:B1 BA                    LDA   (FCBPTR),Y
D3E0:29 07                    AND   #DATALC+IDXALC+TOPALC
D3E2:F0 72   D456             BEQ   TREWRT1
D3E4:A0 00                    LDY   #0               ; FIND OUT IF ENOUGH DISK SPACE IS AVAILABLE FOR
D3E6:C8            TWRTALC    INY                    ; INDEXES AND DATA BLOCK
D3E7:4A                       LSR   A
D3E8:D0 FC   D3E6             BNE   TWRTALC
D3EA:84 04                    STY   REQL
D3EC:85 05                    STA   REQH
D3EE:20 4C C9                 JSR   TSFRBLK
D3F1:B0 0D   D400             BCS   WRITERROR        ; PASS BACK ANY ERRORS.
D3F3:A0 08                    LDY   #FCBSTAT
D3F5:B1 BA                    LDA   (FCBPTR),Y       ; NOW GET MORE SPECIFIC.
D3F7:29 04                    AND   #TOPALC          ; ARE WE LACKING A TREE TOP?
D3F9:F0 23   D41E             BEQ   TSTSAPWR         ; NO, TEST FOR LACK OF SAPLING LEVEL INDEX.
D3FB:20 CB D4                 JSR   TOPDOWN          ; GO ALLOCATE TREE TOP AND ADJUST FILE TYPE.
D3FE:90 29   D429             BCC   DBLOKALC         ; CONTINUE WITH ALLOCATION OF DATA BLOCK.
D400:48            WRITERROR  PHA                    ; SAVE ERROR
D401:A0 15                    LDY   #FCBEOF
D403:B9 DB DB      WRITERR01  LDA   OLDEOF-FCBEOF,Y
D406:91 BA                    STA   (FCBPTR),Y       ; RESTORE OLD EOF UPON ERR
D408:C8                       INY
D409:C0 18                    CPY   #FCBEOF+3
D40B:D0 F6   D403             BNE   WRITERR01
D40D:A0 12                    LDY   #FCBMARK
D40F:B9 E1 DB      WRITERR02  LDA   OLDMARK-FCBMARK,Y
D412:91 BA                    STA   (FCBPTR),Y       ; AND RESTORE OLD MARK!
D414:C8                       INY
D415:C0 15                    CPY   #FCBMARK+3
D417:D0 F6   D40F             BNE   WRITERR02
D419:68                       PLA
D41A:38                       SEC
D41B:60                       RTS                    ; ERROR RETURN
D41C:              *
D41C:50 B9   D3D7  TWRITEGO   BVC   TWRITE           ; A PIGGY-BACK BACKWARD BRANCH
D41E:              *
```

```
D41E:B1 BA         TSTSAPWR   LDA   (FCBPTR),Y       ; GET STATUS BYTE AGAIN.
D420:29 02                    AND   #IDXALC          ; DO WE NEED A SAPLING LEVEL INDEX BLOCK?
D422:F0 05   D429             BEQ   DBLOKALC         ; NO, ASSUME IT'S JUST A DATA BLOCK NEEDED.
D424:20 05 D5                 JSR   SAPDOWN          ; GO ALLOCATE AN INDEX BLOCK AND UPDATE TREE TOP.
D427:B0 D7   D400             BCS   WRITERROR        ; RETURN ANY ERRORS.
D429:20 57 D5      DBLOKALC   JSR   ALCWBLK          ; GO ALLOCATE FOR DATA BLOCK.
D42C:B0 D2   D400             BCS   WRITERROR
D42E:A5 2C                    LDA   TPOSHI           ; CALCULATE POSITION WITHIN INDEX BLOCK.
D430:4A                       LSR   A
D431:A5 2B                    LDA   TPOSLH
D433:6A                       ROR   A
D434:A8                       TAY                    ; NOW PUT BLOCK ADDRESS INTO INDEX BLOCK
D435:E6 B3                    INC   TINDX+1          ; HIGH BYTE FIRST.
D437:AD E4 DB                 LDA   SCRTCH+1
D43A:AA                       TAX
D43B:91 B2                    STA   (TINDX),Y
D43D:C6 B3                    DEC   TINDX+1          ; (RESTORE POINTER TO LOWER PAGE OF INDEX BLOCK)
D43F:AD E3 DB                 LDA   SCRTCH           ; GET LOW BLOCK ADDRESS
D442:91 B2                    STA   (TINDX),Y        ; NOW STORE LOW ADDRESS.
D444:A0 10                    LDY   #FCBDATB         ; ALSO UPDATE FILE CONTROL BLOCK TO INDICATE
D446:91 BA                    STA   (FCBPTR),Y       ; THAT THIS BLOCK IS ALLOCATED.
D448:C8                       INY
D449:8A                       TXA                    ; GET HIGH ADDRESS AGAIN.
D44A:91 BA                    STA   (FCBPTR),Y
D44C:A0 08                    LDY   #FCBSTAT
D44E:B1 BA                    LDA   (FCBPTR),Y
D450:09 80                    ORA   #IDXMOD
D452:29 F8                    AND   #$FF-DATALC-IDXALC-TOPALC ; CLEAR ALLOCATION REQUIREMENT BITS.
D454:91 BA                    STA   (FCBPTR),Y
D456:A2 B0         TREWRT1    LDX   #USRBUF          ; LOCATE POINTER TO ADJUST <SRS 82.162>
D458:20 BD D5                 JSR   WRAPADJ          ; ADJUST FOR BANK CROSSING <SRS 82.162>
D45B:20 7E D2                 JSR   PREPRW           ; WRITE ON
D45E:20 66 D4                 JSR   WRTPART
D461:50 B9   D41C             BVC   TWRITEGO
D463:4C 09 CD      WRITDONE   JMP   RDPOSN           ; UPDATE FCB WITH NEW POSITION.
D466:              *
```

```
D466:8A                WRTPART    TXA
D467:D0 06   D46F                 BNE   WRPART           ; BRANCH IF REQUEST IS NOT A EVEN PAGES
D469:A5 2E                        LDA   RWREQH           ; A CALL OF ZERO BYTES SHOULD NEVER GET HERE!
D46B:F0 3E   D4AB                 BEQ   SETWRDNE         ; DO NOTHING!
D46D:                  *
D46D:C6 2E                        DEC   RWREQH
D46F:CA                WRPART     DEX
D470:B1 B0                        LDA   (USRBUF),Y       ; MOVE DATA FROM USER'S BUFFER
D472:91 BE                        STA   (POSPTR),Y       ; ONE BYTE AT A TIME.
D474:8A                           TXA
D475:F0 16   D48D                 BEQ   ENDWQCHK
D477:C8                WRPART2    INY                    ; PAGE CROSSED?
D478:D0 F5   D46F                 BNE   WRPART           ; NO. MOVE NEXT BYTE.
D47A:A5 BF                        LDA   POSPTR+1         ; TEST FOR END OF BUFFER
D47C:E6 B1                        INC   USRBUF+1         ; BUT FIRST ADJUST USER BUFFER POINTER
D47E:E6 2B                        INC   TPOSLH           ; AND POSITION.
D480:D0 02   D484                 BNE   WRPART3
D482:E6 2C                        INC   TPOSHI
D484:E6 BF             WRPART3    INC   POSPTR+1         ; AND SOS BUFFER HIGH ADDRESS.
D486:45 BD                        EOR   DATPTR+1         ; (CARRY HAS BEEN CLEVERLY UNDISTURBED.)
D488:F0 E5   D46F                 BEQ   WRPART           ; BRANCH IF MORE TO WRITE TO BUFFER.
D48A:B8                           CLV                    ; INDICATE NOT FINISHED.
D48B:50 21   D4AE                 BVC   WRPRTDNE         ; BRANCH ALWAYS.
D48D:                  *
D48D:A5 2E             ENDWQCHK   LDA   RWREQH
D48F:F0 0F   D4A0                 BEQ   WRTRQDNE         ; BRANCH IF REQEST SATISFIED.
D491:C8                           INY                    ; ARE WE DONE WITH THIS BLOCK OF DATA?
D492:D0 06   D49A                 BNE   ENDWCHK1         ; BRANCH IF NOT.
D494:A5 BF                        LDA   POSPTR+1
D496:45 BD                        EOR   DATPTR+1         ; WHILE THIS IS REDUNDANT, IT'S NECESSARY FOR
D498:D0 02   D49C                 BNE   ENDWCHK2         ; PROPER ADJUSTMENT OF REQUEST COUNT.
D49A:C6 2E             ENDWCHK1   DEC   RWREQH           ; (NOT FINISHED- OK TO ADJUST HI BYTE.)
D49C:88                ENDWCHK2   DEY                    ; RESET MODIFIED Y
D49D:4C 77 D4                     JMP   WRPART2
D4A0:                  *
D4A0:C8                WRTRQDNE   INY                    ; AND POSITION.
D4A1:D0 08   D4AB                 BNE   SETWRDNE
D4A3:E6 B1                        INC   USRBUF+1         ; BUMP POINTERS.
D4A5:E6 2B                        INC   TPOSLH
D4A7:D0 02   D4AB                 BNE   SETWRDNE
D4A9:E6 2C                        INC   TPOSHI
D4AB:2C 06 D3          SETWRDNE   BIT   SETVFLG          ; (SET V FLAG)
D4AE:84 2A             WRPRTDNE   STY   TPOSLL           ; SAVE LOW POSITION
D4B0:86 2D                        STX   RWREQL           ; AND REMAINDER OF REQUEST COUNT.
D4B2:08                           PHP                    ; SAVE STATUSES
D4B3:A0 08                        LDY   #FCBSTAT
D4B5:B1 BA                        LDA   (FCBPTR),Y
D4B7:09 50                        ORA   #DATMOD+USEMOD
D4B9:91 BA                        STA   (FCBPTR),Y
D4BB:18                           CLC                    ; ADJUST USER'S LOW BUFFER ADDRESS
D4BC:A5 2A                        LDA   TPOSLL
D4BE:65 B0                        ADC   USRBUF
D4C0:85 B0                        STA   USRBUF
D4C2:90 02   D4C6                 BCC   WRPART4
D4C4:E6 B1                        INC   USRBUF+1         ; ADJUST HI ADDRESS AS NEEDED.
D4C6:20 F4 DD          WRPART4    JSR   FCBUSED          ; SET DIRECTORY FLUSH BIT
```

```
D4C9:28                    PLP                    ; RESTORE RETURN STATUSES
D4CA:60                    RTS
```

```
D4CB:20 13 D5      TOPDOWN    JSR   SWAPDOWN         ; FIRST MAKE CURRENT 1ST BLOCK AN ENTRY IN NEW TOP.
D4CE:B0 42   D512             BCS   TPDWNERR         ; RETURN ANY ERRORS
D4D0:A0 07                    LDY   #FCBSTYP         ; FIND OUT IF STORAGE TYPE HAS BEEN CHANGED TO 'TREE'.
D4D2:B1 BA                    LDA   (FCBPTR),Y       ; (IF NOT, ASSUME IT WAS ORIGINALLY A SEED AND
D4D4:C9 03                    CMP   #TRETYP          ; BOTH LEVELS NEED TO BE BUILT.
D4D6:F0 05   D4DD             BEQ   TOPDWN1          ; OTHERWISE, ONLY AN INDEX NEED BE ALLOCATED)
D4D8:20 13 D5                 JSR   SWAPDOWN         ; MAKE PREVIOUS SWAP A SAP LEVEL INDEX BLOCK.
D4DB:B0 35   D512             BCS   TPDWNERR
D4DD:20 57 D5      TOPDWN1    JSR   ALCWBLK          ; GET ANOTHER BLOCK ADDRESS FOR THE SAP LEVEL INDEX.
D4E0:B0 30   D512             BCS   TPDWNERR
D4E2:A5 2C                    LDA   TPOSHI           ; CALCULATE POSITION OF NEW INDEX BLOCK
D4E4:4A                       LSR   A                ; IN THE TOP OF THE TREE.
D4E5:A8                       TAY
D4E6:AD E3 DB                 LDA   SCRTCH           ; GET ADDRESS OF NEWLY ALOCATED INDEX BLOCK AGAIN
D4E9:AA                       TAX
D4EA:91 B2                    STA   (TINDX),Y
D4EC:E6 B3                    INC   TINDX+1
D4EE:AD E4 DB                 LDA   SCRTCH+1
D4F1:91 B2                    STA   (TINDX),Y        ; SAVE HI ADDRESS
D4F3:C6 B3                    DEC   TINDX+1
D4F5:A0 0F                    LDY   #FCBIDXB+1       ; MAKE NEWLY ALLOCATED BLOCK THE CURRENT INDEX BLOCK.
D4F7:91 BA                    STA   (FCBPTR),Y
D4F9:8A                       TXA
D4FA:88                       DEY
D4FB:91 BA                    STA   (FCBPTR),Y
D4FD:20 73 CF                 JSR   WFCBFST          ; SAVE NEW TOP OF TREE.
D500:B0 10   D512             BCS   TPDWNERR
D502:4C D1 C2                 JMP   ZTMPIDX          ; END BY RE-CLEARING CURRENT (NEW) INDEX BLOCK.
D505:              *
D505:A0 07         SAPDOWN    LDY   #FCBSTYP         ; FIND OUT IF WE'RE DEALING WITH A TREE
D507:B1 BA                    LDA   (FCBPTR),Y       ; OR A SIMPLE SEED.
D509:C9 01                    CMP   #SEEDTYP         ; IF SEED THEN AN ADJUSTMENT TO FILE TYPE IS NECESSARY.
D50B:F0 06   D513             BEQ   SAPDWN1          ; BRANCH IF SEED.
D50D:20 F0 CE                 JSR   RFCBFST          ; OTHERWISE READ IN TOP OF TREE.
D510:90 CB   D4DD             BCC   TOPDWN1          ; BRANCH IF NO ERROR.
D512:60            TPDWNERR   RTS                    ; RETURN ERRORS
D513:              *
```

```
D513:        D513  SAPDWN1    EQU   *                ; MAKE CURRENT SEED INTO A SAPLING
D513:              *
D513:20 57 D5      SWAPDOWN   JSR   ALCWBLK          ; ALLOCATE A BLOCK BEFORE SWAP
D516:B0 3E   D556             BCS   SWAPERR          ; RETURN ERRORS IMMEDIATELY.
D518:A0 0C                    LDY   #FCBFRST         ; GET PREVIOUS FIRST BLOCK
D51A:B1 BA                    LDA   (FCBPTR),Y       ; ADDRESS INTO INDEX BLOCK.
D51C:48                       PHA                    ; SAVE TEMPORARLY WHILE SWAPPING IN NEW TOP INDEX
D51D:AD E3 DB                 LDA   SCRTCH           ; GET NEW BLOCK ADDRESS (LOW)
D520:AA                       TAX
D521:91 BA                    STA   (FCBPTR),Y
D523:C8                       INY
D524:B1 BA                    LDA   (FCBPTR),Y
D526:48                       PHA
D527:AD E4 DB                 LDA   SCRTCH+1         ; AND HIGH ADDRESS TOO.
D52A:91 BA                    STA   (FCBPTR),Y
D52C:A0 0F                    LDY   #FCBIDXB+1       ; MAKE NEW TOP ALSO THE CURRENT INDEX IN MEMORY.
D52E:91 BA                    STA   (FCBPTR),Y
D530:8A                       TXA                    ; GET LOW ADDRESS AGAIN
D531:88                       DEY
D532:91 BA                    STA   (FCBPTR),Y
D534:A0 00                    LDY   #0               ; MAKE PREVIOUS THE FIRST ENTRY IN SUB INDEX
D536:E6 B3                    INC   TINDX+1
D538:68                       PLA
D539:91 B2                    STA   (TINDX),Y
D53B:C6 B3                    DEC   TINDX+1
D53D:68                       PLA
D53E:91 B2                    STA   (TINDX),Y
D540:20 73 CF                 JSR   WFCBFST          ; SAVE NEW FILE TOP.
D543:B0 11   D556             BCS   SWAPERR
D545:A0 07                    LDY   #FCBSTYP         ; NOW ADJUST STORAGE TYPE
D547:A9 01                    LDA   #1               ; BY ADDING 1 (THUS SEED BECOMES SAPLING BECOMES TREE)
D549:71 BA                    ADC   (FCBPTR),Y
D54B:91 BA                    STA   (FCBPTR),Y
D54D:A0 08                    LDY   #FCBSTAT
D54F:B1 BA                    LDA   (FCBPTR),Y       ; MARK STORAGE TYPE MODIFIED.
D551:09 08                    ORA   #STPMOD
D553:91 BA                    STA   (FCBPTR),Y
D555:18                       CLC                    ; RETURN 'NO ERROR' STATUS.
D556:60            SWAPERR    RTS
D557:              *
```

```
D557:20 9C CA      ALCWBLK    JSR   ALC1BLK
D55A:B0 1B   D577             BCS   ALUSERR
D55C:A0 18                    LDY   #FCBUSE
D55E:B1 BA                    LDA   (FCBPTR),Y       ; BUMP CURRENT USAGE COUNT BY 1.
D560:18                       CLC
D561:69 01                    ADC   #1
D563:91 BA                    STA   (FCBPTR),Y
D565:90 07   D56E             BCC   INCUSG1
D567:C8                       INY
D568:B1 BA                    LDA   (FCBPTR),Y
D56A:69 00                    ADC   #0
D56C:91 BA                    STA   (FCBPTR),Y
D56E:A0 08         INCUSG1    LDY   #FCBSTAT         ; MARK USAGE AS MODIFIED.
D570:B1 BA                    LDA   (FCBPTR),Y
D572:09 10                    ORA   #USEMOD
D574:91 BA                    STA   (FCBPTR),Y
D576:18                       CLC                    ; INDICATE NO ERROR
D577:60            ALUSERR    RTS                    ; ALL DONE
D578:              *
D578:A0 08         TSTWPROT   LDY   #FCBSTAT         ; CHECK FOR A 'NEVER BEEN MODIFIED' CONDITION
D57A:B1 BA                    LDA   (FCBPTR),Y       ; GET STATUS BYTE
D57C:29 F0                    AND   #USEMOD+DATMOD+IDXMOD+EOFMOD
D57E:18                       CLC                    ; ANTICIPATE WRITE OK
D57F:D0 F6   D577             BNE   ALUSERR          ; ORDINARY RTS
D581:A0 01                    LDY   #FCBDEVN         ; GET FILE'S DEVICE NUMBER
D583:B1 BA                    LDA   (FCBPTR),Y
D585:85 35                    STA   DEVNUM           ; GET CURRENT STATUS OF BLOCK DEVICE
D587:A9 02         TWRPROT1   LDA   #STATCMD
D589:85 C0                    STA   DHPCMD
D58B:A9 00                    LDA   #STATSUB         ; STORE SUB COMMAND OF STATUS CALL
D58D:85 C2                    STA   DSTATREQ
D58F:A9 BC                    LDA   #>TWRCODE
D591:85 C3                    STA   DSTATBFL         ; FETCH RETURN CODE IN SCRATCH AREA
D593:A9 D5                    LDA   #<TWRCODE
D595:85 C4                    STA   DSTATBFH
D597:A9 00                    LDA   #0               ; MAKE SURE REGULAR RAM IS SELECTED (NO BANKS)
D599:8D C4 14                 STA   SISDSTAT
D59C:8D 00 00                 STA   SERR             ; CLEAR GLOBAL ERROR FLAG
D59F:A5 35                    LDA   DEVNUM           ; SET UP LAST PARM
D5A1:85 C1                    STA   UNITNUM          ; FOR DEVICE CALL
D5A3:20 00 00                 JSR   DMGR             ; MAKE THE EXTERNAL CALL
D5A6:B0 08   D5B0             BCS   WPROTRET         ; RETURN ANY SPECIFIC ERRORS
D5A8:AD BC D5                 LDA   TWRCODE          ; GET STATUS BYTE
D5AB:4A                       LSR   A                ; SHIFT WRITE PROTECT STATE INTO CARRY
D5AC:4A                       LSR   A
D5AD:A9 00                    LDA   #XNOWRITE        ; ANTICIPATE WRITE PROTECTED.
D5AF:60                       RTS                    ; CARRY IS INDETERMINATE
D5B0:        D5B0  WPROTRET   EQU   *
D5B0:C9 00                    CMP   #XDISKSW         ; IF EXPLICITLY DISK SWITCH
D5B2:D0 05   D5B9             BNE   WPROT1           ; BRANCH IF XNODRIVE OR XNOWRITE
D5B4:8D BB D5                 STA   DSWGLOB          ; IF DISKSW, FLAG UNTIL ENTIRE OPERATION IS COMPLETE
D5B7:18                       CLC
D5B8:60                       RTS                    ; DISKSWITCH DOESNT SET CARRY
D5B9:38            WPROT1     SEC
D5BA:60                       RTS
D5BB:        0001  DSWGLOB    DS    1                ; DISK SWITCH GLOBAL
```

```
D5BC:        0001  630 TWRCODE    DS    1                ; A RARE EMBEDDED TEMP STORE
D5BD:              631 *
```

```
D5BD:              *
D5BD:              * MEMORY 'WRAP-AROUND' ADJUST ROUTINE.  THIS ROUTINE ADJUSTS
D5BD:              * ADDRESSES THAT CROSS BANK PAIR BOUNDARIES.  ON ENTRY, X CONTAINS
D5BD:              * THE OFFSET OF THE ZERO PAGE EXTENDED POINTER TO BE ADJUSTED.
D5BD:              * ON EXIT, THE POINTER WILL HAVE BEEN ADJUSTED, IF NECESSARY,
D5BD:              * AND THE ASSOCIATED X-BYTE WILL ALSO HAVE BEEN ADJUSTED.
D5BD:              * ONLY ADDRESSES IN THE RANGE $8200-$8E00 WILL BE ADJUSTED.
D5BD:              *
D5BD:              * UPON EXIT, A CONTAINS HIGH BYTE OF ADDRESS & Y CONTAINS UPDATED X-BYTE.
D5BD:              * THIS ROUTINE LEAVES X UNCHANGED.
D5BD:              *
D5BD:B5 01         WRAPADJ    LDA   1,X              ; GET HIGH ADDRESS BYTE <SRS 82.162>
D5BF:BC 01 14                 LDY   SISTER+1,X       ; CHECK X-BYTE <SRS 82.162>
D5C2:10 10   D5D4             BPL   WRAPDNE          ; NOT AN EXTENDED ADDRESS. <SRS 82.162>
D5C4:C9 82                    CMP   #$82             ; DOES IT NEED UPDATING? <SRS 82.162>
D5C6:90 0C   D5D4             BCC   WRAPDNE          ; NO <SRS 82.162>
D5C8:C0 8F                    CPY   #$8F             ; SPECIAL BANK? <SRS 82.162>
D5CA:B0 08   D5D4             BCS   WRAPDNE          ; NO <SRS 82.162>
D5CC:29 7F                    AND   #$7F             ; ADJUST THE ADDRESS <SRS 82.162>
D5CE:95 01                    STA   1,X              ; UPDATE <SRS 82.162>
D5D0:FE 01 14                 INC   SISTER+1,X       ; INCREMENT X-BYTE <SRS 82.162>
D5D3:C8                       INY                     ; UPDATE Y ALSO <SRS 82.162>
D5D4:              *
D5D4:60            WRAPDNE    RTS                    ; RETURN VALID HIGH ADDRESS AND BANK BYTE.
D5D5:              *
D5D5:                         CHN   CLOSE.EOF
```

```
D5D5:               *
D5D5:               *
D5D5:A5 A1          CLOSE      LDA   C.REFNUM         ; CLOSE ALL?
D5D7:D0 40   D619              BNE   CLOSE1           ; NO, JUST ONE OF 'EM
D5D9:8D 18 D6                  STA   CFERR            ; CLEAR GLOBAL CLOSE ERROR
D5DC:20 81 D7                  JSR   GFCBADR          ; SET UP POINTER TO FCB
D5DF:A9 00          CLOSALL    LDA   #0               ; BEGIN AT THE BEGINNING.
D5E1:85 BA          CLSALL1    STA   FCBPTR           ; SAVE CURRENT LOW BYTE OF POINTER
D5E3:A0 1B                     LDY   #FCBLEVL         ; FETCH THE LEVEL AT WHICH
D5E5:B1 BA                     LDA   (FCBPTR),Y       ; FILE WAS OPENED
D5E7:CD 00 00                  CMP   LEVEL            ; TEST AGAINST CURRENT GLOBAL LEVEL
D5EA:90 14   D600              BCC   NXTCLOS          ; DONT CLOSE IF FILES LEVEL IS < GLOBAL LEVEL
D5EC:A0 00                     LDY   #FCBREFN         ; INDEX TO REFERENCE NUMBER
D5EE:B1 BA                     LDA   (FCBPTR),Y       ; IS THIS REFERENCE FILE OPEN?
D5F0:F0 0E   D600              BEQ   NXTCLOS          ; NO, TRY NEXT.
D5F2:20 7F D6                  JSR   FLUSH2           ; CLEAN IT OUT...
D5F5:B0 4F   D646              BCS   CLOSERR          ; RETURN FLUSH ERRORS
D5F7:20 1E D6                  JSR   CLOSE2           ; UPDATE FCB & VCB
D5FA:A4 A1                     LDY   C.REFNUM
D5FC:F0 02   D600              BEQ   NXTCLOS          ; NO ERR IF CLOSE ALL
D5FE:B0 46   D646              BCS   CLOSERR
D600:A5 BA          NXTCLOS    LDA   FCBPTR           ; BUMP POINTER TO NEXT FILE CONTROL BLOCK.
D602:18                        CLC
D603:69 20                     ADC   #$20
D605:90 DA   D5E1              BCC   CLSALL1          ; BRANCH IF WITHIN SAME PAGE.
D607:A5 BB                     LDA   FCBPTR+1
D609:E6 BB                     INC   FCBPTR+1         ; BUMP TO NEXT PAGE.
D60B:CD 28 00                  CMP   FCBADDRH         ; HAVE WE CHECKED BOTH PAGES?
D60E:F0 CF   D5DF              BEQ   CLOSALL          ; YES, RETURN NO ERROR.
D610:18                        CLC
D611:AD 18 D6                  LDA   CFERR            ; ON FINAL CLOSE OF CLOSE ALL REPORT LOGGED ERRORS
D614:F0 01   D617              BEQ   C3               ; BRANCH IF NO ERRORS
D616:38                        SEC
D617:60             C3         RTS
D618:               *
D618:               *
D618:        0001   CFERR      DS    1                ; GLOBAL ERROR FLAG FOR FLUSH AND CLOSE ALL
D619:               *
D619:               *
D619:20 87 D6       CLOSE1     JSR   FLUSH1           ; FLUSH FILE FIRST (INCLUDING UPDATING BIT MAP)
D61C:B0 28   D646              BCS   CLOSERR
D61E:A0 0B          CLOSE2     LDY   #FCBBUFN
D620:B1 BA                     LDA   (FCBPTR),Y
D622:20 00 00                  JSR   RELBUF
D625:B0 1F   D646              BCS   CLOSERR
D627:A9 00                     LDA   #0
D629:A0 00                     LDY   #FCBREFN
D62B:91 BA                     STA   (FCBPTR),Y
D62D:C8                        INY                    ; BUMP TO 'FCBDEVN'
D62E:B1 BA                     LDA   (FCBPTR),Y
D630:85 35                     STA   DEVNUM           ; GO LOOK FOR ASSOCIATED VCB.
D632:20 48 C8                  JSR   DEVVCB
D635:A6 B6                     LDX   VCBPTR           ; GET VCBPTR
D637:DE 1E 11                  DEC   VCB+VCBOPNC,X    ; INDICATE ONE LESS FILE OPEN.
D63A:D0 08   D644              BNE   CLOSEND          ; BRANCH IF THAT WASN'T THE LAST...
D63C:BD 11 11                  LDA   VCB+VCBSTAT,X
```

```
D63F:29 7F                        AND   #$7F             ; STRIP 'FILES OPEN' BIT
D641:9D 11 11                     STA   VCB+VCBSTAT,X
D644:18               CLOSEND     CLC
D645:60                           RTS
D646:4C 78 D7         CLOSERR     JMP   GLBERR           ; DON'T REPORT CLOSALL ERR NOW
D649:                 *
```

```
D649:               *
D649:A5 A1          FLUSH      LDA   C.REFNUM          ; FLUSH ALL?
D64B:D0 3A   D687              BNE   FLUSH1            ; NO, JUST ONE OF 'EM
D64D:8D 18 D6                  STA   CFERR             ; CLEAR GLOBAL FLUSH ERROR
D650:20 81 D7                  JSR   GFCBADR           ; SET UP POINTER TO FCB
D653:A9 00          FLSHALL    LDA   #0                ; BEGIN AT THE BEGINNING.
D655:85 BA          FLSHAL1    STA   FCBPTR            ; SAVE CURRENT LOW BYTE OF POINTER
D657:A0 00                     LDY   #FCBREFN          ; INDEX TO REFERENCE NUMBER
D659:B1 BA                     LDA   (FCBPTR),Y        ; IS THIS REFERENCE FILE OPEN?
D65B:F0 07   D664              BEQ   NXFLUSH           ; NO, TRY NEXT.
D65D:20 7F D6                  JSR   FLUSH2            ; CLEAN IT OUT...
D660:B0 1A   D67C              BCS   FLSHERR           ; RETURN ANY ERRORS
D662:               *
D662:B0 E2   D646              BCS   CLOSERR
D664:A5 BA          NXFLUSH    LDA   FCBPTR            ; BUMP POINTER TO NEXT FILE CONTROL BLOCK.
D666:18                        CLC
D667:69 20                     ADC   #$20
D669:90 EA   D655              BCC   FLSHAL1           ; BRANCH IF WITHIN SAME PAGE.
D66B:A5 BB                     LDA   FCBPTR+1
D66D:E6 BB                     INC   FCBPTR+1          ; BUMP TO NEXT PAGE.
D66F:CD 28 00                  CMP   FCBADDRH          ; HAVE WE CHECKED BOTH PAGES?
D672:F0 DF   D653              BEQ   FLSHALL           ; YES, RETURN NO ERROR.
D674:18             FLUSHEND   CLC
D675:AD 18 D6                  LDA   CFERR             ; ON LAST FLUSH OF A FLUSH(0)
D678:F0 01   D67B              BEQ   F3                ; BRANCH IF NO LOGGED ERRORS
D67A:38                        SEC                     ; REPORT ERROR NOW
D67B:60             F3         RTS
D67C:4C 78 D7       FLSHERR    JMP   GLBERR            ; FLUSH ALL OR ONE?
D67F:               *
D67F:20 A0 BE       FLUSH2     JSR   FNDFCBUF          ; MUST SET UP ASSOCIATED VCB AN BUFFER LOCATIONS FIRST.
D682:90 0D   D691              BCC   FLUSH2A           ; BRANCH IF NO ERROR ENCOUNTERED.
D684:4C 78 D7                  JMP   GLBERR            ; CHECK FOR CLOSE OR FLUSH ALL
D687:               *
D687:A9 00          FLUSH1     LDA   #0                ; CLEAR
D689:8D 18 D6                  STA   CFERR             ; GLOBAL ERROR FOR NORMAL REFNUM FLUSH
D68C:20 75 BE                  JSR   FINDFCB           ; SET UP POINTER TO FCB USER REFERENCES
D68F:B0 EB   D67C              BCS   FLSHERR           ; RETURN ANY ERRORS
D691:A0 09          FLUSH2A    LDY   #FCBATTR          ; TEST TO SEE IF FILE IS
D693:B1 BA                     LDA   (FCBPTR),Y        ; MODIFIED. FIRST TEST WRITE ENABLED.
D695:29 02                     AND   #WRITEN
D697:F0 DB   D674              BEQ   FLUSHEND          ; BRANCH IF 'READ ONLY'
D699:A0 1C                     LDY   #FCBDIRTY         ; SEE IF EOF HAS BEEN MODIFIED
D69B:B1 BA                     LDA   (FCBPTR),Y
D69D:30 08   D6A7              BMI   FLUSH2B           ; BRANCH IF IT HAS
D69F:A0 08                     LDY   #FCBSTAT          ; NOW TEST FOR DATA MODIFIED.
D6A1:B1 BA                     LDA   (FCBPTR),Y        ; (IN OTHER WORDS: WAS FILE ACTUALLY
D6A3:29 70                     AND   #USEMOD+EOFMOD+DATMOD ; WRITTEN TO WHILE IT'S BEEN OPEN?)
D6A5:F0 CD   D674              BEQ   FLUSHEND          ; BRANCH IF FILE NOT MODIFIED.
D6A7:20 87 D5       FLUSH2B    JSR   TWRPROT1          ; DISK SWITCH CHECKING
D6AA:AD BB D5                  LDA   DSWGLOB
D6AD:F0 04   D6B3              BEQ   FLUSH2C           ; BRANCH IF NO SWITCH
D6AF:A9 00                     LDA   #XDISKSW
D6B1:38                        SEC
D6B2:60                        RTS                     ; FORCES A VERIFIED RETRY
D6B3:A0 08          FLUSH2C    LDY   #FCBSTAT          ; NOW TEST FOR DATA MODIFIED.
D6B5:B1 BA                     LDA   (FCBPTR),Y
```

```
D6B7:29 40                     AND   #DATMOD          ; DOES CURRENT DATA BUFFER NEED TO BE
D6B9:F0 05   D6C0              BEQ   FLUSH3           ; WRITTEN? BRANCH IF NOT.
D6BB:20 84 CF                  JSR   WFCBDAT          ; IF SO, GO WRITE IT STUPID!
D6BE:B0 BC   D67C              BCS   FLSHERR
D6C0:A0 08          FLUSH3     LDY   #FCBSTAT         ; CHECK TO SEE IF THE INDEX BLOCK (TREE FILES ONLY)
D6C2:B1 BA                     LDA   (FCBPTR),Y       ; NEEDS TO BE WRITTEN.
D6C4:29 80                     AND   #IDXMOD
D6C6:F0 05   D6CD              BEQ   FLUSH4           ; BRANCH IF NOT...
D6C8:20 94 CF                  JSR   WFCBIDX
D6CB:B0 AF   D67C              BCS   FLSHERR          ; RETURN ANY ERRORS.
```

```
D6CD:                  *
D6CD:A0 06             FLUSH4     LDY   #FCBENTN         ; NOW PREPARE TO UPDATE DIRECTORY
D6CF:B1 BA             OWNRMOV    LDA   (FCBPTR),Y       ; NOTE: THIS CODE DEPENDS ON THE
D6D1:99 B3 DB                     STA   D.DEV-FCBDEVN,Y  ; DEFINED ORDER OF THE FILE CONTROL
D6D4:88                           DEY                    ; BLOCK AND THE TEMPORARY DIRECTORY AREA IN 'WORKSPC'!
*************
D6D5:C0 00                        CPY   #FCBDEVN-1
D6D7:D0 F6   D6CF                 BNE   OWNRMOV
D6D9:AD B5 DB                     LDA   D.HEAD           ; READ IN THE DIRECTORY HEADER FOR THIS FILE
D6DC:85 C6                        STA   BLOKNML
D6DE:AD B6 DB                     LDA   D.HEAD+1
D6E1:85 C7                        STA   BLOKNMH
D6E3:AD B4 DB                     LDA   D.DEV
D6E6:85 35                        STA   DEVNUM
D6E8:20 58 CC                     JSR   RDGBUF           ; READ IT INTO THE GENERAL PURPOSE BUFFER
D6EB:B0 8F   D67C                 BCS   FLSHERR          ; BRANCH IF ERROR.
D6ED:20 2A C6                     JSR   MOVHED0          ; MOVE HEADER INFO.
D6F0:AD B7 DB                     LDA   D.ENTBLK         ; GET ADDRESS OF DIRECTORY BLOCK THAT
D6F3:AC B8 DB                     LDY   D.ENTBLK+1       ; CONTAINS THE FILE ENTRY.
D6F6:CD B5 DB                     CMP   D.HEAD           ; TEST TO SEE IF IT'S THE SAME BLOCK THAT
D6F9:D0 05   D700                 BNE   FLSHEBLK         ; THE HEADER IS IN. BRANCH IF NOT.
D6FB:CC B6 DB                     CPY   D.HEAD+1
D6FE:F0 07   D707                 BEQ   FLUSH5           ; BRANCH IF HEADER BLOCK = ENTRY BLOCK.
D700:85 C6             FLSHEBLK   STA   BLOKNML
D702:84 C7                        STY   BLOKNMH
D704:20 58 CC                     JSR   RDGBUF           ; GET BLOCK WITH FILE ENTRY IN GENERAL BUFFER.
D707:20 D6 C3          FLUSH5     JSR   ENTCALC          ; SET UP POINTER TO ENTRY
D70A:20 85 C4                     JSR   MOVENTRY         ; MOVE ENTRY TO TEMP ENTRY BUFFER IN 'WORKSPC'
D70D:A0 18                        LDY   #FCBUSE          ; UPDATE 'BLOCKS USED' COUNT.
D70F:B1 BA                        LDA   (FCBPTR),Y
D711:8D CD DB                     STA   DFIL+D.USAGE
D714:C8                           INY
D715:B1 BA                        LDA   (FCBPTR),Y
D717:8D CE DB                     STA   DFIL+D.USAGE+1   ; HI BYTE TOO...
D71A:A0 15                        LDY   #FCBEOF          ; AND MOVE IN END OF FILE MARK WHETHER
D71C:B1 BA             EOFUPDTE   LDA   (FCBPTR),Y       ; WE NEED TO OR NOT.
D71E:99 BA DB                     STA   DFIL+D.EOF-FCBEOF,Y
D721:C8                           INY                    ; MOVE ALL THREE BYTES.
D722:C0 18                        CPY   #FCBEOF+3
D724:D0 F6   D71C                 BNE   EOFUPDTE
D726:A0 0C                        LDY   #FCBFRST         ; ALSO MOVE IN THE ADDRESS OF
D728:B1 BA                        LDA   (FCBPTR),Y       ; THE FILE'S FIRST BLOCK SINCE
D72A:C8                           INY                    ; IT MIGHT HAVE CHANGED SINCE THE FILE
D72B:8D CB DB                     STA   DFIL+D.FRST      ; FIRST OPENED.
D72E:B1 BA                        LDA   (FCBPTR),Y
D730:8D CC DB                     STA   DFIL+D.FRST+1
```

```
D733:A0 07                     LDY   #FCBSTYP         ; AND THE LAST THING TO UPDATE IS
D735:B1 BA                     LDA   (FCBPTR),Y       ; THE STORAGE TYPE.
D737:0A                        ASL   A                ; (SHIFT IT INTO THE HI NIBBLE)
D738:0A                        ASL   A
D739:0A                        ASL   A
D73A:0A                        ASL   A
D73B:8D E3 DB                  STA   SCRTCH
D73E:AD BA DB                  LDA   DFIL+D.STOR      ; GET OLD TYPE BYTE (IT MIGHT BE THE SAME)
D741:29 0F                     AND   #$F              ; STRIP OFF OLD TYPE
D743:0D E3 DB                  ORA   SCRTCH           ; ADD IN THE NEW TYPE,
D746:8D BA DB                  STA   DFIL+D.STOR      ; AND PUT IT AWAY.
D749:20 F0 C3                  JSR   DREVISE          ; GO UPDATE DIRECTORY!
D74C:B0 2A   D778              BCS   FLUSHERR
D74E:A0 1C                     LDY   #FCBDIRTY        ; MARK
D750:B1 BA                     LDA   (FCBPTR),Y       ; FCB/DIRECTORY
D752:29 7F                     AND   #$FF-FCBMOD      ; AS
D754:91 BA                     STA   (FCBPTR),Y       ; UNDIRTY
D756:A2 00                     LDX   #0               ; NOW CHECK TO SEE IF A BIT MAP
D758:AD B4 DB                  LDA   D.DEV            ; IS LYING AROUND THAT SHOULD BE WRITTEN.
D75B:C5 1D                     CMP   BMADEV           ; IS IT IN MAP BUFFER A?
D75D:F0 06   D765              BEQ   BMAPUP           ; YES, PUT IT ON THE DISK IF NECESSARY.
D75F:A2 06                     LDX   #BMTABSZ         ; SET INDEX TO BIT MAP TABLE 'B'
D761:C5 23                     CMP   BMBDEV           ; NO, WHAT ABOUT BIT MAP BUFFER B?
D763:D0 11   D776              BNE   FLSHEND1         ; NOPE, ALL DONE.
D765:B5 1C          BMAPUP     LDA   BMASTAT,X        ; TEST TO SEE IF IT'S BEEN MODIFIED.
D767:10 0D   D776              BPL   FLSHEND1         ; NOPE, ALL DONE AS I SAID.
D769:86 1A                     STX   BMTAB
D76B:20 4F CC                  JSR   WRTBMAP          ; GO PUT IT AWAY.
D76E:B0 08   D778              BCS   FLUSHERR
D770:A6 1A                     LDX   BMTAB            ; MARK MAP AS UPDATED
D772:A9 00                     LDA   #0
D774:95 1C                     STA   BMASTAT,X
D776:18             FLSHEND1   CLC
D777:60                        RTS
D778:        D778   FLUSHERR   EQU   *                ; DROP INTO GLBERR
D778:               *
D778:        D778   GLBERR     EQU   *                ; REPORT ERROR IMMEDIATELY
D778:               * ONLY IF NOT A CLOSE ALL OR FLUSH ALL
D778:A6 A1                     LDX   C.REFNUM
D77A:D0 04   D780              BNE   GLBERR1          ; NOT AN 'ALL' SO REPORT NOW
D77C:18                        CLC
D77D:8D 18 D6                  STA   CFERR            ; SAVE FOR LATER
D780:60             GLBERR1    RTS
D781:               *
D781:               *
D781:A5 29          GFCBADR    LDA   FCBANKNM         ; GET BANK THAT FCB IS IN
D783:8D BB 14                  STA   SISFCBP
D786:AD 28 00                  LDA   FCBADDRH         ; AND HIGH BYTE ADDRESS OF FILE CONTORL BLOCK.
D789:85 BB                     STA   FCBPTR+1
D78B:60                        RTS                    ; SILLY THAT IT'S SO SHORT...
D78C:               *
D78C:A9 00          SETERR     LDA   #ACCSERR
D78E:38                        SEC
D78F:60             EOFRETN    RTS
```

```
D790:              *
D790:A0 07         SETEOF     LDY   #FCBSTYP         ; ONLY KNOW HOW TO MOVE EOF OF TREE TYPE
D792:B1 BA                    LDA   (FCBPTR),Y
D794:C9 04                    CMP   #TRETYP+1
D796:B0 F4   D78C             BCS   SETERR           ; BRANCH IF OTHER THAN TREE
D798:A0 09                    LDY   #FCBATTR         ; NOW CHECK TO INSURE WRITE IS ENABLED.
D79A:B1 BA                    LDA   (FCBPTR),Y
D79C:29 02                    AND   #WRITEN          ; CAN WE SET NEW EOF?
D79E:F0 EC   D78C             BEQ   SETERR           ; NOPE, ACCESS ERROR.
D7A0:20 78 D5                 JSR   TSTWPROT         ; FIND OUT IF MOD IS POSIBLE (HARDWARE WRITE PROTECT)
D7A3:B0 E7   D78C             BCS   SETERR
D7A5:A0 17                    LDY   #FCBEOF+2        ; SAVE OLD EOF
D7A7:A2 02                    LDX   #2               ; SO IT CAN BE SEEN
D7A9:B1 BA         SETSAVE    LDA   (FCBPTR),Y       ; WHETHER BLOCKS NEED
D7AB:9D F0 DB                 STA   OLDEOF,X         ; TO BE RELEASED
D7AE:88                       DEY                    ; UPON
D7AF:CA                       DEX                    ; CONTRACTION
D7B0:10 F7   D7A9             BPL   SETSAVE          ; ALL THREE BYTES OF THE EOF
D7B2:20 CD CC                 JSR   ADJMARK          ; GET ADJUSTED END OF FILE ACCORDING TO 'C.BASE' INTO TPOS.
D7B5:B0 D8   D78F             BCS   EOFRETN          ; RETURN ANY ERROR IMMEDIATELY
D7B7:A2 02                    LDX   #2
D7B9:B5 2A         NEOFPOS    LDA   TPOSLL,X         ; POSITION MARK TO NEW EOF
D7BB:95 A3                    STA   C.NEWEOF,X
D7BD:CA                       DEX
D7BE:10 F9   D7B9             BPL   NEOFPOS
D7C0:A0 14                    LDY   #FCBMARK+2       ; FIND OUT IF EOF < MARK.
D7C2:A2 02                    LDX   #2
D7C4:B1 BA         NEOFTST    LDA   (FCBPTR),Y
D7C6:D5 A3                    CMP   C.NEWEOF,X       ; COMPARE UNTIL NOT EQUAL OR CARRY CLEAR
D7C8:90 0B   D7D5             BCC   SETEOF1          ; BRANCH IF EOF>MARK
D7CA:D0 04   D7D0             BNE   SETEOF0          ; BRANCH IF EOF<MARK
D7CC:88                       DEY
D7CD:CA                       DEX
D7CE:10 F4   D7C4             BPL   NEOFTST          ; LOOP ON ALL THREE BYTES
D7D0:20 09 CD      SETEOF0    JSR   RDPOSN           ; READ IN NEW POSITION.
D7D3:B0 BA   D78F             BCS   EOFRETN          ; RETURN ANY ERRORS.
D7D5:A2 02         SETEOF1    LDX   #2
D7D7:A0 17                    LDY   #FCBEOF+2        ; MOVE NEW EOF TO FCB.
D7D9:B5 A3         SETEOF2    LDA   C.NEWEOF,X
D7DB:91 BA                    STA   (FCBPTR),Y
D7DD:88                       DEY
D7DE:CA                       DEX
D7DF:10 F8   D7D9             BPL   SETEOF2          ; MOVE ALL THREE BYTES.
D7E1:20 F4 DD                 JSR   FCBUSED          ; MARK FCB AS DIRTY (FOR FLUSH)
D7E4:              *
D7E4:A2 02                    LDX   #2               ; POINT TO THIRD BYTE
D7E6:BD F0 DB      PURTEST    LDA   OLDEOF,X         ; SEE IF EOF MOVED BACKWARDS
D7E9:D5 A3                    CMP   C.NEWEOF,X       ; SO BLOCKS CAN
D7EB:90 05   D7F2             BCC   PURTEST1         ; BE RELEASED (BRANCH IF NOT)
D7ED:D0 09   D7F8             BNE   PURGE            ; BRANCH IF BLOCKS TO BE RELEASED
D7EF:CA                       DEX
D7F0:10 F4   D7E6             BPL   PURTEST          ; ALL THREE BYTES
D7F2:4C 76 D7      PURTEST1   JMP   FLSHEND1         ; NEW EOF NOT SMALLER
D7F5:4C 7B D8      TRELEAS1   JMP   TRELEASE         ; OVERFLOW PREVENTER
D7F8:              *
D7F8:A0 07         PURGE      LDY   #FCBSTYP         ; FIND OUT WHAT TYPE OF TREE
```

```
D7FA:B1 BA                    LDA   (FCBPTR),Y       ; TO PERFORM THE PROPER
D7FC:C9 01                    CMP   #SEEDTYP         ; STYLE OF BLOCK RELEASE
D7FE:F0 3B   D83B             BEQ   EOFOUT           ; SEED DON'T DEALLOCATE
D800:C9 03                    CMP   #TRETYP          ; FULL TREE?
D802:F0 F1   D7F5             BEQ   TRELEAS1         ; BRANCH IF YES
D804:              *
D804:              * IF WE GET HERE, WE ARE RELEASING
D804:              * BLOCKS AT THE END OF A SAPLING FILE: CALCULATE CORRECT POSITION
D804:              * WITHIN THE INDEX BLOCK AND ALLOW SUBROUTINE
D804:              * PURGE LATTER BLOCKS TO DEALLOCATE
D804:              * ALL THE DATA BLOCKS THAT FOLLOW
D804:              *
D804:20 7F CB                 JSR   FNDBMAP          ; REFRESH THE RIGHT MAP FOR THIS VOLUME
D807:A6 2C                    LDX   TPOSHI           ; PRELOAD
D809:A4 2B                    LDY   TPOSLH           ;  THE THREE EOF
D80B:A5 2A                    LDA   TPOSLL           ;    BYTES
D80D:D0 0A   D819             BNE   PUR1               ; BRANCH IF NO BOUNDARY ADJUSTMENT NEEDED
D80F:C0 00                    CPY   #0
D811:D0 05   D818             BNE   PUR2             ; MIDDLE BYTE ZERO MEANS NO CARRY
D813:E0 00                    CPX   #0               ; ALL BYTES ZERO??
D815:F0 02   D819             BEQ   PUR1             ; BRANCH IF YES
D817:CA                       DEX
D818:              *
D818:              * THESE LINES IF CODE, SOMEWHAT CRYPTIC,
D818:              * CALCULATE THE POINT AT WHICH THE
D818:              * LAST BLOCK CONTAINING THE LAST BIT
D818:              * OF DATUM
D818:              *
D818:              * THE FOLLOWING IS ROUGHLY A /512
D818:              * ALGORITHM
D818:              *
D818:88            PUR2       DEY
D819:8A            PUR1       TXA
D81A:4A                       LSR   A
D81B:98                       TYA
D81C:6A                       ROR   A
D81D:              *
D81D:20 3D D8                 JSR   PURLBLKS         ; MAKES A GOOD PTR TO DO THE RELEASING
D820:A0 08                    LDY   #FCBSTAT         ; MARK INDEX BLOCK
D822:B1 BA                    LDA   (FCBPTR),Y       ; AS DIRTY
D824:09 80                    ORA   #IDXMOD
D826:91 BA                    STA   (FCBPTR),Y
D828:AD 79 D8                 LDA   PURUSE           ; INDICATE NEW NUMBER OF BLOCKS USED
D82B:18                       CLC
D82C:69 02                    ADC   #2               ; ACCOUNT FOR CARDINAL AND INDEX
D82E:A0 18                    LDY   #FCBUSE
D830:91 BA                    STA   (FCBPTR),Y       ; FILE LOW BYTE
D832:C8                       INY
D833:A9 00                    LDA   #0               ; ANTICIPATE <257 BLOCKS
D835:90 02   D839             BCC   PURHI
D837:A9 01                    LDA   #1               ; >256 BLOCKS IN FILE
D839:91 BA         PURHI      STA   (FCBPTR),Y       ; HIGH BYTE BLOCKS USED
D83B:18            EOFOUT     CLC
D83C:60                       RTS                    ; NO ERRORS POSSIBLE
D83D:              *
D83D:        D83D  PURLBLKS   EQU   *                ; PURGE LATTER BLOCKS
```

```
D83D:              * INPUT ARG: A REGISTER CONTAINING
D83D:              * POINTER TO CURRENT DATA BLOCK WITHIN THE
D83D:              * CURRENT INDEX BLOCK (TINDX)
D83D:              * DEALLOCATE ALL LEGAL BLOCKS AFTER
D83D:              * THE A REGISTER PTR. NO ERRORS POSSIBLE
D83D:              *
D83D:A8                       TAY                    ; MAKE PROPER INDEX
D83E:8C 79 D8                 STY   PURUSE           ; INDICATES NUMBER OF BLOCKS IN USE IN FILE
D841:C8            PURLOOP    INY                    ; POINT TO A PTR TO DATA BLK TO DEALLOCATE
D842:F0 34   D878             BEQ   PURLRTS          ; NO MORE BLOCKS IN INDEX
D844:E6 B3                    INC   TINDX+1          ; GET HIGH PART OF BLOCK ADDR
D846:B1 B2                    LDA   (TINDX),Y
D848:AA                       TAX                    ; X IS A PASSING PARM
D849:A9 00                    LDA   #0               ; TELL INDEX BLOCK THAT THE DATA
D84B:91 B2                    STA   (TINDX),Y        ; BLOCK IS NOW FREE
D84D:8A                       TXA
D84E:C6 B3                    DEC   TINDX+1          ; AND LOW PART
D850:11 B2                    ORA   (TINDX),Y
D852:F0 ED   D841             BEQ   PURLOOP          ; INDICATED ADDR WAS ZERO-ZERO
D854:B1 B2                    LDA   (TINDX),Y        ; A REG IS ANOTHER PASSING PARM
D856:48                       PHA
D857:A9 00                    LDA   #0
D859:91 B2                    STA   (TINDX),Y        ; AND SET LOW DATA ADDR AS FREED
D85B:68                       PLA
D85C:8C 7A D8                 STY   PURPLACE         ; TEMP STORAGE
D85F:20 04 CA                 JSR   DEALLOC          ; DEALLOCATE BLOCK (ADDR: A (LOW), X ( HIGH)
D862:A0 14                    LDY   #VCBTFRE
D864:18                       CLC
D865:B1 B6                    LDA   (VCBPTR),Y       ; ADJUST NUMBER OF FREE BLOCKS ON VOLUME
D867:69 01                    ADC   #1
D869:91 B6                    STA   (VCBPTR),Y
D86B:C8                       INY
D86C:B1 B6                    LDA   (VCBPTR),Y       ; HIGH BYTE OF TOTAL FREE
D86E:69 00                    ADC   #0
D870:91 B6                    STA   (VCBPTR),Y
D872:AC 7A D8                 LDY   PURPLACE
D875:4C 41 D8                 JMP   PURLOOP
D878:60            PURLRTS    RTS
D879:        0001  PURUSE     DS    1                ; CURRENT NUMBER OF BLOCKS USED
D87A:        0001  PURPLACE   DS    1                ; CURRENT PLACE IN RELEASE-BLOCK CYCLE
D87B:        D87B  TRELEASE   EQU   *
D87B:4C 3B D8                 JMP   EOFOUT           ; RELEASE TWO LEVEL TREE CODE GOES HERE
D87E:              *
D87E:A0 15         GETEOF     LDY   #FCBEOF          ; INDEX TO END OF FILE MARK
D880:A2 00                    LDX   #0               ; WE'VE GOT INDIRECT BOTH WAYS (IN & OUT)
D882:B1 BA         OUTEOF     LDA   (FCBPTR),Y
D884:81 A2                    STA   (C.OUTEOF,X)
D886:C8                       INY
D887:C0 18                    CPY   #FCBEOF+3
D889:F0 22   D8AD             BEQ   OFFRTS           ; BRANCH IF ALL THREE BYTES TRANSFERED.
D88B:E6 A2                    INC   C.OUTEOF         ; BUMP USER'S POINTER.
D88D:D0 F3   D882             BNE   OUTEOF
D88F:E6 A3                    INC   C.OUTEOF+1
D891:D0 EF   D882             BNE   OUTEOF           ; BRANCH ALWAYS
D893:              *
D893:                         CHN   DESTROY
```

```
D893:               *
D893:A0 09          NEWLINE    LDY   #FCBATTR         ; ADJUST NEWLINE STATUS FOR OPEN FILE.
D895:A5 A2                     LDA   C.ISNEWL         ; ON OR OFF?
D897:10 0E   D8A7              BPL   OFFNEWL          ; BRANCH IF NEW LINE IS TO BE CLEARED.
D899:A9 10                     LDA   #NLINEN
D89B:11 BA                     ORA   (FCBPTR),Y       ; SET NEW LINE BIT IN ATTRIBUTES
D89D:91 BA                     STA   (FCBPTR),Y
D89F:A0 0A                     LDY   #FCBNEWL         ; AND MOVE IN NEW 'NEW-LINE' BYTE.
D8A1:A5 A3                     LDA   C.NEWL
D8A3:91 BA                     STA   (FCBPTR),Y
D8A5:18                        CLC
D8A6:60                        RTS                    ; NO ERROR POSSIBLE.
D8A7:               *
D8A7:A9 EF          OFFNEWL    LDA   #$FF-NLINEN
D8A9:31 BA                     AND   (FCBPTR),Y
D8AB:91 BA                     STA   (FCBPTR),Y       ; CLEAR NEW-LINE BIT.
D8AD:18             OFFRTS     CLC                    ; THE NEW LINE CHARACTER DOES'T MATTER...
D8AE:60                        RTS
```

```
D8AF:               *
D8AF:20 80 C4       GETINFO    JSR   FINDFILE         ; LOOK FOR FILE THEY WANT OT KNOW ABOUT.
D8B2:90 37   D8EB              BCC   GTINFO1          ; BRANCH IF NO ERRORS.
D8B4:C9 00                     CMP   #BADPATH         ; WAS IT A ROOT DIRECTORY FILE?
D8B6:38                        SEC                    ; (IN CASE OF NO MATCH)
D8B7:D0 56   D90F              BNE   GINFOERR
D8B9:A9 F0                     LDA   #$F0
D8BB:8D BA DB                  STA   DFIL+D.STOR      ; FOR GET INFO, REPORT PROPER STORAGE TYPE
D8BE:A9 00                     LDA   #0               ; FORCE A COUNT OF FREE BLOCKS.
D8C0:85 04                     STA   REQL
D8C2:85 05                     STA   REQH
D8C4:20 4C C9                  JSR   TSFRBLK          ; (RETURNS IF IMMEDIATELY IF COUNT HAS PREVIOUSLY BEEN TAKEN)
D8C7:A0 15                     LDY   #VCBTFRE+1
D8C9:B1 B6                     LDA   (VCBPTR),Y       ; RETURN TOTAL BLOCKS AND TOTAL IN USE.
D8CB:85 05                     STA   REQH             ; FIRST TRANSFER 'FREE' BLOCKS TO ZPAGE FOR LATER SUBTRACT
D8CD:88                        DEY
D8CE:B1 B6                     LDA   (VCBPTR),Y       ; TO DETERMINE THE 'USED' COUNT
D8D0:85 04                     STA   REQL
D8D2:88                        DEY
D8D3:B1 B6                     LDA   (VCBPTR),Y       ; TRANSFER TO 'D.' TABLE AS AUX I.D.
D8D5:8D DA DB                  STA   DFIL+D.AUXID+1   ; (TOTAL BLOCK COUNT IS CONSIDERED AUX I.D. FOR THE VOLUME)
D8D8:AA                        TAX
D8D9:88                        DEY
D8DA:B1 B6                     LDA   (VCBPTR),Y
D8DC:8D D9 DB                  STA   DFIL+D.AUXID
D8DF:38                        SEC                    ; NOW SUBTRACT AND REPORT THE NUMBER OF BLOCKS 'IN USE'
D8E0:E5 04                     SBC   REQL
D8E2:8D CD DB                  STA   DFIL+D.USAGE
D8E5:8A                        TXA
D8E6:E5 05                     SBC   REQH
D8E8:8D CE DB                  STA   DFIL+D.USAGE+1
D8EB:A0 00          GTINFO1    LDY   #0               ; TRANSFER BYTES FROM THERE INTERNAL ORDER TO CALL SPEC VIA
'INFTABL' TRANSLATION
D8ED:B9 58 D9       GTINFO2    LDA   INFTABL,Y
D8F0:10 11   D903              BPL   GTINFO3          ; BRANCH IF THIS IS DATA IS VALID AS IS.
D8F2:29 7F                     AND   #$7F             ; IS THIS THE 4TH BYTE OF THE EOF PARAMETER?
D8F4:F0 11   D907              BEQ   GTINFO4          ; YES, AND IT'S ALWAYS A ZERO.
D8F6:C9 01                     CMP   #D.STOR+1        ; IS THIS THE STORAGE TYPE BYTE?
D8F8:D0 14   D90E              BNE   GINFOEND         ; NO, IT'S THE END OF INFO THAT CAN BE RETURNED.
D8FA:AD BA DB                  LDA   DFIL+D.STOR      ; GET STORAGE TYPE
D8FD:4A                        LSR   A
D8FE:4A                        LSR   A
D8FF:4A                        LSR   A
D900:4A                        LSR   A                ; MAKE IT A VALUE 1-$F BY SHIFTING OUT FILE NAME LENGTH.
D901:10 04   D907              BPL   GTINFO4          ; BRANCH ALWAYS
D903:               *
D903:AA             GTINFO3    TAX                    ; USE AS OFFSET INTO 'D.' TABLE.
D904:BD BA DB                  LDA   DFIL,X
D907:91 A3          GTINFO4    STA   (C.FILIST),Y     ; PASS TO USER'S BUFFER
D909:C8                        INY
D90A:C4 A5                     CPY   C.FILSTLN        ; HAS REQUEST BEEN FILLED?
D90C:D0 DF   D8ED              BNE   GTINFO2          ; NO, PASS NEXT
D90E:18             GINFOEND   CLC                    ; INDICATE NO ERRORS
D90F:60             GINFOERR   RTS
D910:               *
D910:               *
```

```
D910:               *
D910:20 80 C4       SETINFO    JSR   FINDFILE         ; FIND WHAT USER WANTS...
D913:B0 6F   D984              BCS   SINFOERR         ; RETURN ANY FAILURE.
D915:A5 A5                     LDA   C.FILSTLN        ; TEST FOR NUL CHANGE
D917:F0 21   D93A              BEQ   SINFEND          ; BRANCH IF NOTHING TO CHANGE.
D919:A0 00                     LDY   #0               ; INIT POINTER TO USER SUPPLIED LIST.
D91B:B1 A3                     LDA   (C.FILIST),Y     ; FETCH FILE ATTRIBUTES
D91D:29 1C                     AND   #$1C             ; FORBIDDEN BITS? <SRS 82.162>
D91F:F0 04   D925              BEQ   SETINF1          ; NO
D921:A9 00                     LDA   #ACCSERR         ; YES
D923:38                        SEC
D924:60                        RTS                    ; RETURN AN ERROR
D925:AD 00 00       SETINF1    LDA   BACKMASK         ; GET CURRENT BACKMASK <SRS 82.162>
D928:               * BACKUP KNOWS HOW TO RESET THIS BIT. <SRS 82.162>
D928:8D 57 D9                  STA   BKBITFLG         ; BIT (USED BY DREVISE)
D92B:BE 58 D9       SETINF1X   LDX   INFTABL,Y        ; GET INDEX INTO CORESPONDING 'D.' TABLE
D92E:30 0D   D93D              BMI   SETINF2          ; BRANCH IF WE'VE REACHED STORAGE TYPE PARAMETER
D930:B1 A3                     LDA   (C.FILIST),Y
D932:9D BA DB                  STA   DFIL,X
D935:C8                        INY                    ; HAS USER'S REQUEST BEEN SATISFIED?
D936:C4 A5                     CPY   C.FILSTLN
D938:D0 F1   D92B              BNE   SETINF1X         ; NO, MOVE NEXT BYTE.
D93A:4C F0 C3       SINFEND    JMP   DREVISE          ; GO UPDATE DIRECTORY WITH CURRENT TIME.
D93D:               *
D93D:A4 A5          SETINF2    LDY   C.FILSTLN        ; TEST TO SEE IF USER WANTS HIS TIME STAMP ADDED
D93F:C0 0F                     CPY   #$F              ; (LIST MUST BE AT LEAST $F BYTES LONG)
D941:90 F7   D93A              BCC   SINFEND          ; NO PUT CURRENT TIME INSTEAD.
D943:A0 0B                     LDY   #$B              ; MOVE IN THE NEXT GROUP OF BYTES
D945:BE 58 D9       SETINF3    LDX   INFTABL,Y
D948:30 0A   D954              BMI   SINFEND1
D94A:B1 A3                     LDA   (C.FILIST),Y
D94C:9D BA DB                  STA   DFIL,X
D94F:C8                        INY
D950:C4 A5                     CPY   C.FILSTLN        ; SATISFACTION YET?
D952:D0 F1   D945              BNE   SETINF3          ; NOPE, KEEP EM PUMPIN'
D954:4C FE C3       SINFEND1   JMP   DREVISE1
D957:               *
D957:        0001   BKBITFLG   DS    1                ; FOR TURNING OFF BACKUP BIT
D958:               *
D958:               *
D958:1E 10 1F 20    INFTABL    DFB   D.ATTR,D.FILID,D.AUXID,D.AUXID+1
D95C:81 15 16 17               DFB   D.STOR+1+$80,D.EOF,D.EOF+1,D.EOF+2 ; (D.STOR=0 THUS D.STOR+1 WAS NECESSARY)
D960:80 13 14 21               DFB   $80,D.USAGE,D.USAGE+1,D.MODDT ; (THE $80 IS FOR THE FOURTH BYTE OF EOF)
D964:22 23 24 FF               DFB   D.MODDT+1,D.MODTM,D.MODTM+1,$FF ; TABLE ALWAYS ENDS IN $FF
```

```
D968:              *
D968:20 93 C4      RENAME     JSR   LOOKFILE         ; LOOK FOR SOURCE (ORIGINAL) FILE.
D96B:90 32   D99F             BCC   RNAME0           ; BRANCH IF FOUND.
D96D:C9 00                    CMP   #BADPATH         ; TRYING TO RENAME A VOLUME?
D96F:D0 48   D9B9             BNE   RNAMERR          ; NO, RETURN OTHER ERROR.
D971:20 41 DA                 JSR   RENPATH          ; SYNTAX NEW NAME.
D974:B0 43   D9B9             BCS   RNAMERR
D976:A5 B4                    LDA   WRKPATH          ; FIND OUT IF ONLY ROOTNAME FOR NEW NAME
D978:C5 B0                    CMP   PATHNML
D97A:D0 72   D9EE             BNE   RNBADPTH         ; NOT SINGLE NAME, RETURN ERROR!
D97C:A0 11                    LDY   #VCBSTAT         ; TEST FOR OPEN FILES BEFORE CHANGING
D97E:B1 B6                    LDA   (VCBPTR),Y
D980:10 03   D985             BPL   RNAMEVOL         ; BRANCH IF VOLUME NOT BUSY
D982:A9 00                    LDA   #FILBUSY
D984:        D984  SINFOERR   EQU   *
D984:60                       RTS                    ; (CARRY IS SET)
D985:A0 00         RNAMEVOL   LDY   #0               ; GET NEWNAME'S LENGTH.
D987:B1 B4                    LDA   (WRKPATH),Y
D989:A8                       TAY
D98A:09 F0                    ORA   #$F0             ; (ROOT FILE STORAGE TYPE)
D98C:20 33 DA                 JSR   MVROTNAM         ; UPDATE ROOT DIRECTORY.
D98F:B0 28   D9B9             BCS   RNAMERR
D991:A0 00                    LDY   #0
D993:B1 B4                    LDA   (WRKPATH),Y      ; UPDATE VCB ALSO.
D995:A8                       TAY
D996:B1 B4         RNMEVOL    LDA   (WRKPATH),Y
D998:91 B6                    STA   (VCBPTR),Y
D99A:88                       DEY
D99B:10 F9   D996             BPL   RNMEVOL
D99D:18                       CLC
D99E:60                       RTS
D99F:              *
D99F:20 41 DA      RNAME0     JSR   RENPATH          ; SET UP AND SYNTAX NEW NAME.
D9A2:B0 15   D9B9             BCS   RNAMERR
D9A4:A0 00                    LDY   #0               ; VERIFY THAT BOTH NAMES HAVE SAME ROOT.
D9A6:B1 B0                    LDA   (PATHNML),Y
D9A8:A8                       TAY
D9A9:B1 B0         TSTSMROT   LDA   (PATHNML),Y      ; COMPARE NEWNAME'S ROOT NAME WITH
D9AB:D1 B6                    CMP   (VCBPTR),Y       ; OLD NAME'S VOLUME NAME.
D9AD:D0 3F   D9EE             BNE   RNBADPTH         ; RETURN 'BADPATH' IF NOT SAME VOLUME.
D9AF:88                       DEY
D9B0:10 F7   D9A9             BPL   TSTSMROT         ; (TEST SAME 'ROT')
D9B2:20 93 C4                 JSR   LOOKFILE         ; TEST FOR DUPLICATE FILE NAME.
D9B5:B0 04   D9BB             BCS   TSTFNF1          ; BRANCH IF ERROR TO TEST FOR FILE NOT FOUND.
D9B7:A9 00                    LDA   #DUPERR          ; TELL USER THAT NEW NAME ALREADY EXISTS.
D9B9:38            RNAMERR    SEC
D9BA:60                       RTS
```

```
D9BB:C9 00         TSTFNF1    CMP   #FNFERR          ; WAS IT A VALID FILE NOT FOUND?
D9BD:D0 FA   D9B9             BNE   RNAMERR          ; NO, RETURN OTHER ERROR CODE.
D9BF:A2 02                    LDX   #2               ; NOW MOVE NEW NAME'S OWNERSHIP (DIRECTORY HEADER) I.D.
D9C1:BD B4 DB      SVENEWID   LDA   D.DEV,X          ; THIS CONSISTS OF THE UNIT NUMBER,
D9C4:95 31                    STA   NPATHDEV,X       ; AND THE ADDRESS OF THE DIRECTORY THE FILE
D9C6:CA                       DEX                    ; WASN'T FOUND IN. LOGIC BY NEGATION...
D9C7:10 F8   D9C1             BPL   SVENEWID
D9C9:20 D5 BC                 JSR   SETPATH          ; NOW SYNTAX THE PATHNAME OF THE FILE TO BE CHANGED.
D9CC:B0 EB   D9B9             BCS   RNAMERR
D9CE:20 80 C4                 JSR   FINDFILE         ; GET ALL THE INFO ON THIS ONE.
D9D1:B0 E6   D9B9             BCS   RNAMERR
D9D3:20 F6 D0                 JSR   TSTOPEN          ; DON'T ALLOW RENAME TO OCCUR IF FILE IS IN USE.
D9D6:A9 00                    LDA   #FILBUSY         ; ANTICIPATE ERROR
D9D8:B0 DF   D9B9             BCS   RNAMERR
D9DA:AD D8 DB                 LDA   DFIL+D.ATTR      ; TEST BIT THAT SAYS IT'S OK TO RENAME
D9DD:29 40                    AND   #RENAMEN
D9DF:D0 04   D9E5             BNE   RNAME1           ; BRANCH IF IT'S ALRIGHT TO RENAME.
D9E1:A9 00                    LDA   #ACCSERR         ; OTHERWISE REPORT ILLEGAL ACCESS.
D9E3:38                       SEC
D9E4:60                       RTS
D9E5:              *
D9E5:A2 02         RNAME1     LDX   #2               ; NOW TEST TO SEE IF NEW PATHNAME FITS IN THE
D9E7:BD B4 DB      SAMOWNR    LDA   D.DEV,X          ; SAME DIRECTORY FILE.
D9EA:D5 31                    CMP   NPATHDEV,X
D9EC:F0 04   D9F2             BEQ   RNAME2
D9EE:A9 00         RNBADPTH   LDA   #BADPATH         ; TELL USER THAT PATHNAMES INCOMPATABLE.
D9F0:38                       SEC
D9F1:60                       RTS
D9F2:              *
D9F2:CA            RNAME2     DEX                    ; TEST ALL THREE BYTES.
D9F3:10 F2   D9E7             BPL   SAMOWNR
D9F5:20 41 DA                 JSR   RENPATH          ; WELL... SINCE BOTH NAMES WOULD GO INTO THE
D9F8:B0 BF   D9B9             BCS   RNAMERR          ; DIRECTORY, RE-SYNTAX THE NEW NAME TO GET LOCAL NAME ADDRESS.
D9FA:98                       TYA                    ; (Y CONTAINS THE LOCAL NAME LENGTH+1)
D9FB:F0 F1   D9EE             BEQ   RNBADPTH         ; REPORT ERROR IF LENGTH INFO NOT IMMEDIATELY AVAILABLE.
D9FD:88                       DEY                    ; (REMOVE THE +1)
D9FE:B1 B4         RNAME3     LDA   (WRKPATH),Y      ; MOVE LOCAL NAME TO DIR ENTRY WORKSPACE.
DA00:99 BA DB                 STA   DFIL+D.STOR,Y
DA03:88                       DEY
DA04:D0 F8   D9FE             BNE   RNAME3
DA06:AD BA DB                 LDA   DFIL+D.STOR      ; PRESERVE FILE STORAGE TYPE.
DA09:29 F0                    AND   #$F0             ; STRIP OFF OLD NAME LENGTH.
DA0B:AA                       TAX
DA0C:11 B4                    ORA   (WRKPATH),Y      ; ADD IN NEW NAME'S LENGTH
DA0E:8D BA DB                 STA   DFIL+D.STOR
DA11:E0 D0                    CPX   #DIRTYP*16       ; THAT FILE MUST BE CHANGED ALSO.
DA13:D0 1B   DA30             BNE   RNAMDONE         ; BRANCH IF NOT DIRECTORY TYPE.
```

```
DA15:AD CB DB                 LDA   DFIL+D.FRST      ; READ IN FIRST (HEADER) BLOCK OF SUB DIRECTORY
DA18:85 C6                    STA   BLOKNML
DA1A:AD CC DB                 LDA   DFIL+D.FRST+1
DA1D:85 C7                    STA   BLOKNMH
DA1F:20 58 CC                 JSR   RDGBUF
DA22:B0 95   D9B9             BCS   RNAMERR          ; REPORT ERRORS
DA24:A0 00                    LDY   #0               ; CHANGE THE HEADER'S NAME TO MATCH THE OWNER'S NEW NAME.
DA26:B1 B4                    LDA   (WRKPATH),Y      ; GET LOCAL NAME LENGTH AGAIN
DA28:A8                       TAY
DA29:09 E0                    ORA   #HEDTYP*16       ; ASSUME IT'S A HEADER.
DA2B:20 33 DA                 JSR   MVROTNAM
DA2E:B0 89   D9B9             BCS   RNAMERR
DA30:4C FE C3      RNAMDONE   JMP   DREVISE1         ; END BY UPDATING ALL PATH DIRECTORIES
DA33:              *
DA33:              *
DA33:8D 04 12      MVROTNAM   STA   GBUF+4
DA36:B1 B4         MVHEDNAM   LDA   (WRKPATH),Y
DA38:99 04 12                 STA   GBUF+4,Y
DA3B:88                       DEY
DA3C:D0 F8   DA36             BNE   MVHEDNAM
DA3E:4C 54 CC                 JMP   WRTGBUF          ; WRITE CHANGED HEADER BLOCK.
DA41:              *
DA41:              *
DA41:A5 A3         RENPATH    LDA   C.NWPATH         ; GET ADDRESS TO NEW PATHNAME.
DA43:85 B2                    STA   TPATH
DA45:A5 A4                    LDA   C.NWPATH+1       ; SET UP FOR SYNTAXING ROUTINE (SYNPATH).
DA47:85 B3                    STA   TPATH+1
DA49:AD A4 14                 LDA   SSNWPATH         ; (MOVE BYTE FOR SISTER PAGE, TOO.)
DA4C:8D B3 14                 STA   SISTPATH
DA4F:4C E3 BC                 JMP   SYNPATH          ; GO SYNTAX IT. (RETURNS LAST LOCAL NAME LENGTH IN Y).
DA52:              *
DA52:              *
DA52:A0 00         DEALBLK    LDY   #0               ; BEGIN AT THE BEGINNING.
DA54:84 0E         DALBLK1    STY   SAPTR            ; SAVE CURRENT INDEX.
DA56:B9 00 12                 LDA   GBUF,Y           ; GET ADDRESS (LOW) OF BLOCK TO BE DEALLOCATED.
DA59:D9 00 13                 CMP   GBUF+$100,Y      ; TEST FOR NUL BLOCK.
DA5C:D0 04   DA62             BNE   DALBLK2          ; BRANCH IF NOT NUL.
DA5E:C9 00                    CMP   #0
DA60:F0 0A   DA6C             BEQ   DALBLK3          ; SKIP IT IF NUL.
DA62:BE 00 13      DALBLK2    LDX   GBUF+$100,Y      ; GET THE REST OF THE BLOCK ADDRESS.
DA65:20 04 CA                 JSR   DEALLOC          ; FREE IT UP ON VOLUME BIT MAP.
DA68:B0 06   DA70             BCS   DALBLKERR        ; RETURN ANY ERROR.
DA6A:A4 0E                    LDY   SAPTR            ; GET INDEX TO SAPLING LEVEL INDEX BLOCK AGAIN.
DA6C:C8            DALBLK3    INY                    ; POINT AT NEXT BLOCK ADDRESS.
DA6D:D0 E5   DA54             BNE   DALBLK1          ; BRANCH IF MORE TO DEALLOCATE (OR TEST).
DA6F:18                       CLC                    ; INDICATE NO ERROR.
DA70:60            DALBLKERR  RTS
DA71:              *
DA71:              *
```

```
DA71:              *
DA71:20 80 C4      DESTROY    JSR   FINDFILE         ; LOOK FOR FILE TO BE WIPED OUT.
DA74:B0 4B   DAC1             BCS   DESTERR          ; PASS BACK ANY ERROR.
DA76:20 F6 D0                 JSR   TSTOPEN          ; IS THIS FILE OPEN?
DA79:A5 08                    LDA   TOTENT
DA7B:F0 04   DA81             BEQ   DSTROY1          ; BRANCH IF FILE NOT OPEN.
DA7D:A9 00                    LDA   #FILBUSY
DA7F:38                       SEC                    ; INFORM USER THAT FILE CAN'T BE DESTORYED AT THIS TIME.
DA80:60                       RTS
DA81:              *
DA81:A9 00         DSTROY1    LDA   #0               ; FORCE PROPER FREE COUNT IN VOLUME.
DA83:85 04                    STA   REQL             ; (NO DISK ACCESS OCCURS IF ALREADY PROPER)
DA85:85 05                    STA   REQH
DA87:20 4C C9                 JSR   TSFRBLK
DA8A:90 05   DA91             BCC   DSTROY2
DA8C:C9 00                    CMP   #OVRERR          ; WAS IT JUST A FULL DISK?
DA8E:38                       SEC
DA8F:D0 30   DAC1             BNE   DESTERR          ; NOPE, REPORT ERROR.
DA91:              *
DA91:AD D8 DB      DSTROY2    LDA   DFIL+D.ATTR      ; MAKE SURE IT'S OK TO DESTROY THIS FILE.
DA94:29 80                    AND   #DSTROYEN
DA96:D0 05   DA9D             BNE   DSTROY3          ; BRANCH IF OK.
DA98:A9 00                    LDA   #ACCSERR         ; TELL USER IT'S NOT KOSHER.
DA9A:20 00 00                 JSR   SYSERR           ; (RETURNS TO CALLER OF DESTORY)
DA9D:              *
DA9D:20 87 D5      DSTROY3    JSR   TWRPROT1         ; BEFORE GOING THRU DEALLOCATION,
DAA0:B0 1F   DAC1             BCS   DESTERR          ; TEST FOR WRITE PROTECTED HARDWARE.
DAA2:AD BA DB                 LDA   DFIL+D.STOR      ; FIND OUT WHICH STORAGE TYPE.
DAA5:29 F0                    AND   #$F0             ; STRIP OFF NAME LENGTH.
DAA7:C9 40                    CMP   #TRETYP+1*$10    ; IS IT A SEED, SAPLING, OR TREE?
DAA9:90 03   DAAE             BCC   DSTREE           ; BRANCH IF IT IS.
DAAB:4C 4F DB                 JMP   DSTDIR           ; OTHERWISE TEST FOR DIRECTORY DESTROY.
DAAE:              *
DAAE:20 0A CB      DSTREE     JSR   GTTINDX          ; GET A BIT MAP BUFFER AND TEMPORARY INDEX BUFFER.
DAB1:B0 0E   DAC1             BCS   DESTERR
DAB3:AD BA DB                 LDA   DFIL+D.STOR      ; GET STORAGE TYPE AGAIN
DAB6:29 F0                    AND   #$F0
DAB8:C9 30                    CMP   #TRETYP*$10      ; IS THIS A TREE (FULL 2-LEVEL)?
DABA:D0 06   DAC2             BNE   DSTSAP           ; NO, TEST FOR SAPLING.
DABC:20 90 CC                 JSR   RDFRST           ; READ IN ROOT INDEX FOR THIS FILE.
DABF:90 18   DAD9             BCC   DSTRE2           ; BRANCH IF ALL IS WELL.
DAC1:60            DESTERR    RTS                    ; OTHERWISE RETURN ERROR.
DAC2:              *
DAC2:C9 20         DSTSAP     CMP   #SAPTYP*$10      ; IS IT A SAPLING
DAC4:D0 4D   DB13             BNE   DSTLAST          ; NO, JUST DEALLOCATE FIRST (AND ONLY) BLOCK.
DAC6:20 D1 C2                 JSR   ZTMPIDX          ; CLEAR OUT TEMPORARY INDEX BUFFER.
DAC9:AD CB DB                 LDA   DFIL+D.FRST      ; MAKE THIS SAP LOOK LIKE A TREE...
DACC:A0 00                    LDY   #0               ; THIS IS DONE BY PLACING THE FIRST BLOCK ADDRESS
DACE:91 B2                    STA   (TINDX),Y        ; IN THE TEMP (TOP) INDEX BUFFER AS
DAD0:E6 B3                    INC   TINDX+1
DAD2:AD CC DB                 LDA   DFIL+D.FRST+1    ; A SUB INDEX WOULD APPEAR.
DAD5:91 B2                    STA   (TINDX),Y
DAD7:C6 B3                    DEC   TINDX+1
DAD9:A0 00         DSTRE2     LDY   #0               ; BEGIN SCAN OF TOP LEVEL INDEX AT ZERO.
DADB:84 0F         DSTNXT     STY   TREPTR           ; SAVE POINTER TO TREE LEVEL.
DADD:B1 B2                    LDA   (TINDX),Y        ; GET BLOCK ADDRESS OF A SUB INDEX BLOCK
```

```
DADF:E6 B3                    INC   TINDX+1          ; (TEST FOR NUL BLOCK)
DAE1:D1 B2                    CMP   (TINDX),Y
DAE3:D0 04   DAE9             BNE   DSTRE3           ; BRANCH IF WE'VE GOT AN BLOCK TO DEALLOCATE.
DAE5:C9 00                    CMP   #0               ; IS ENTIRE ADDRESS ZERO?
DAE7:F0 07   DAF0             BEQ   DSTRE4           ; YES, DO NEXT. (CARRY SET)
DAE9:18                DSTRE3     CLC                    ; INDICATE THERE IS A BLOCK OF INDEXES TO FREE UP.
DAEA:85 C6                    STA   BLOKNML
DAEC:B1 B2                    LDA   (TINDX),Y        ; GET HI ADDRESS TOO.
DAEE:85 C7                    STA   BLOKNMH
DAF0:C6 B3             DSTRE4     DEC   TINDX+1          ; (RESTORE PROPER ADDRESS FOR BUFFER)
DAF2:B0 1C   DB10             BCS   DSTNXT1          ; BRANCH IF NO SUB INDEX.
DAF4:20 58 CC                 JSR   RDGBUF           ; USE GENERAL BUFFER FOR SUB INDEX BUFFER.
DAF7:B0 C8   DAC1             BCS   DESTERR
DAF9:20 52 DA                 JSR   DEALBLK          ; GO FREE UP BLOCKS IN SUB INDEX
DAFC:B0 C3   DAC1             BCS   DESTERR
DAFE:A4 0F                    LDY   TREPTR           ; AND FREE UP SUB INDEX BLOCK TOO.
DB00:E6 B3                    INC   TINDX+1
DB02:B1 B2                    LDA   (TINDX),Y
DB04:AA                       TAX
DB05:C6 B3                    DEC   TINDX+1
DB07:B1 B2                    LDA   (TINDX),Y
DB09:20 04 CA                 JSR   DEALLOC
DB0C:B0 B3   DAC1             BCS   DESTERR
DB0E:A4 0F                    LDY   TREPTR
DB10:C8                DSTNXT1    INY                    ; HAVE ALL SUB INDEXES BEEN LOCATED?
DB11:D0 C8   DADB             BNE   DSTNXT           ; NO, DO NEXT...
DB13:AD CB DB          DSTLAST    LDA   DFIL+D.FRST      ; DEALLOCATE FIRST BLCOK OF FILE.
DB16:AE CC DB                 LDX   DFIL+D.FRST+1
DB19:20 04 CA                 JSR   DEALLOC
DB1C:B0 A3   DAC1             BCS   DESTERR
DB1E:A9 00                    LDA   #0               ; UPDATE DIRECTORY TO FREE ENTRY SPACE.
DB20:8D BA DB                 STA   DFIL+D.STOR
DB23:CD A9 DB                 CMP   H.FCNT           ; FILE ENTRY WRAP?
DB26:D0 03   DB2B             BNE   DST1             ; BRANCH IF NO CARRY ADJUSTMENT
DB28:CE AA DB                 DEC   H.FCNT+1         ; TAKE CARRY FROM HIGH BYTE OF FILE ENTRIES
DB2B:CE A9 DB          DST1       DEC   H.FCNT           ; MARK HEADER WITH ONE LESS FILE
DB2E:A6 1A                    LDX   BMTAB            ; UPDATE (LAST) BITMAP.
DB30:20 65 D7                 JSR   BMAPUP
DB33:B0 8C   DAC1             BCS   DESTERR
DB35:A0 14                    LDY   #VCBTFRE
DB37:AD CD DB                 LDA   DFIL+D.USAGE
DB3A:71 B6                    ADC   (VCBPTR),Y
DB3C:91 B6                    STA   (VCBPTR),Y       ; UPDATE CURRENT FREE BLOCK COUNT.
DB3E:C8                       INY
DB3F:AD CE DB                 LDA   DFIL+D.USAGE+1
DB42:71 B6                    ADC   (VCBPTR),Y
DB44:91 B6                    STA   (VCBPTR),Y
DB46:A9 00                    LDA   #0               ; FORCE RESCAN FROM FIRST BITMAP
DB48:A0 1C                    LDY   #VCBCMAP
DB4A:91 B6                    STA   (VCBPTR),Y
DB4C:4C F0 C3                 JMP   DREVISE          ; UPDATE DIRECTORY LAST...
DB4F:                  *
```

```
DB4F:              *
DB4F:C9 D0         DSTDIR     CMP   #DIRTYP*16       ; IS THIS A DIRECTORY FILE?
DB51:F0 05   DB58             BEQ   DSDIR1           ; YES, PROCEED.
DB53:A9 00                    LDA   #CPTERR          ; FILE IS NOT COMPATABLE.
DB55:20 00 00                 JSR   SYSERR           ; GIVE UP.
DB58:              *
DB58:20 7F CB      DSDIR1     JSR   FNDBMAP          ; MAKE SURE A BUFFER IS AVAILABLE FOR THE BITMAP.
DB5B:B0 41   DB9E             BCS   DSDIRERR
DB5D:AD CB DB                 LDA   DFIL+D.FRST      ; READ IN FIRST BLOCK OF DIRECTORY INTO GBUF.
DB60:85 C6                    STA   BLOKNML
DB62:AD CC DB                 LDA   DFIL+D.FRST+1
DB65:85 C7                    STA   BLOKNMH
DB67:20 58 CC                 JSR   RDGBUF
DB6A:B0 32   DB9E             BCS   DSDIRERR
DB6C:AD 25 12                 LDA   GBUF+HCENT+4     ; FIND OUT IF ANY FILES EXIST ON THIS DIRECTORY.
DB6F:D0 05   DB76             BNE   DSDIRACC         ; BRANCH IF ANY EXIST.
DB71:AD 26 12                 LDA   GBUF+HCENT+5
DB74:F0 05   DB7B             BEQ   DSDIR2
DB76:A9 00         DSDIRACC   LDA   #ACCSERR
DB78:20 00 00                 JSR   SYSERR
DB7B:              *
DB7B:AD 02 12      DSDIR2     LDA   GBUF+2           ; GET FORWARD LINK.
DB7E:CD 03 12                 CMP   GBUF+3           ; TEST FOR NO LINK.
DB81:D0 04   DB87             BNE   DSDIR3
DB83:C9 00                    CMP   #0
DB85:F0 8C   DB13             BEQ   DSTLAST          ; IF NO LINK, THEN FINISHED.
DB87:AE 03 12      DSDIR3     LDX   GBUF+3
DB8A:20 04 CA                 JSR   DEALLOC          ; FREE THIS BLOCK.
DB8D:B0 0F   DB9E             BCS   DSDIRERR
DB8F:AD 02 12                 LDA   GBUF+2
DB92:85 C6                    STA   BLOKNML
DB94:AD 03 12                 LDA   GBUF+3
DB97:85 C7                    STA   BLOKNMH          ; READ IN LINKED BLOCK.
DB99:20 58 CC                 JSR   RDGBUF
DB9C:90 DD   DB7B             BCC   DSDIR2           ; LOOP UNTIL ALL ARE FREED.
DB9E:60            DSDIRERR   RTS
DB9F:              *
DB9F:              *
```

```
DB9F:        DB9F  WORKSPC    EQU   *
DB9F:        0001  V.STATUS   DS    1                ; VOLUME STATUS, INCLUDES 'ACTIVE' IN BIT 7
DBA0:        0002  H.CREDT    DS    2                ; DIRECTORY CREATION DATE
DBA2:        0002             DS    2                ; DIRECTORY CREATION TIME
DBA4:        0001             DS    1                ; VERSION UNDER WHICH THIS DIRECTORY WAS CREATED
DBA5:        0001             DS    1                ; EARLIEST VERSION THAT IT'S COMPATABLE WITH
DBA6:        0001  H.ATTR     DS    1                ; ATTRIBUTES (PROTECT BIT, ETC.)
DBA7:        0001  H.ENTLN    DS    1                ; LENGTH OF EACH ENTRY IN THIS DIRECTORY.
DBA8:        0001  H.MAXENT   DS    1                ; MAXIMUM NUMBER OF ENTRIES PER BLOCK
DBA9:        0002  H.FCNT     DS    2                ; CURRENT NUMBER OF FILES IN THIS DIRECTORY
DBAB:        0002             DS    2                ; ADDRESS OF FIRST ALLOCATION BIT MAP
DBAD:        0002             DS    2                ; TOTAL NUMBER OF BLOCKS ON THIS UNIT
DBAF:        0005             DS    5                ; (FOR FUTURE EXPANSION)
DBB4:              *
DBB4:        0001  D.DEV      DS    1                ; DEVICE NUMBER OF THIS DIRECTORY ENTRY
DBB5:        0002  D.HEAD     DS    2                ; ADDRESS OF <SUB> DIRECTORY HEADER
DBB7:        0002  D.ENTBLK   DS    2                ; ADDRESS OF BLOCK WHICH CONTAINS THIS ENTRY
DBB9:        0001  D.ENTNUM   DS    1                ; ENTRY NUMBER WITHIN BLOCK.
DBBA:        DBBA  DFIL       EQU   *
DBBA:        0000  D.STOR     EQU   *-DFIL           ; STORAGE TYPE * 16 + FILE NAME LENGTH
DBBA:        0001             DS    1
DBBB:              ; *-DFIL ; FILE NAME
DBBB:        000F             DS    15
DBCA:        0010  D.FILID    EQU   *-DFIL           ; USER'S IDENTIFICATION BYTE
DBCA:        0001             DS    1
DBCB:        0011  D.FRST     EQU   *-DFIL           ; FIRST BLOCK OF FILE
DBCB:        0002             DS    2
DBCD:        0013  D.USAGE    EQU   *-DFIL           ; NUMBER OF BLOCKS CURRENTLY ALLOCATED TO THIS FILE
DBCD:        0002             DS    2
DBCF:        0015  D.EOF      EQU   *-DFIL           ; CURRENT END OF FILE MARKER
DBCF:        0003             DS    3
DBD2:        0018  D.CREDT    EQU   *-DFIL           ; DATE OF FILE'S CREATION
DBD2:        0002             DS    2
DBD4:              ; *-DFIL ; TIME OF FILE'S CREATION
DBD4:        0002             DS    2
DBD6:              ;  EQU *-DFIL ; SOS VERSION THAT CREATED THIS FILE
DBD6:        0001             DS    1
DBD7:        001D  D.COMP     EQU   *-DFIL           ; BACKWARD VERSION COMPATABILTY
DBD7:        0001             DS    1
DBD8:        001E  D.ATTR     EQU   *-DFIL           ; 'PROTECT', READ/WRITE 'ENABLE' ETC.
DBD8:        0001             DS    1
DBD9:        001F  D.AUXID    EQU   *-DFIL           ; USER AUXILLARY IDENTIFACATION
DBD9:        0002             DS    2
DBDB:        0021  D.MODDT    EQU   *-DFIL           ; FILE'S LAST MODIFICATION DATE
DBDB:        0002             DS    2
DBDD:        0023  D.MODTM    EQU   *-DFIL           ; FILE'S LAST MODIFICATION TIME
DBDD:        0002             DS    2
DBDF:        0025  D.DHDR     EQU   *-DFIL           ; HEADER BLOCK ADDRESS OF FILE'S DIRECTORY
DBDF:        0002             DS    2
DBE1:              *
DBE1:        0002  CMDADR     DS    2
DBE3:        000D  SCRTCH     DS    13               ; SCRATCH AREA FOR ALLOCATION ADDRESS CONVERSION
DBF0:        0003  OLDEOF     DS    3                ; TEMP USED IN W/R
DBF3:        0003  OLDMARK    DS    3                ; USED BY 'RDPOSN' AND 'WRITE'
DBF6:        00DB  SCRHIGH    EQU   <SCRTCH          ; AND DEVICE NUMBERS FROM BOB'S CODE.
DBF6:              *
```

```
DBF6:                                  CHN   SWAPOUT.IN
DBF6:             DBF6  SWAPOUT    EQU   *
DBF6:                   *
DBF6:                   * SWAP OUT A VOLUME LOGGED ON A DEVICE
DBF6:                   * INPUT ARGUMENT: DEVICE NUMBER "A"
DBF6:                   * (STORED AS "DEVNUM")
DBF6:                   * OUTPUT ARGUMENT: NONE
DBF6:                   * CONDITION CODE: CARRY SET USER DID NOT COMPLY WITH REQUEST
DBF6:                   *
DBF6:                   * SAVE VCBPTR, FCBPTR, DEVNUM ON STACK
DBF6:                   * 1) FIND UNSWAPPED VOLUME IN VCB
DBF6:                   * 2) IF DIRTY BIT MAP FOR THIS VOLUME THEN DO
DBF6:                   *   IF NOT ONLINE, REQUEST USER TO INSERT
DBF6:                   *   IF REQUEST DENIED, UNCONDITIONALLY  CLOSE ALL FILES ON THIS VOLUME AND RTS
DBF6:                   *   IF ONLINE, UPDATE AND RELEASE BIT MAP
DBF6:                   * DOEND
DBF6:                   * 3) SWAP IT (MARK VCBSWAP FIELD $80, MARK ALL FILES ON THIS VOLUME WITH SWAP MARK $8X WHERE X=VCB
ENTRY)
DBF6:                   * "VCB ENTRY" DEFINED AS: HIGH ORDER NIBBLE OF LOW ORDER BYTE OF ENTRIES VCB ADDRESS
DBF6:                   * RESTORE VCBPTR, FCBPTR
DBF6:                   * RTS
DBF6:                   *
DBF6:AA                            TAX                       ; SAVE DEVICE NUMBER
DBF7:20 9C DC                      JSR   SAVECBS
DBFA:86 35                         STX   DEVNUM              ; PERMANENTLY
DBFC:20 48 C8           SWAPOUTX   JSR   DEVVCB              ; FIND MATCHING UNSWAPPED ACTIVE VCB ENTRY (BY DEVNUM)
DBFF:B0 44        DC45             BCS   SORTS               ; NO FIND--RETURN WITHOUT ERROR
DC01:A0 11                         LDY   #VCBSTAT
DC03:B1 B6                         LDA   (VCBPTR),Y          ; GET STATUS OF FILES ON THIS VOLUME
DC05:10 43        DC4A             BPL   UNLOG               ; IF NO OPEN FILES, JUST THROW VOLUME AWAY
DC07:A5 35                         LDA   DEVNUM              ; DIRTY BM EXIST ON THIS VOLUME?
DC09:A2 00                         LDX   #0
DC0B:D5 1D                         CMP   BMADEV,X            ; IN BIT MAP "A"?
DC0D:F0 09        DC18             BEQ   FDIRBM              ; BRANCH IF YES
DC0F:A2 06                         LDX   #6                  ; BIT MAP HEADER TABLE SIZE
DC11:D5 1D                         CMP   BMADEV,X            ; IN BIT MAP "B"?
DC13:F0 03        DC18             BEQ   FDIRBM              ; BRANCH IF YES
DC15:4C 33 DC                      JMP   MARKSWAP            ; NO NEED TO WRITE BIT MAP
DC18:B5 1C              FDIRBM     LDA   BMASTAT,X           ; IS BIT MAP DIRTY?
DC1A:10 17        DC33             BPL   MARKSWAP            ; BRANCH IF NOT
DC1C:20 0A C9           GETVOL     JSR   VERFYVOL            ; IS THE CORRECT VOLUME ON LINE NOW?
DC1F:90 0D        DC2E             BCC   VONLINE             ; BRANCH IF YES
DC21:20 2F DD                      JSR   USRREQ              ; OTHERWISE, REQUEST USER INSERTION
DC24:90 F6        DC1C             BCC   GETVOL              ; AND VERIFY IT AGAIN
DC26:20 9B DD                      JSR   CLOSEU              ; USER SAID "NO": UNCONDITIONALLY CLOSE VOLUME
DC29:20 B6 DC                      JSR   RESTCBS
DC2C:38                            SEC
DC2D:60                            RTS                       ; ERROR RETURN TO CALLER
DC2E:A6 35              VONLINE    LDX   DEVNUM              ; UPDATE THE
DC30:20 E4 CB                      JSR   UPBMAP              ; DIRTY BIT MAP
DC33:A5 B6              MARKSWAP   LDA   VCBPTR              ; CALCULATE
DC35:4A                            LSR   A                    ; SWAP BYTE
DC36:4A                            LSR   A                    ; AND
DC37:4A                            LSR   A                    ; MARK ALL FILES
DC38:4A                            LSR   A                    ; BELONGING TO THIS VOLUME
DC39:38                            SEC                       ; AS SWAPPED OUT
DC3A:09 80                         ORA   #$80
```

```
DC3C:48                          PHA                  ; SAVE SWAP BYTE
DC3D:20 D0 DC                    JSR   FCBSCAN
DC40:68                          PLA                  ; MARK VCBSWAP
DC41:A0 1F                       LDY   #VCBSWAP       ; BYTE
DC43:91 B6                       STA   (VCBPTR),Y
DC45:20 B6 DC         SORTS      JSR   RESTCBS        ; RESTORE FCBPTR, VCBPTR, DEVNUM
DC48:18                          CLC
DC49:60                          RTS                  ; SUCCESSFUL SWAP OUT
DC4A:A9 00            UNLOG      LDA   #0
DC4C:9D 00 11                    STA   VCB,X          ; UNLOG VOLUME
DC4F:F0 F4   DC45                BEQ   SORTS          ; SWAP THE EASY WAY! (BRANCH ALWAYS)
DC51:                 *
DC51:                 *
DC51:                 *
DC51:        DC51     SWAPIN     EQU   *
DC51:                 *
DC51:                 * UNSWAP A VOLUME AND ALL ITS FILES
DC51:                 *
DC51:                 * INPUT ARGUMENT: VOLUME NAME (VCBPTR)
DC51:                 * OUTPUT ARGUMENT: NONE
DC51:                 * CONDITION CODE: CARRY SET : USER DID NOT COMPLY WITH REQUEST
DC51:                 *
DC51:                 * SAVE VCBPTR, FCBPTR ON STACK
DC51:                 * 1) FIND SWAPPED VOLUME IN VCB, IF NOT FOUND, THEN RTS.
DC51:                 * 2) IF ANOTHER UNSWAPPED VOLUME ON DEVICE, THEN SWAP IT
DC51:                 * 3) VERIFY UNSWAPPED VOLUME, IF NOT OK THEN REQUEST INSERTION
DC51:                 * 4) UNMARK VCB'S AND FCB'S
DC51:                 * RTS
DC51:20 9C DC                    JSR   SAVECBS        ; SAVE FCB, VCB POINTERS, DEVNUM
DC54:A0 00                       LDY   #VCBNML        ; MAKE SURE VOLUME
DC56:B1 B6                       LDA   (VCBPTR),Y     ; IS AT LEAST OPEN
DC58:F0 38   DC92                BEQ   USRTS          ; BRANCH IF NOT RIGHT BACK TO CALLER
DC5A:A0 1F                       LDY   #VCBSWAP       ; SEE IF
DC5C:B1 B6                       LDA   (VCBPTR),Y     ; CURRENTLY SWAPPED
DC5E:F0 32   DC92                BEQ   USRTS          ; IF NOT, RETURN IMMEDIATELY TO CALLER
DC60:A0 10                       LDY   #VCBDEV        ; SAVE DEVICE NUMBER
DC62:B1 B6                       LDA   (VCBPTR),Y
DC64:85 35                       STA   DEVNUM
DC66:48                          PHA                  ; SAVE DEVNUM AGAIN (SWAPOUTX TRASHES DEVNUM ON RETURN)
DC67:20 FC DB                    JSR   SWAPOUTX       ; AND MAKE SURE ANY CURRENT ACTIVE VOLUME IS SWAPPED OUT (NOTICE
ENTRY POINT)
DC6A:68                          PLA                  ; RECALL CURRENT DEVICE NUMBER
DC6B:85 35                       STA   DEVNUM         ; AND SAVE IT TO ITS PROPER PLACE
DC6D:20 0A C9         SI1        JSR   VERFYVOL       ; VERIFY THE CURRENT VOLUME MOUNTED
DC70:90 0D   DC7F                BCC   UNMARK         ; IF THE RIGHT ONE, GO MARK IT AS UNSWAPPED
DC72:20 2F DD                    JSR   USRREQ         ; ELSE REQUEST USER TO INSERT
DC75:90 F6   DC6D                BCC   SI1            ; USER SAID 'OK'
DC77:20 9B DD                    JSR   CLOSEU         ; OTHERWISE UNCONDITIONALLY CLOSE
DC7A:20 B6 DC                    JSR   RESTCBS
DC7D:38                          SEC
DC7E:60                          RTS
DC7F:A0 1F            UNMARK     LDY   #VCBSWAP       ; FETCH
DC81:B1 B6                       LDA   (VCBPTR),Y     ; VOLUME
DC83:48                          PHA                  ; SWAP BYTE
DC84:A9 00                       LDA   #0             ; BUT CLEAR
DC86:91 B6                       STA   (VCBPTR),Y     ; VOLUME SWAP
DC88:68                          PLA
```

```
DC89:18                         CLC                   ; "UNSWAPPED"
DC8A:20 D0 DC                   JSR   FCBSCAN
DC8D:A5 35                      LDA   DEVNUM          ; MAKE SURE BIT MAPS
DC8F:20 F8 CB                   JSR   CLEARBMS        ; ARE MARKED AS INVALID ON THIS DEVICE
DC92:20 B6 DC          USRTS    JSR   RESTCBS         ; RESTORE VCB, FCB PTRS
DC95:18                         CLC                   ; NO ERRORS
DC96:60                         RTS
DC97:                  *
DC97:        0005      SAVEPTRS DS    5               ; A RARE EMBEDDED TEMP SAVE AREA, USED ONLY BY ...
DC9C:                  *
DC9C:                  *
DC9C:        DC9C      SAVECBS  EQU   *               ; SAVE FCBPTR, VCBPTR IN A TEMP SAVE AREA
DC9C:A5 B6                      LDA   VCBPTR
DC9E:8D 97 DC                   STA   SAVEPTRS
DCA1:A5 B7                      LDA   VCBPTR+1
DCA3:8D 98 DC                   STA   SAVEPTRS+1
DCA6:A5 BA                      LDA   FCBPTR
DCA8:8D 99 DC                   STA   SAVEPTRS+2
DCAB:A5 BB                      LDA   FCBPTR+1
DCAD:8D 9A DC                   STA   SAVEPTRS+3
DCB0:A5 35                      LDA   DEVNUM
DCB2:8D 9B DC                   STA   SAVEPTRS+4
DCB5:60                         RTS
DCB6:                  *
DCB6:        DCB6      RESTCBS  EQU   *               ; RESTORE FCBPTR, VCBPTR
DCB6:                  * NOTICE THERE EXISTS A SEQUENCE OF CALLS (SWAPIN, WHICH MAY CALL SWAPOUT) THAT JSR'S TO SAVECBS
ONCE BUT JSR'S RESTCBS TWICE.
DCB6:AD 97 DC                   LDA   SAVEPTRS
DCB9:85 B6                      STA   VCBPTR
DCBB:AD 98 DC                   LDA   SAVEPTRS+1
DCBE:85 B7                      STA   VCBPTR+1
DCC0:AD 99 DC                   LDA   SAVEPTRS+2
DCC3:85 BA                      STA   FCBPTR
DCC5:AD 9A DC                   LDA   SAVEPTRS+3
DCC8:85 BB                      STA   FCBPTR+1
DCCA:AD 9B DC                   LDA   SAVEPTRS+4
DCCD:85 35                      STA   DEVNUM
DCCF:60                         RTS
DCD0:                  *
DCD0:                  *
DCD0:                  * MARK ALL FILES BELONGING TO A VOLUME
DCD0:                  * AS SWAPPED-IN OR SWAPPED-OUT.
DCD0:                  *
DCD0:                  * INPUT ARGS: DEVNUM -- DEVICE NUMBER OF MOUNTED VOLUME
DCD0:                  *             A REGISTER - SWAP BYTE
DCD0:                  *             CARRY -- CARRY FLAG SET MEANS SWAP OUT; ELSE SWAP IN
DCD0:                  *
DCD0:                  * OUTPUT ARGS: NONE
DCD0:                  * GLOBALS AFFECTED: FCB, FCBPTR
DCD0:                  * REGISTER STATUS: SCRAMBLED
DCD0:                  *
DCD0:        DCD0      FCBSCAN  EQU   *               ; MARK FILES BELONGING TO VOLUME AS SWAPPED OR UNSWAPPED
DCD0:                  *
DCD0:AA                         TAX                   ; SAVE SWAP BYTE
DCD1:AC 28 00                   LDY   FCBADDRH        ; POINT TO
DCD4:84 BB                      STY   FCBPTR+1        ; BEGINNING TO FCB
DCD6:A0 00                      LDY   #0
```

```
DCD8:84 BA                    STY   FCBPTR
DCDA:B0 18   DCF4             BCS   FCBOUT           ; SWAP OUT A VOLUMES FILES
DCDC:        DCDC  FCBIN      EQU   *                ; SWAPIN A VOLUMES FILES
DCDC:20 0B DD                 JSR   FCBFETCH         ; GET NEXT ACTIVE FCB CANDIDATE
DCDF:B0 29   DD0A             BCS   FCBRTS           ; NO MORE FILES TO PROCESS
DCE1:A0 1A                    LDY   #FCBSWAP
DCE3:8A                       TXA
DCE4:D1 BA                    CMP   (FCBPTR),Y       ; SWAP BYTES MATCH?
DCE6:D0 04   DCEC             BNE   FCBIN1           ; BRANCH IF NOT
DCE8:A9 00                    LDA   #0
DCEA:91 BA                    STA   (FCBPTR),Y       ; MARK FILE AS SWAPPED IN
DCEC:20 1B DD      FCBIN1     JSR   NEXTFCB          ; ADVANCE FCB POINTER
DCEF:B0 19   DD0A             BCS   FCBRTS           ; NO MORE TO LOOK AT
DCF1:4C DC DC                 JMP   FCBIN            ; AND LOOK AT NEXT FILE
DCF4:              *
DCF4:        DCF4  FCBOUT     EQU   *                ; SWAPPED OUT A VOLUMES FILES
DCF4:20 0B DD                 JSR   FCBFETCH         ; GET NEXT ACTIVE FILE IN FCB
DCF7:B0 11   DD0A             BCS   FCBRTS           ; NO MORE FILES -- RETURN TO USER
DCF9:A0 1A                    LDY   #FCBSWAP         ; COMPARE
DCFB:B1 BA                    LDA   (FCBPTR),Y
DCFD:D0 03   DD02             BNE   FCBOUT1          ; ALREADY SWAPPED OUT
DCFF:8A                       TXA
DD00:91 BA                    STA   (FCBPTR),Y       ; MARK AS SWAPPED
DD02:20 1B DD      FCBOUT1    JSR   NEXTFCB          ; ADVANCE FCB POINTER
DD05:B0 03   DD0A             BCS   FCBRTS
DD07:4C F4 DC                 JMP   FCBOUT           ; SWAP OUT NEXT FILE
DD0A:              *
DD0A:60            FCBRTS     RTS
DD0B:        DD0B  FCBFETCH   EQU   *                ; GET NEXT ACTIVE FILE FROM FCB
DD0B:              * X REGISTER MUST NOT BE DISTURBED
DD0B:              * USES FCBPTR
DD0B:A0 01                    LDY   #FCBDEVN         ; MAKE
DD0D:B1 BA                    LDA   (FCBPTR),Y       ; SURE DEVICE
DD0F:C5 35                    CMP   DEVNUM           ; MATCHES
DD11:D0 08   DD1B             BNE   NEXTFCB
DD13:A0 00                    LDY   #FCBREFN         ; MAKE SURE FILE IS ACTIVE
DD15:B1 BA                    LDA   (FCBPTR),Y
DD17:F0 02   DD1B             BEQ   NEXTFCB          ; BRANCH IF NOT
DD19:18                       CLC
DD1A:60                       RTS                    ; RETURN WITH CARRY CLEAR SHOWING AN ACTIVE FILE
DD1B:A5 BA         NEXTFCB    LDA   FCBPTR
DD1D:18                       CLC
DD1E:69 20                    ADC   #$20             ; FCB ENTRY SIZE
DD20:85 BA                    STA   FCBPTR
DD22:90 E7   DD0B             BCC   FCBFETCH         ; BRANCH IF NO PAGE CROSS
DD24:A5 BB                    LDA   FCBPTR+1
DD26:E6 BB                    INC   FCBPTR+1         ; SECOND PAGE
DD28:CD 28 00                 CMP   FCBADDRH
DD2B:F0 DE   DD0B             BEQ   FCBFETCH         ; LOOK AT PAGE TWO
DD2D:38            NEXTEND    SEC
DD2E:60                       RTS                    ; SHOW NO MORE FILES TO LOOK AT
DD2F:        DD2F  USRREQ     EQU   *                ; OPERATOR CONSOLE MESSAGE INTERFACE
DD2F:              * PRODUCES A MESSAGE REQUESTING
DD2F:              * THE SYSTEM OPERATOR TO MOUNT THE VOLUME
DD2F:              * SPECIFIED BY "VCBPTR" ON DEVICE SPECIFIED
DD2F:              * BY DEVNUM.  THIS MODULE INSISTS
```

```
DD2F:              * UPON THE CORRECT OPERATOR ACTION
DD2F:              * UPON THREE FAILURES TO COMPLY,
DD2F:              * THE MODULE WILL SIGNIFY FAILURE WITH
DD2F:              * CARRY SET.  IF THE CORRECT ACTION IS TAKEN,
DD2F:              * CARRY WILL BE RETURNED CLEAR
DD2F:              *
DD2F:              * INPUT ARGS: VOLUME NAME (VCBPTR)
DD2F:              *             DEVICE NUMBER (DEVNUM)
DD2F:              *
DD2F:              * OUTPUT ARGS: CC = OPERATOR COMPLIED WITH REQUESTED ACTION
DD2F:              *              CS = OPERATOR COULDN'T/DIDN'T COMPLY
DD2F:              *
DD2F:              * GLOBALS AFFECTED: NONE
DD2F:              *
DD2F:              * STATUS OF REGISTERS: UNCERTAIN
DD2F:              *
DD2F:        003D  VNML       EQU   ZPGTEMP          ; VOLUME NAME LENGTH
DD2F:A0 00                    LDY   #VCBNML          ; IF ILLEGAL VCB
DD31:B1 B6                    LDA   (VCBPTR),Y       ; GET OUT QUICK
DD33:F0 F8   DD2D             BEQ   NEXTEND          ; BRANCH TO SEC RTS
DD35:A2 0E                    LDX   #$E              ; LENGTH OF NAMED AREA-1
DD37:A9 00                    LDA   #$0              ; NULLS
DD39:9D 32 DE      UR1        STA   MDEV,X           ; BOTH CLEAR
DD3C:9D 13 DE                 STA   MVOL,X           ; IN ONE LOOP
DD3F:CA                       DEX
DD40:10 F7   DD39             BPL   UR1
DD42:              *
DD42:              * DO A D-INFO TO FETCH THE DEVICE NAME
DD42:              *
DD42:A9 05                    LDA   #5               ; DO ALL
DD44:85 C0                    STA   $C0              ; NECESSARY
DD46:A5 35                    LDA   DEVNUM           ; HOUSKEEPING
DD48:85 C1                    STA   $C1              ; TO SET UP
DD4A:A9 31                    LDA   #>MDEV-1         ; A DEVICE MANAGER CALL
DD4C:85 C2                    STA   $C2
DD4E:A9 DE                    LDA   #<MDEV-1
DD50:85 C3                    STA   $C3
DD52:A9 8F                    LDA   #$8F             ; EXTEND BYTE
DD54:8D C3 14                 STA   $14C3
DD57:A9 00                    LDA   #0
DD59:8D C2 14                 STA   $14C2
DD5C:85 C4                    STA   $C4
DD5E:85 C5                    STA   $C5
DD60:85 C6                    STA   $C6              ; ZERO SUPERFLUOUS PARMS
DD62:8D 03 DE                 STA   URDERR           ; RESET FAILURE COUNT
DD65:20 3E CF                 JSR   RPEATIO0         ; GET INFO FROM BOBS CODE
DD68:A9 20                    LDA   #$20             ; "SPACE" RESTORED
DD6A:8D 31 DE                 STA   MDEV-1           ; RESTORED
DD6D:A0 00                    LDY   #VCBNML
DD6F:B1 B6                    LDA   (VCBPTR),Y       ; LENGTH OF VOLUME NAME
DD71:85 3D                    STA   VNML             ; SAVED FOR WORK
DD73:A9 00                    LDA   #0
DD75:AA                       TAX
DD76:A0 01                    LDY   #VCBNAM          ; POINT TO BEGINNING OF VOLUME NAME
DD78:B1 B6         UR2        LDA   (VCBPTR),Y
DD7A:9D 13 DE                 STA   MVOL,X
```

```
DD7D:E8                       INX
DD7E:C8                       INY                    ; VOLUME NAME MOVED
DD7F:C6 3D                    DEC   VNML             ; TO MESSAGE BUFFER
DD81:D0 F5   DD78             BNE   UR2              ; CHARACTER BY CHARACTER
DD83:A2 04         URDU       LDX   #>UMB            ; PASS THE AREA'S ADDR
DD85:A0 DE                    LDY   #<UMB            ; IN X AND Y REGS,LOW, HIGH)
DD87:20 00 00                 JSR   OPMSGRPLY        ; HAVE MESSAGE SYSTEM PRINT IT
DD8A:20 0A C9                 JSR   VERFYVOL         ; DID THE USER COMPLY?
DD8D:B0 01   DD90             BCS   URDU1            ; BRANCH IF NOT
DD8F:60                       RTS                    ; EXIT--CARRY IS CLEAR
DD90:EE 03 DE      URDU1      INC   URDERR           ; COLLECT USER ERRORS
DD93:AD 03 DE                 LDA   URDERR
DD96:C9 03                    CMP   #3               ; ONLY THREE TRIES ALLOWED
DD98:90 E9   DD83             BCC   URDU             ; RETRY MESSAGE IF LESS THAN THREE TRIES
DD9A:60                       RTS                    ; OTHERWISE RETURN WITH CARRY SET
DD9B:              *
DD9B:              *
DD9B:              *
DD9B:              *
DD9B:              *
DD9B:              * CLOSE UNCONDITIONAL
DD9B:              *
DD9B:              * (USER HAS REPLIED 'N' TO A VOLUME MOUNT REQUEST
DD9B:              * CLOSE ALL FILES ON VOLUME/UNLOG VOLUME
DD9B:              *
DD9B:              * INPUT ARGUMENT: (VCBPTR)
DD9B:              * OUTPUT ARGUMENT: NONE
DD9B:              *
DD9B:        DD9B  CLOSEU     EQU   *
DD9B:        003D  VSWA       EQU   ZPGTEMP          ; THE 'SWAP BYTE' STORED HERE
DD9B:A0 10                    LDY   #VCBDEV          ; FETCH
DD9D:B1 B6                    LDA   (VCBPTR),Y       ; THE DEVICE NUMBER
DD9F:85 35                    STA   DEVNUM           ; OF THIS VOLUME & SAVE IT
DDA1:A0 1F                    LDY   #VCBSWAP         ; FETCH THE
DDA3:B1 B6                    LDA   (VCBPTR),Y       ; SWAP BYTE
DDA5:85 3D                    STA   VSWA             ; SAVE FOR REFERENCE, TOO
DDA7:A9 00                    LDA   #0
DDA9:A0 00                    LDY   #VCBNML          ; UNLOG THE VOLUME
DDAB:91 B6                    STA   (VCBPTR),Y       ; BY SETTING LEN OF VOL NAME TO ZERO
DDAD:A0 1F                    LDY   #VCBSWAP
DDAF:91 B6                    STA   (VCBPTR),Y       ; TURN OFF SWAP FLAG
DDB1:AC 28 00                 LDY   FCBADDRH         ; SET UP FCB SCAN FROM BEGINNING OF FCB
DDB4:84 BB                    STY   FCBPTR+1
DDB6:A0 00                    LDY   #0
DDB8:84 BA                    STY   FCBPTR
DDBA:A0 01         VFCBLOP    LDY   #FCBDEVN         ; FETCH
DDBC:B1 BA                    LDA   (FCBPTR),Y       ; THE DEVICE
DDBE:C5 35                    CMP   DEVNUM           ; NUMBER AND SEE IF A MATCH
DDC0:D0 1F   DDE1             BNE   VFCBNXT          ; BRANCH IF NO MATCH
DDC2:A0 00                    LDY   #FCBREFN         ; SEE EVEN IF FILE OPEN
DDC4:B1 BA                    LDA   (FCBPTR),Y
DDC6:F0 19   DDE1             BEQ   VFCBNXT          ; BRANCH IF NOT
DDC8:A0 1A                    LDY   #FCBSWAP         ; CHECK TO SEE IF ATTACHED
DDCA:B1 BA                    LDA   (FCBPTR),Y        ; TO SAME VOLUME
DDCC:C5 3D                    CMP   VSWA
DDCE:D0 11   DDE1             BNE   VFCBNXT          ; BRANCH IF NOT
```

```
DDD0:A0 0B                    LDY   #FCBBUFN          ; RELEASE
DDD2:B1 BA                    LDA   (FCBPTR),Y       ; ANY
DDD4:20 00 00                 JSR   RELBUF           ; BUFFERS ASSOCIATED
DDD7:A0 1A                    LDY   #FCBSWAP         ; AND CLEAR
DDD9:A9 00                    LDA   #0               ; THE SWAP BYTE
DDDB:91 BA                    STA   (FCBPTR),Y
DDDD:A0 00                    LDY   #FCBREFN         ; AND FINALLY
DDDF:91 BA                    STA   (FCBPTR),Y       ; SAY 'CLOSED'
DDE1:A5 BA         VFCBNXT    LDA   FCBPTR
DDE3:18                       CLC
DDE4:69 20                    ADC   #$20              ; FCB ENTRY SIZE
DDE6:85 BA                    STA   FCBPTR
DDE8:90 D0   DDBA             BCC   VFCBLOP
DDEA:A5 BB                    LDA   FCBPTR+1
DDEC:E6 BB                    INC   FCBPTR+1         ; LOOK AT SECOND PAGE
DDEE:CD 28 00                 CMP   FCBADDRH
DDF1:F0 C7   DDBA             BEQ   VFCBLOP          ; CHECK PAGE TWO OF FCB
DDF3:60                       RTS                    ; RETURN TO USER W/O ERROR
DDF4:              *
DDF4:        DDF4  FCBUSED    EQU   *                ; MARK AS FCB AS DIRTY SO
DDF4:              * THE DIRECTORY WILL BE FLUSHED ON 'FLUSH'
DDF4:84 3D                    STY   ZPGTEMP
DDF6:48                       PHA                    ; SAVE REGS
DDF7:A0 1C                    LDY   #FCBDIRTY
DDF9:B1 BA                    LDA   (FCBPTR),Y       ; FETCH CURRENT FCBDIRTY BYTE
DDFB:09 80                    ORA   #FCBMOD          ; MARK FCB AS DIRTY
DDFD:91 BA                    STA   (FCBPTR),Y       ; SAVE IT BACK
DDFF:68                       PLA
DE00:A4 3D                    LDY   ZPGTEMP          ; AND RESTORE REGS
DE02:60                       RTS
DE03:              *
DE03:        0001  URDERR     DS    1                ; ERROR COUNT FOR USRREQ
DE04:              *
DE04:              *
DE04:        DE04  UMB        EQU   *
DE04:49 6E 73 65              DFB   $49,$6E,$73,$65,$72,$74,$20
DE0B:76 6F 6C 75              DFB   $76,$6F,$6C,$75,$6D,$65
DE11:3A 20                    DFB   $3A,$20          ; "INSERT VOLUME: "
DE13:        000F  MVOL       DS    15
DE22:0D                       DFB   $0D              ; CR LINE TERMINATOR
DE23:20 20 20 20              DFB   $20,$20,$20,$20,$69,$6E,$20
DE2A:64 65 76 69              DFB   $64,$65,$76,$69,$63,$65
DE30:3A 20                    DFB   $3A,$20          ; "    IN DEVICE: "
DE32:        000F  MDEV       DS    15
DE41:0D                       DFB   $0D              ; CR LINE TERMINATOR
DE42:74 68 65 6E              DFB   $74,$68,$65,$6E,$20,$70,$72
DE49:65 73 73 20              DFB   $65,$73,$73,$20,$74,$68,$65,$20
DE51:41 4C 50 48              DFB   $41,$4C,$50,$48,$41,$20,$4C
DE58:4F 43 4B 20              DFB   $4F,$43,$4B,$20,$6B,$65,$79
DE5F:20 74 77 69              DFB   $20,$74,$77,$69,$63,$65
DE65:              * "THEN PRESS THE ALPHA LOCK KEY TWICE"
DE65:              * FOLLOWED WITH $FF MESSAGE TERMINATOR (HIGH BIT SIGNIFICANT)
DE65:FF                       DFB   $FF              ; MESSAGE TERMINATOR (HIGH BIT)
DE66:              *
DE66:        2266  ZZLEN      EQU   *-ZZORG
DE66:        DE66  ZZEND      EQU   *
```

```
DE66:        0000  392            IFNE  ZZLEN-LENBFM
 S                 393            FAIL  2,"SOSORG       FILE IS INCORRECT FORMBFM"
DE66:              394            FIN
```

```
X0029 ACCSERR        CADB ADCALC         CCDC ADDPOSN        CCCD ADJMARK
 CD00 ADJMRK0        CCFE ADJMRK         CD07 ADJMRK1        D190 ADJSTCNT
 BD73 ADPREFIX       CA9C ALC1BLK       X0036 ALCERR         CA6E ALCIDXS
 D557 ALCWBLK        CA9B ALDXEND        BD9E ALFA1          BDB2 ALFA2
 CA77 ALIDX1         D577 ALUSERR        CFD0 ASGNFCB       X0019 BACKMASK
X002D BADLSTCNT     X001B BADPATH       X001D BADREFNUM        18 BASVAL
N001C BFMFCB1       N001D BFMFCB2       NBC00 BFMGR          CAC3 BITFOUND
X0033 BITMAPADR      D957 BKBITFLG         20 BKBITVAL       3200 BLABFM
?2E00 BLABFMI        6B52 BLABUFMG       6955 BLACFM         5E99 BLADISK3
 64D9 BLADMGR        68F4 BLAFMGR       ?2CF8 BLAGLOB       ?2AF8 BLAINIT
 55C0 BLAIPL         2000 BLALODR       ?6E6E BLAMEMMG       5466 BLAOMSG
 5466 BLAPATCH       665E BLASCMGR       6404 BLASERR        5A8B BLAUMGR
X0017 BLKDLST          C7 BLOKNMH          C6 BLOKNML        C51A BLOKSAVE
   21 BMACMAP          1F BMADADR          1D BMADEV           B8 BMADR
N001E BMAMADR       N00B8 BMAPAGE        C987 BMAPRD         D765 BMAPUP
   1C BMASTAT          23 BMBDEV        N0024 BMBMADR       N00BA BMBPAGE
   22 BMBSTAT          1B BMBUFBNK         0D BMCNT          CB9D BMFOUND
   17 BMPTR            1A BMTAB            06 BMTABSZ        CAE1 BOUNCE
   C8 BRDPTR        X002A BTSERR         D088 BUFREQST         2F BULKCNT
 BD7F BUMPATH          A7 C.AUXID          A2 C.BASE           A4 C.BYTES
   A1 C.DNAMP          AD C.EOFHH          AC C.EOFHL          AB C.EOFLH
   AA C.EOFLL          A6 C.FILID          A3 C.FILIST         A5 C.FILSTLN
   A2 C.ISNEWL         A3 C.MARK           A3 C.MAXPTH         A2 C.MRKPTR
   A3 C.NEWEOF         A3 C.NEWL           A3 C.NWPATH         A5 C.OPLIST
   A7 C.OPLSTLN        A5 C.OUTBLK         A2 C.OUTBUF         A6 C.OUTCNT
   A2 C.OUTEOF         A3 C.OUTREF         A3 C.OUTVOL         A1 C.PATH
   A1 C.REFNUM         A9 C.STOR           A5 C.XLEN           A3 C.XLIST
 D617 C3             D618 CFERR          C517 CFLAG          CA00 CFREE1
 C99C CHGVCB         D124 CHKACTV        CBC4 CHKBMB         CD46 CHKDSKS1
 CD38 CHKDSKSW       C706 CHKROOT        C7A0 CHKVLOG        C713 CKROOT1
 CD09 CKSAMBLK       CBF8 CLEARBMS       D5DF CLOSALL        D5D5 CLOSE
 D619 CLOSE1         D61E CLOSE2         D644 CLOSEND        D646 CLOSERR
 DD9B CLOSEU         CC05 CLRBM1         CC04 CLRBM2         C6A0 CLRDSP
 BC78 CLRDSWT        CFD4 CLRFCB         BC35 CLRSIS         CE7B CLRSTATS
 D5E1 CLSALL1        DBE1 CMDADR         BC9F CMDTABLE         37 CMDTEMP
 CCBC CMPEOF         C9B1 CMPFREB        C676 CMPNAME        C8F2 CMPVCB
   0B CNTENT         C9F5 CNTFREE          A0 COMMAND        C9C3 COUNT
X0025 CPTERR         C1B0 CREALC         C109 CREAT1         C0F1 CREATE
 C130 CRENAM1        C12B CRENAM         C235 CRERR          C0FB CRERR1
 C2E2 CRERR2         C36B CRETIME        C387 CRNXTDIR       001E D.ATTR
 001F D.AUXID        001D D.COMP         0018 D.CREDT        DBB4 D.DEV
 0025 D.DHDR         DBB7 D.ENTBLK       DBB9 D.ENTNUM       0015 D.EOF
 0010 D.FILID        0011 D.FRST         DBB5 D.HEAD         0021 D.MODDT
 0023 D.MODTM        0000 D.STOR         0013 D.USAGE        C2FE DADD1
 DA54 DALBLK1        DA62 DALBLK2        DA6C DALBLK3        DA70 DALBLKERR
   01 DATALC           01 DATBLKH          00 DATBLKL        C2BC DATDONE
?  39 DATEHI           38 DATELO        X0011 DATETIME       C292 DATINIT
 C2AD DATIT1         CDFD DATLEVEL         40 DATMOD           BC DATPTR
 D429 DBLOKALC         C3 DBUFPH           C2 DBUFPL         C4E9 DCRENTH
 DA52 DEALBLK        CA62 DEALERR1       CA61 DEALERR        CA46 DEALL1
 CA4E DEALL2         CA58 DEALL3         CA04 DEALLOC          AD DEBUPTR
 D080 DEFBUFR        D004 DEFOPEN        C3CE DERROR        ?C460 DERROR1
 C3EF DERROR2        DAC1 DESTERR        DA71 DESTROY          C0 DEVICE
   35 DEVNUM         C848 DEVVCB         DBBA DFIL             C0 DHPCMD
 C3AC DIRCREND      X0032 DIRERR        X0024 DIRFULL        CEA8 DIRFWRD
 CE84 DIRMARK        C2F0 DIROVR         CEB5 DIRPOS1        CE8D DIRPOS
 CEC6 DIRPOS2          0D DIRTYP         CE9B DIRVRSE        C3B4 DIRWRT
 C3C3 DIRWRT1        BCC3 DISPTCH          2F DLIMIT         CF49 DMGRGO
X0012 DMGR           CC6A DOBITMAP       CC32 DOBMAP         CEF6 DOFILEIO
 CC7E DOFRST         CC6A DOIDX            B5 DRBUFPH          B4 DRBUFPL
 D31E DREAD          D354 DREDERR        D353 DREDONE        C3FE DREVISE1
 C3F0 DREVISE        BD0A DRIVENAM       CEC4 DRPOSERR       C342 DRSTUF1
 C33C DRSTUF         DB58 DSDIR1         DB7B DSDIR2         DB87 DSDIR3
 DB76 DSDIRACC       DB9E DSDIRERR       C9BF DSKFULL        DB2B DST1
   C4 DSTATBFH         C3 DSTATBFL         C2 DSTATREQ       DB4F DSTDIR
 DB13 DSTLAST        DADB DSTNXT         DB10 DSTNXT1        DAD9 DSTRE2
 DAE9 DSTRE3         DAF0 DSTRE4         DAAE DSTREE         DA81 DSTROY1
 DA91 DSTROY2        DA9D DSTROY3          80 DSTROYEN       DAC2 DSTSAP
 D5BB DSWGLOB          40 DSWITCH       X0021 DUPERR           3C DUPLFLAG
X0022 DUPVOL         C84A DVCB1          C85C DVCB2            C3 DVDNUM
 BC6A DVERIFY          C1 DVNAMP         C3DF ECALC0         C3E0 ECALC1
 C3EC ECALC2         C240 ENDCRE         C24F ENDCRE0        C261 ENDCRE1
 C259 ENDCRX         BDD6 ENDPATH        D2D9 ENDRCHK1       D2DB ENDRCHK2
 D2CC ENDRQCHK       D49A ENDWCHK1       D49C ENDWCHK2       D48D ENDWQCHK
 C636 ENTADR         C3D6 ENTCALC          0A ENTCNTH          09 ENTCNTL
 C785 ENTVCB         C793 ENTVCB2        D03F EOFCBMV       X0027 EOFERR
   20 EOFMOD         D83B EOFOUT         D78F EOFRETN        D182 EOFTEST
 D71C EOFUPDTE       D361 ERRACCS        CA95 ERRALC1        D072 ERRBTS
 CFBE ERRBUSY        BC93 ERRCMD         D020 ERRCMPAT      ?C620 ERRCOMP
 C4ED ERRDIR         D355 ERRDRD         D25F ERRFIX         D264 ERRFIX1
 C51C ERRFNF         C3CE ERRGBUF        CCCA ERRMEOF        BEF9 ERRNOREF
 BF01 ERRNOTBLK      D0C5 ERROPEN2       CFC0 ERROPN         D07F ERROPN1
 BC98 ERRORSYS       C5C8 ERRPATH1       CCFA ERRPOSN        BDFE ERRSYN1
 BD6F ERRSYN         C5CD ERTS           BC80 EXECUTE        D67B F3
 CC92 FADDR         N0028 FCBADDRH         29 FCBANKNM         09 FCBATTR
   0B FCBBUFN          10 FCBDATB          01 FCBDEVN          1C FCBDIRTY
   06 FCBENTN          15 FCBEOF         DD0B FCBFETCH         0C FCBFRST
X001C FCBFULL          0E FCBIDXB        DCDC FCBIN          DCEC FCBIN1
   1B FCBLEVL        CEE5 FCBLOKNM         12 FCBMARK          80 FCBMOD
   0A FCBNEWL        DD02 FCBOUT1        DCF4 FCBOUT         CFDB FCBOWNR
   BA FCBPTR           00 FCBREFN        DD0A FCBRTS         DCD0 FCBSCAN
   08 FCBSTAT          07 FCBSTYP          1A FCBSWAP        D08D FCBUFFER
 CFA9 FCBUPDAT         18 FCBUSE         DDF4 FCBUSED       N00BA FCBZPP
 DC18 FDIRBM         CD53 FERRTYP       X002B FILBUSY        CF25 FILEIO2
 CF0A FILEIO         CF0E FILEIO1        C636 FILFOUND       CFAF FILIOERR
 C20E FILLTREE       BE75 FINDFCB        C480 FINDFILE       C71E FINDVOL
 D07A FIXDBUF          12 FLINK          D655 FLSHAL1        D653 FLSHALL
 D700 FLSHEBLK       D776 FLSHEND1       D67C FLSHERR        D687 FLUSH1
 D691 FLUSH2A        D67F FLUSH2         D6CD FLUSH4         D707 FLUSH5
 D674 FLUSHEND       D778 FLUSHERR       D649 FLUSH          D6A7 FLUSH2B
 D6B3 FLUSH2C        D6C0 FLUSH3         CB7F FNDBMAP        C883 FNDDUP1
 C622 FNDERR1        C4EF FNDERR         BEA0 FNDFCBUF       BEEC FNDFV.1
 BEEE FNDFV1         BEC6 FNDFVOL        CB85 FNDMAP1        C729 FNDVOL1
 C5C0 FNF0           C514 FNF0X          C5CE FNF1          X0020 FNFERR
 C886 FOUNDDUP       C882 FOUNDEV        C759 FOUNDVOL       CBBA FRBMBUF
 C9CD FRCNT1         C9D5 FRCNT2         C9F4 FRCNT3         C9C5 FRCONT
 CBCE FREBUF1        CBD4 FREBUFA        CB20 FREEA          CB1E FREEBE
 C7AA FREEVCB        CB89 FRESHMAP       BD87 FRSTCHAR       BF05 FVOLFOUND
 D307 FXDATPTR       1200 GBUF           CB77 GETA.BUF       CB7B GETB.BUF
 CAA5 GETBITS1       CAB2 GETBITS2      X0015 GETBUFADR      BF0D GETDNUM
 D87E GETEOF         D8AF GETINFO        CC9B GETMARK        BE3D GETPREFX
 CF5B GETPRMS        C7C0 GETROOT        C91E GETROT0        DC1C GETVOL
 D781 GFCBADR        D90E GINFOEND       D90F GINFOERR       D778 GLBERR
 D780 GLBERR1        CC9D GMARK1         BC9C GOCMD          BC9B GOODOP
 D1B4 GORDDNE        BE73 GOTPRFX        CC10 GTBMAP         BEA4 GTBUFFRS
 D8EB GTINFO1        D8ED GTINFO2        D903 GTINFO3        D907 GTINFO4
 CB0A GTTINDX        DBA6 H.ATTR         DBA0 H.CREDT        DBA7 H.ENTLN
 DBA9 H.FCNT         DBA8 H.MAXENT         19 HALF             1E HATTR
   21 HCENT            1D HCMP             18 HCRDT            0E HEDTYP
   00 HNLEN         ?  11 HPASS            10 HPENAB           23 HRBLK
   26 HRELN            25 HRENT            1C HVER             03 IDXADRH
   02 IDXADRL          02 IDXALC           80 IDXMOD         C18A INCDATA
 BE07 INCPTH1        BE01 INCTPTH        D56E INCUSG1          06 INDXBLK
 D958 INFTABL          34 IOACCESS       C2F4 ISDIR1         C2EA ISDIR
 C66D ISNAME         BF88 KNOTSOS        2266 LENBFM        ?0400 LENBFMI
 031C LENBUFMG       01FD LENCFM         056B LENDISK3       0185 LENDMGR
   61 LENFMGR       ?01B2 LENINIT        04CB LENIPL         0AF8 LENLODR
?0751 LENMEMMG       015A LENOMSG          00 LENPATCH       0296 LENSCMGR
   D5 LENSERR        040E LENUMGR          07 LEVELS        X000F LEVEL
 C89A LOGVCB1        C88F LOGVCB         C770 LOKDEV1        C658 LOKNAM1
 C681 LOKNAM2        C7B2 LOKVOL1        C7CC LOKVOL2        C4CD LOOKFIL0
 C4D2 LOOKFIL1       C4F4 LOOKFIL2       C493 LOOKFILE       C64D LOOKNAM
 C764 LOOKVOL1       C762 LOOKVOL        BDED LSTNAME        C237 LSTSAP
 DC33 MARKSWAP         0F MAXTEMPS       DE32 MDEV           C3F6 MODTIME
 C488 MOVENT1        C485 MOVENTRY       C624 MOVHEAD        C62A MOVHED0
 C62C MOVHED1        CCAA MOVMRK         C8B4 MOVOLNM        C123 MOVPARM
 BE34 MOVPRFX        C422 MVDENT         DA36 MVHEDNAM       C360 MVHNAME
 DE13 MVOL           DA33 MVROTNAM       C4F1 NAMFOJMP       C5D1 NAMFOUND
 D7B9 NEOFPOS        D7C4 NEOFTST        D893 NEWLINE        DD2D NEXTEND
 DD1B NEXTFCB        BF78 NFOPEN           30 NLCHAR           10 NLINEN
 CE14 NODATA         C882 NODUPVOL       C492 NOFIND           0C NOFREE
 CBE3 NOGO           CDEB NOIDXDAT       CBB6 NOMORBIT       C622 NONAME
 D299 NONEWLIN       C845 NONSOS         BC42 NOPATH         BD79 NOPREFX
 BC4B NOPREREF       D24C NOSTUF        X002F NOTBLKDEV      C461 NOTDIR
 C8EF NOTLOG0        C929 NOTLOG1        C946 NOTLOG2        C2E3 NOTREE
 C71D NOTROOT        C927 NOTSAME       X002C NOTSOS         CBF6 NOUPDAT
 BFD2 NOVOLM         C91A NOVRFY         C91B NOVRFY1          31 NPATHDEV
 BE70 NULPREFX       D664 NXFLUSH        CB57 NXTBMAP        BDA4 NXTCHAR
 D600 NXTCLOS        C7D6 NXTDEV         C504 NXTDIR0        C75B NXTVCB
 D8A7 OFFNEWL        D8AD OFFRTS         DBF0 OLDEOF         DBF3 OLDMARK
 D076 ONEKTST        CFC2 OPEN1          D01B OPEN2          D02C OPEN4
 CFB0 OPEN           CFB9 OPEN0          D024 OPEN3          D0DB OPENDONE
X0010 OPMSGRPLY      D0D6 OPNDIR         D0BA OPNPOS1        D0AF OPNPOS
 BC00 ORGBFM         B800 ORGBFMI        F552 ORGBUFMG       F355 ORGCFM
 E899 ORGDISK3       EED9 ORGDMGR        FFBF ORGEND         F2F4 ORGFMGR
?18FC ORGGLOB        28F8 ORGINIT        DFC0 ORGIPL         1E00 ORGLODR
 F86E ORGMEMMG       DE66 ORGOMSG        DE66 ORGPATCH       F05E ORGSCMGR
 EE04 ORGSERR        E48B ORGUMGR        D882 OUTEOF         C16C OVFLOW
X0023 OVRERR         D6CF OWNRMOV          A0 PAR           N1000 PATHBUF
   14 PATHCNT          B1 PATHNMH          B0 PATHNML       X001E PATHNOTFND
N0015 PFIXPTR        C4A8 PHANTM1        C4B2 PHANTM2        CDCF POSERR
 CDD1 POSINDEX      X0028 POSNERR        CD6E POSNEW1        CD9B POSNEW2
 CE44 POSNEW3        CDA8 POSNIDX          BE POSPTR           80 PREPATH
 C692 PREPROOT       D27E PREPRW         D289 PREPRW1          40 PREREF
   20 PRETIME        BD21 PREVOLM1       BD1C PREVOLM        BD41 PREVOLM2
 CE79 PRITZ          D819 PUR1           D818 PUR2           D7F8 PURGE
 D839 PURHI          D83D PURLBLKS       D841 PURLOOP        D878 PURLRTS
 D87A PURPLACE       D7E6 PURTEST        D7F2 PURTEST1       D879 PURUSE
```

```
   00 RDCMD          D205 RDFAST         D20A RDFAST0        D21F RDFAST1
 D22E RDFAST2        CEEF RDFCBERR       CC90 RDFRST         CC58 RDGBUF
 D2F8 RDONE1         D2AB RDPART0        D2AC RDPART         D2B3 RDPART1
 D2B5 RDPART2        D2C3 RDPART3        D305 RDPART4        CD09 RDPOSN
 D2F3 RDPRTDNE       D2E5 RDRQDNE        D1A8 READ2          D1B7 READ3
   01 READEN         D26B READONE        D2A2 READPART       D154 READ
 D161 READ1          D249 REALRD         BEFD REEFER         BF00 REEFER1
X0016 RELBUF           40 RENAMEN        D968 RENAME         DA41 RENPATH
 C2BD REPEATIO      X0013 REQBUF        X0014 REQFXBUF         05 REQH
   04 REQL           BE16 RESETPFX       DCB6 RESTCBS        CB02 RET1BLK
 C928 RETROT2        CECA RFCBDAT        CEF0 RFCBFST        CED8 RFCBIDX
 DA30 RNAMDONE       D99F RNAME0         D9E5 RNAME1         D9F2 RNAME2
 D9FE RNAME3         D9B9 RNAMERR        D985 RNAMEVOL       D9EE RNBADPTH
?C3D6 RNDTAB         CE4A RNEWPOS        D996 RNMEVOL        C697 ROOT0
 C69C ROOT1          C6D6 ROOT2          C6E9 ROOT3          C705 ROOTERR
 C6EB ROOTINFO       C4C5 ROOTSTUF       CF3E RPEATIO0       CF3A RPEATIO1
 CF59 RPEATIO2       CF69 RPTBLOK          09 RPTCMD           C5 RQCNTH
   C4 RQCNTL         CF58 RRITZ          BFDC RTV1           BFD5 RTVOLNAM
   2E RWREQH           2D RWREQL         D9E7 SAMOWNR        D505 SAPDOWN
 D513 SAPDWN1        C237 SAPFILE        C26D SAPINDX        CDF0 SAPLEVEL
 C1A4 SAPLING          0E SAPTR            02 SAPTYP         DC9C SAVECBS
 DC97 SAVEPTRS       BDC6 SAVPATH        CF40 SAVPRMS        00DB SCRHIGH
 DBE3 SCRTCH         CA96 SECNDHAF       C170 SEED1          C166 SEED
   01 SEEDTYP        BE4C SENDPRFX      X0018 SERR           D7D5 SETEOF1
 D7D9 SETEOF2        D790 SETEOF         D7D0 SETEOF0        D78C SETERR
 D925 SETINF1        D92B SETINF1X       D93D SETINF2        D945 SETINF3
 D910 SETINFO        CCB2 SETMARK        BCD5 SETPATH        BE08 SETPREFX
 BE1B SETPRFX1       BE2A SETPRFX3       BE2D SETPRFX4       D2F0 SETRDNE
 D7A9 SETSAVE        D306 SETVFLG        D4AB SETWRDNE       DC6D SI1
 D954 SINFEND1       D93A SINFEND        D984 SINFOERR       14B9 SISBMADR
 14C3 SISBPH         14BD SISDATP        14C4 SISDSTAT       14BB SISFCBP
 14A3 SISOUTBF       14A2 SISPATH        14BF SISPOSP        14B0 SISTEMPS
N1400 SISTER         14B3 SISTPATH       14B1 SISUSRBF       CCB8 SMARK1
 BE5F SNDLMIT        BE63 SNDPRFX1       C802 SNSWIT1        C80F SNSWIT2
 C840 SNSWIT5        C841 SNSWIT6        C7E0 SNSWIT         C82E SNSWIT3
 C834 SNSWIT4        CB56 SOMERR1        DC45 SORTS          C3D0 SOSTMPH
 C3CF SOSTMPL        C3D1 SOSVER         BD49 SPATH2         BD66 SPATH3
 BD71 SPTHERR        BF24 SRCHDEV        CAA1 SRCHFRE        C6E8 SRITZ
 14C9 SSBRDPH        14A4 SSNWPATH       14B3 SSTIDXH          02 STATCMD
   00 STATSUB          08 STPMOD         CCEA SUBMARK        CCEE SUBPOSN
 D011 SVATTR1        D009 SVATTRB        BF05 SVCBADR        D9C1 SVENEWID
 C438 SVENTDIR       BE96 SVFCBLO        CC5A SVGCMD         CE54 SVMARK
 CE58 SVMRK1         D513 SWAPDOWN       D556 SWAPERR        DC51 SWAPIN
 DBFC SWAPOUTX       DBF6 SWAPOUT        BCE3 SYNPATH       X0034 SYSDEATH
X001A SYSERR         C5B0 TELFREE        C511 TELFREEX       C9C2 TFBERR
?  3B TIMEHI        ?  3A TIMELO           B2 TINDX            10 TLINK
X0037 TOOLONG          04 TOPALC         D4CB TOPDOWN        D4DD TOPDWN1
   36 TOTDEVS          08 TOTENT           B2 TPATH          D512 TPDWNERR
   2C TPOSHI           2B TPOSLH           2A TPOSLL         CF67 TRASH
 D1DC TREAD0         D1D4 TREAD          C1AD TREE           D7F5 TRELEAS1
 D87B TRELEASE       CD5B TREPOS           0F TREPTR         C247 TRETIME
   03 TRETYP         D456 TREWRT1        CB95 TRYMAP2        C2FF TSDIRSZ
 C867 TSDUPV1        C87B TSDUPV2        C95F TSFR01         C94C TSFRBLK
 D142 TSNXFCB        BDBE TSTDLIM        C863 TSTDUPVOL      C463 TSTERR
 C0FD TSTFNF         D9BB TSTFNF1        CD88 TSTINY         D2DF TSTNEWL
 D0F6 TSTOPEN        D108 TSTOPN1        D110 TSTOPN2        C191 TSTSAP
 D41E TSTSAPWR       C19C TSTSEED        D9A9 TSTSMROT       C465 TSTSOS
 BDF6 TSTVALD        D578 TSTWPROT       BC54 TSWVRFY        CC76 TTLINK
 C518 TTSAVE         D5BC TWRCODE        D41C TWRITEGO       D3D7 TWRITE
 D587 TWRPROT1       D3E6 TWRTALC       X0026 TYPERR         CD2B TYPMARK
 D064 UBUFSPEC       DE04 UMB              C1 UNITNUM        DC4A UNLOG
 DC7F UNMARK         CBEE UPBM1          CBE4 UPBMAP         C44C UPHEAD
 C44E UPHED1         DD39 UR1            DD78 UR2            DE03 URDERR
 DD90 URDU1          DD83 URDU           CB33 USEBUF           10 USEMOD
   B0 USRBUF         DD2F USRREQ         DC92 USRTS          DB9F V.STATUS
   23 VBMAP            1C VCBCMAP        C8FE VCBCMP1          10 VCBDEV
   1A VCBDMAP          3E VCBENTRY      X0035 VCBERR         C8EE VCBLOGD
   01 VCBNAM           00 VCBNML           1E VCBOPNC          B6 VCBPTR
   16 VCBROOT          20 VCBSIZE          11 VCBSTAT          1F VCBSWAP
N1100 VCB              12 VCBTBLK          14 VCBTFRE        C90A VERFYVOL
 DDBA VFCBLOP        DDE1 VFCBNXT        C026 VFOUND         C03F VFOUND1
 C051 VFOUND2        C087 VFREEX         C069 VFREE          C0B2 VINFO1
 C0C2 VINFO2         C09E VINFO          C01A VLOGGED        C09B VLOGIN
 BF39 VLOOK00        BF4B VLOOK0         BF90 VLOOK1         BF93 VLOOK2
 BFA3 VLOOK3         BF8C VLOOK7         C057 VNEW           C061 VNEW1
X001F VNFERR         C08C VNFIL            3D VNML           BF7F VNOSWIT1
 BF79 VNOSWIT        BF46 VNOTEQ         C0CE VNOTSOS        C0E0 VNS2
 C0E9 VNXTVCB        BFF9 VOL2           BFF2 VOL7           C00A VOL8
 BF7D VOLERR1        C0CC VOLERR         C73D VOLNAM         BF30 VOLOOK
 C0CD VOLRET         BFDE VOLUME         DC2E VONLINE          3D VSWA
 C039 VSWAPIN          25 VTBLK          D399 WADJEOF        D38B WEOFTST
 CF84 WFCBDAT        CF73 WFCBFST        CF94 WFCBIDX        CA66 WHICHBIT
 D12C WHOWNS        NDB9F WORKSPC        D364 WPERROR        D5B9 WPROT1
 D5B0 WPROTRET       D5BD WRAPADJ        D5D4 WRAPDNE        D463 WRITDONE
 D3AB WRITE2         D3BA WRITE3           02 WRITEN         D358 WRITE
 D365 WRITE1         D403 WRITERR01      D40F WRITERR02      D400 WRITERROR
   B4 WRKPATH        D477 WRPART2        D4C6 WRPART4        D46F WRPART
 D484 WRPART3        D4AE WRPRTDNE       D39C WRTADJEOF      CC4F WRTBMAP
   01 WRTCMD         CC8C WRTDFRST       CC54 WRTGBUF        CC78 WRTINDX
 D466 WRTPART        D4A0 WRTRQDNE      X002E XDISKSW       X0031 XIOERROR
X0030 XNOWRITE       C10D ZERCALL        C2C4 ZERGBUF        C89E ZERVCB
 C2C7 ZGBUF          C2D4 ZINDX1         C2DB ZINDX2         CE35 ZIPDAT0
 CE32 ZIPDATA        CE21 ZIPIDX         CE3C ZPDAT1           3D ZPGTEMP
 CE28 ZPIDX1           B0 ZTEMPS         C2D1 ZTMPIDX       ?DE66 ZZEND
 2266 ZZLEN          BC00 ZZORG
 0000 D.STOR           00 FCBREFN          00 HNLEN            00 VCBNML
   00 DATBLKL          00 RDCMD            00 LENPATCH         00 STATSUB
   01 WRTCMD           01 VCBNAM           01 SEEDTYP          01 DATALC
   01 DATBLKH          01 READEN           01 FCBDEVN          02 WRITEN
   02 STATCMD          02 IDXADRL          02 IDXALC           02 SAPTYP
   03 TRETYP           03 IDXADRH          04 REQL             04 TOPALC
   05 REQH             06 INDXBLK          06 BMTABSZ          06 FCBENTN
   07 FCBSTYP          07 LEVELS           08 STPMOD           08 FCBSTAT
   08 TOTENT           09 ENTCNTL          09 RPTCMD           09 FCBATTR
   0A ENTCNTH          0A FCBNEWL          0B FCBBUFN          0B CNTENT
   0C FCBFRST          0C NOFREE           0D BMCNT            0D DIRTYP
   0E HEDTYP           0E SAPTR            0E FCBIDXB          0F TREPTR
   0F MAXTEMPS      X000F LEVEL          0010 D.FILID          10 USEMOD
   10 TLINK            10 FCBDATB          10 NLINEN           10 HPENAB
X0010 OPMSGRPLY        10 VCBDEV        ?  11 HPASS          0011 D.FRST
   11 VCBSTAT       X0011 DATETIME         12 FCBMARK          12 FLINK
   12 VCBTBLK       X0012 DMGR          X0013 REQBUF         0013 D.USAGE
   14 PATHCNT       X0014 REQFXBUF         14 VCBTFRE          15 FCBEOF
 0015 D.EOF         N0015 PFIXPTR       X0015 GETBUFADR        16 VCBROOT
X0016 RELBUF           17 BMPTR         X0017 BLKDLST          18 BASVAL
X0018 SERR             18 HCRDT          0018 D.CREDT          18 FCBUSE
   19 HALF          X0019 BACKMASK         1A FCBSWAP          1A VCBDMAP
   1A BMTAB         X001A SYSERR           1B BMBUFBNK      X001B BADPATH
   1B FCBLEVL          1C FCBDIRTY         1C VCBCMAP          1C HVER
   1C BMASTAT       X001C FCBFULL       N001C BFMFCB1       X001D BADREFNUM
N001D BFMFCB2          1D HCMP           001D D.COMP           1D BMADEV
 001E D.ATTR           1E HATTR         X001E PATHNOTFND    N001E BMAMADR
   1E VCBOPNC          1F BMADADR          1F VCBSWAP       X001F VNFERR
 001F D.AUXID          20 EOFMOD        X0020 FNFERR           20 PRETIME
   20 BKBITVAL         20 VCBSIZE        0021 D.MODDT          21 HCENT
X0021 DUPERR           21 BMACMAP       X0022 DUPVOL           22 BMBSTAT
   23 HRBLK            23 VBMAP         X0023 OVRERR         0023 D.MODTM
   23 BMBDEV        X0024 DIRFULL       N0024 BMBMADR       X0025 CPTERR
 0025 D.DHDR           25 VTBLK            25 HRENT         X0026 TYPERR
   26 HRELN         X0027 EOFERR        X0028 POSNERR       N0028 FCBADDRH
   29 FCBANKNM      X0029 ACCSERR          2A TPOSLL        X002A BTSERR
   2B TPOSLH        X002B FILBUSY          2C TPOSHI        X002C NOTSOS
   2D RWREQL        X002D BADLSTCNT        2E RWREQH        X002E XDISKSW
   2F BULKCNT          2F DLIMIT        X002F NOTBLKDEV        30 NLCHAR
X0030 XNOWRITE      X0031 XIOERROR         31 NPATHDEV      X0032 DIRERR
X0033 BITMAPADR     X0034 SYSDEATH         34 IOACCESS         35 DEVNUM
X0035 VCBERR        X0036 ALCERR           36 TOTDEVS       X0037 TOOLONG
   37 CMDTEMP          38 DATELO        ?  39 DATEHI        ?  3A TIMELO
?  3B TIMEHI           3C DUPLFLAG         3D ZPGTEMP          3D VSWA
   3D VNML             3E VCBENTRY         40 DSWITCH          40 RENAMEN
   40 PREREF           40 DATMOD           61 LENFMGR          80 IDXMOD
   80 DSTROYEN         80 PREPATH          80 FCBMOD           A0 COMMAND
   A0 PAR              A1 C.PATH           A1 C.DNAMP          A1 C.REFNUM
   A2 C.OUTBUF         A2 C.MRKPTR         A2 C.OUTEOF         A2 C.BASE
   A2 C.ISNEWL         A3 C.OUTVOL         A3 C.MARK           A3 C.OUTREF
   A3 C.FILIST         A3 C.NEWEOF         A3 C.MAXPTH         A3 C.NEWL
   A3 C.XLIST          A3 C.NWPATH         A4 C.BYTES          A5 C.XLEN
   A5 C.FILSTLN        A5 C.OPLIST         A5 C.OUTBLK         A6 C.OUTCNT
   A6 C.FILID          A7 C.OPLSTLN        A7 C.AUXID          A9 C.STOR
   AA C.EOFLL          AB C.EOFLH          AC C.EOFHL          AD DEBUPTR
   AD C.EOFHH          B0 ZTEMPS           B0 USRBUF           B0 PATHNML
   B1 PATHNMH          B2 TINDX            B2 TPATH            B4 WRKPATH
   B4 DRBUFPL          B5 DRBUFPH          B6 VCBPTR           B8 BMADR
N00B8 BMAPAGE       N00BA FCBZPP           BA FCBPTR        N00BA BMBPAGE
   BC DATPTR           BE POSPTR           C0 DEVICE           C0 DHPCMD
   C1 UNITNUM          C1 DVNAMP           C2 DBUFPL           C2 DSTATREQ
   C3 DSTATBFL         C3 DBUFPH           C3 DVDNUM           C4 RQCNTL
   C4 DSTATBFH         C5 RQCNTH           C6 BLOKNML          C7 BLOKNMH
   C8 BRDPTR           D5 LENSERR        00DB SCRHIGH        015A LENOMSG
 0185 LENDMGR       ?01B2 LENINIT        01FD LENCFM         0296 LENSCMGR
 031C LENBUFMG      ?0400 LENBFMI        040E LENUMGR        04CB LENIPL
 056B LENDISK3      ?0751 LENMEMMG       0AF8 LENLODR       N1000 PATHBUF
N1100 VCB            1200 GBUF          N1400 SISTER         14A2 SISPATH
 14A3 SISOUTBF       14A4 SSNWPATH       14B0 SISTEMPS       14B1 SISUSRBF
 14B3 SSTIDXH        14B3 SISTPATH       14B9 SISBMADR       14BB SISFCBP
 14BD SISDATP        14BF SISPOSP        14C3 SISBPH         14C4 SISDSTAT
 14C9 SSBRDPH       ?18FC ORGGLOB        1E00 ORGLODR        2000 BLALODR
 2266 LENBFM         2266 ZZLEN          28F8 ORGINIT       ?2AF8 BLAINIT
?2CF8 BLAGLOB       ?2E00 BLABFMI        3200 BLABFM         5466 BLAOMSG
 5466 BLAPATCH       55C0 BLAIPL         5A8B BLAUMGR        5E99 BLADISK3
 6404 BLASERR        64D9 BLADMGR        665E BLASCMGR       68F4 BLAFMGR
 6955 BLACFM         6B52 BLABUFMG      ?6E6E BLAMEMMG       B800 ORGBFMI
 BC00 ZZORG          BC00 ORGBFM        NBC00 BFMGR          BC35 CLRSIS
 BC42 NOPATH         BC4B NOPREREF       BC54 TSWVRFY        BC6A DVERIFY
 BC78 CLRDSWT        BC80 EXECUTE        BC93 ERRCMD         BC98 ERRORSYS
 BC9B GOODOP         BC9C GOCMD          BC9F CMDTABLE       BCC3 DISPTCH
 BCD5 SETPATH        BCE3 SYNPATH        BD0A DRIVENAM       BD1C PREVOLM
 BD21 PREVOLM1       BD41 PREVOLM2       BD49 SPATH2         BD66 SPATH3
 BD6F ERRSYN         BD71 SPTHERR        BD73 ADPREFIX       BD79 NOPREFX
 BD7F BUMPATH        BD87 FRSTCHAR       BD9E ALFA1          BDA4 NXTCHAR
 BDB2 ALFA2          BDBE TSTDLIM        BDC6 SAVPATH        BDD6 ENDPATH
 BDED LSTNAME        BDF6 TSTVALD        BDFE ERRSYN1        BE01 INCTPTH
 BE07 INCPTH1        BE08 SETPREFX       BE16 RESETPFX       BE1B SETPRFX1
 BE2A SETPRFX3       BE2D SETPRFX4       BE34 MOVPRFX        BE3D GETPREFX
 BE4C SENDPRFX       BE5F SNDLMIT        BE63 SNDPRFX1       BE70 NULPREFX
 BE73 GOTPRFX        BE75 FINDFCB        BE96 SVFCBLO        BEA0 FNDFCBUF
 BEA4 GTBUFFRS       BEC6 FNDFVOL        BEEC FNDFV.1        BEEE FNDFV1
 BEF9 ERRNOREF       BEFD REEFER         BF00 REEFER1        BF01 ERRNOTBLK
 BF05 FVOLFOUND      BF05 SVCBADR        BF0D GETDNUM        BF24 SRCHDEV
 BF30 VOLOOK         BF39 VLOOK00        BF46 VNOTEQ         BF4B VLOOK0
 BF78 NFOPEN         BF79 VNOSWIT        BF7D VOLERR1        BF7F VNOSWIT1
 BF88 KNOTSOS        BF8C VLOOK7         BF90 VLOOK1         BF93 VLOOK2
 BFA3 VLOOK3         BFD2 NOVOLM         BFD5 RTVOLNAM       BFDC RTV1
 BFDE VOLUME         BFF2 VOL7           BFF9 VOL2           C00A VOL8
 C01A VLOGGED        C026 VFOUND         C039 VSWAPIN        C03F VFOUND1
 C051 VFOUND2        C057 VNEW           C061 VNEW1          C069 VFREE
 C087 VFREEX         C08C VNFIL          C09B VLOGIN         C09E VINFO
 C0B2 VINFO1         C0C2 VINFO2         C0CC VOLERR         C0CD VOLRET
 C0CE VNOTSOS        C0E0 VNS2           C0E9 VNXTVCB        C0F1 CREATE
 C0FB CRERR1         C0FD TSTFNF         C109 CREAT1         C10D ZERCALL
 C123 MOVPARM        C12B CRENAM         C130 CRENAM1        C166 SEED
 C16C OVFLOW         C170 SEED1          C18A INCDATA        C191 TSTSAP
 C19C TSTSEED        C1A4 SAPLING        C1AD TREE           C1B0 CREALC
 C20E FILLTREE       C235 CRERR          C237 SAPFILE        C237 LSTSAP
 C240 ENDCRE         C247 TRETIME        C24F ENDCRE0        C259 ENDCRX
 C261 ENDCRE1        C26D SAPINDX        C292 DATINIT        C2AD DATIT1
 C2BC DATDONE        C2BD REPEATIO       C2C4 ZERGBUF        C2C7 ZGBUF
 C2D1 ZTMPIDX        C2D4 ZINDX1         C2DB ZINDX2         C2E2 CRERR2
 C2E3 NOTREE         C2EA ISDIR          C2F0 DIROVR         C2F4 ISDIR1
 C2FE DADD1          C2FF TSDIRSZ        C33C DRSTUF         C342 DRSTUF1
 C360 MVHNAME        C36B CRETIME        C387 CRNXTDIR       C3AC DIRCREND
 C3B4 DIRWRT         C3C3 DIRWRT1        C3CE DERROR         C3CE ERRGBUF
 C3CF SOSTMPL        C3D0 SOSTMPH        C3D1 SOSVER         C3D6 ENTCALC
?C3D6 RNDTAB         C3DF ECALC0         C3E0 ECALC1         C3EC ECALC2
 C3EF DERROR2        C3F0 DREVISE        C3F6 MODTIME        C3FE DREVISE1
 C422 MVDENT         C438 SVENTDIR       C44C UPHEAD         C44E UPHED1
?C460 DERROR1        C461 NOTDIR         C463 TSTERR         C465 TSTSOS
 C480 FINDFILE       C485 MOVENTRY       C488 MOVENT1        C492 NOFIND
 C493 LOOKFILE       C4A8 PHANTM1        C4B2 PHANTM2        C4C5 ROOTSTUF
 C4CD LOOKFIL0       C4D2 LOOKFIL1       C4E9 DCRENTH        C4ED ERRDIR
 C4EF FNDERR         C4F1 NAMFOJMP       C4F4 LOOKFIL2       C504 NXTDIR0
 C511 TELFREEX       C514 FNF0X          C517 CFLAG          C518 TTSAVE
 C51A BLOKSAVE       C51C ERRFNF         C5B0 TELFREE        C5C0 FNF0
 C5C8 ERRPATH1       C5CD ERTS           C5CE FNF1           C5D1 NAMFOUND
?C620 ERRCOMP        C622 FNDERR1        C622 NONAME         C624 MOVHEAD
 C62A MOVHED0        C62C MOVHED1        C636 ENTADR         C636 FILFOUND
 C64D LOOKNAM        C658 LOKNAM1        C66D ISNAME         C676 CMPNAME
 C681 LOKNAM2        C692 PREPROOT       C697 ROOT0          C69C ROOT1
 C6A0 CLRDSP         C6D6 ROOT2          C6E8 SRITZ          C6E9 ROOT3
 C6EB ROOTINFO       C705 ROOTERR        C706 CHKROOT        C713 CKROOT1
 C71D NOTROOT        C71E FINDVOL        C729 FNDVOL1        C73D VOLNAM
 C759 FOUNDVOL       C75B NXTVCB         C762 LOOKVOL        C764 LOOKVOL1
 C770 LOKDEV1        C785 ENTVCB         C793 ENTVCB2        C7A0 CHKVLOG
 C7AA FREEVCB        C7B2 LOKVOL1        C7C0 GETROOT        C7CC LOKVOL2
 C7D6 NXTDEV         C7E0 SNSWIT         C802 SNSWIT1        C80F SNSWIT2
 C82E SNSWIT3        C834 SNSWIT4        C840 SNSWIT5        C841 SNSWIT6
 C845 NONSOS         C848 DEVVCB         C84A DVCB1          C85C DVCB2
 C863 TSTDUPVOL      C867 TSDUPV1        C87B TSDUPV2        C882 NODUPVOL
 C882 FOUNDEV        C883 FNDDUP1        C886 FOUNDDUP       C88F LOGVCB
 C89A LOGVCB1        C89E ZERVCB         C8B4 MOVOLNM        C8EE VCBLOGD
 C8EF NOTLOG0        C8F2 CMPVCB         C8FE VCBCMP1        C90A VERFYVOL
 C91A NOVRFY         C91B NOVRFY1        C91E GETROT0        C927 NOTSAME
 C928 RETROT2        C929 NOTLOG1        C946 NOTLOG2        C94C TSFRBLK
 C95F TSFR01         C987 BMAPRD         C99C CHGVCB         C9B1 CMPFREB
 C9BF DSKFULL        C9C2 TFBERR         C9C3 COUNT          C9C5 FRCONT
 C9CD FRCNT1         C9D5 FRCNT2         C9F4 FRCNT3         C9F5 CNTFREE
 CA00 CFREE1         CA04 DEALLOC        CA46 DEALL1         CA4E DEALL2
 CA58 DEALL3         CA61 DEALERR        CA62 DEALERR1       CA66 WHICHBIT
 CA6E ALCIDXS        CA77 ALIDX1         CA95 ERRALC1        CA96 SECNDHAF
 CA9B ALDXEND        CA9C ALC1BLK        CAA1 SRCHFRE        CAA5 GETBITS1
 CAB2 GETBITS2       CAC3 BITFOUND       CADB ADCALC         CAE1 BOUNCE
 CB02 RET1BLK        CB0A GTTINDX        CB1E FREEBE         CB20 FREEA
 CB33 USEBUF         CB56 SOMERR1        CB57 NXTBMAP        CB77 GETA.BUF
 CB7B GETB.BUF       CB7F FNDBMAP        CB85 FNDMAP1        CB89 FRESHMAP
 CB95 TRYMAP2        CB9D BMFOUND        CBB6 NOMORBIT       CBBA FRBMBUF
 CBC4 CHKBMB         CBCE FREBUF1        CBD4 FREBUFA        CBE3 NOGO
 CBE4 UPBMAP         CBEE UPBM1          CBF6 NOUPDAT        CBF8 CLEARBMS
 CC04 CLRBM2         CC05 CLRBM1         CC10 GTBMAP         CC32 DOBMAP
 CC4F WRTBMAP        CC54 WRTGBUF        CC58 RDGBUF         CC5A SVGCMD
 CC6A DOBITMAP       CC6A DOIDX          CC76 TTLINK         CC78 WRTINDX
 CC7E DOFRST         CC8C WRTDFRST       CC90 RDFRST         CC92 FADDR
 CC9B GETMARK        CC9D GMARK1         CCAA MOVMRK         CCB2 SETMARK
 CCB8 SMARK1         CCBC CMPEOF         CCCA ERRMEOF        CCCD ADJMARK
 CCDC ADDPOSN        CCEA SUBMARK        CCEE SUBPOSN        CCFA ERRPOSN
 CCFE ADJMRK         CD00 ADJMRK0        CD07 ADJMRK1        CD09 CKSAMBLK
 CD09 RDPOSN         CD2B TYPMARK        CD38 CHKDSKSW       CD46 CHKDSKS1
 CD53 FERRTYP        CD5B TREPOS         CD6E POSNEW1        CD88 TSTINY
 CD9B POSNEW2        CDA8 POSNIDX        CDCF POSERR         CDD1 POSINDEX
 CDEB NOIDXDAT       CDF0 SAPLEVEL       CDFD DATLEVEL       CE14 NODATA
 CE21 ZIPIDX         CE28 ZPIDX1         CE32 ZIPDATA        CE35 ZIPDAT0
 CE3C ZPDAT1         CE44 POSNEW3        CE4A RNEWPOS        CE54 SVMARK
 CE58 SVMRK1         CE79 PRITZ          CE7B CLRSTATS       CE84 DIRMARK
 CE8D DIRPOS         CE9B DIRVRSE        CEA8 DIRFWRD        CEB5 DIRPOS1
 CEC4 DRPOSERR       CEC6 DIRPOS2        CECA RFCBDAT        CED8 RFCBIDX
 CEE5 FCBLOKNM       CEEF RDFCBERR       CEF0 RFCBFST        CEF6 DOFILEIO
 CF0A FILEIO         CF0E FILEIO1        CF25 FILEIO2        CF3A RPEATIO1
 CF3E RPEATIO0       CF40 SAVPRMS        CF49 DMGRGO         CF58 RRITZ
 CF59 RPEATIO2       CF5B GETPRMS        CF67 TRASH          CF69 RPTBLOK
 CF73 WFCBFST        CF84 WFCBDAT        CF94 WFCBIDX        CFA9 FCBUPDAT
 CFAF FILIOERR       CFB0 OPEN           CFB9 OPEN0          CFBE ERRBUSY
 CFC0 ERROPN         CFC2 OPEN1          CFD0 ASGNFCB        CFD4 CLRFCB
 CFDB FCBOWNR        D004 DEFOPEN        D009 SVATTRB        D011 SVATTR1
 D01B OPEN2          D020 ERRCMPAT       D024 OPEN3          D02C OPEN4
 D03F EOFCBMV        D064 UBUFSPEC       D072 ERRBTS         D076 ONEKTST
 D07A FIXDBUF        D07F ERROPN1        D080 DEFBUFR        D088 BUFREQST
 D08D FCBUFFER       D0AF OPNPOS         D0BA OPNPOS1        D0C5 ERROPEN2
 D0D6 OPNDIR         D0DB OPENDONE       D0F6 TSTOPEN        D108 TSTOPN1
 D110 TSTOPN2        D124 CHKACTV        D12C WHOWNS         D142 TSNXFCB
 D154 READ           D161 READ1          D182 EOFTEST        D190 ADJSTCNT
 D1A8 READ2          D1B4 GORDDNE        D1B7 READ3          D1D4 TREAD
 D1DC TREAD0         D205 RDFAST         D20A RDFAST0        D21F RDFAST1
 D22E RDFAST2        D249 REALRD         D24C NOSTUF         D25F ERRFIX
 D264 ERRFIX1        D26B READONE        D27E PREPRW         D289 PREPRW1
 D299 NONEWLIN       D2A2 READPART       D2AB RDPART0        D2AC RDPART
 D2B3 RDPART1        D2B5 RDPART2        D2C3 RDPART3        D2CC ENDRQCHK
 D2D9 ENDRCHK1       D2DB ENDRCHK2       D2DF TSTNEWL        D2E5 RDRQDNE
 D2F0 SETRDNE        D2F3 RDPRTDNE       D2F8 RDONE1         D305 RDPART4
 D306 SETVFLG        D307 FXDATPTR       D31E DREAD          D353 DREDONE
 D354 DREDERR        D355 ERRDRD         D358 WRITE          D361 ERRACCS
 D364 WPERROR        D365 WRITE1         D38B WEOFTST        D399 WADJEOF
 D39C WRTADJEOF      D3AB WRITE2         D3BA WRITE3         D3D7 TWRITE
 D3E6 TWRTALC        D400 WRITERROR      D403 WRITERR01      D40F WRITERR02
 D41C TWRITEGO       D41E TSTSAPWR       D429 DBLOKALC       D456 TREWRT1
 D463 WRITDONE       D466 WRTPART        D46F WRPART         D477 WRPART2
 D484 WRPART3        D48D ENDWQCHK       D49A ENDWCHK1       D49C ENDWCHK2
 D4A0 WRTRQDNE       D4AB SETWRDNE       D4AE WRPRTDNE       D4C6 WRPART4
 D4CB TOPDOWN        D4DD TOPDWN1        D505 SAPDOWN        D512 TPDWNERR
 D513 SWAPDOWN       D513 SAPDWN1        D556 SWAPERR        D557 ALCWBLK
 D56E INCUSG1        D577 ALUSERR        D578 TSTWPROT       D587 TWRPROT1
 D5B0 WPROTRET       D5B9 WPROT1         D5BB DSWGLOB        D5BC TWRCODE
 D5BD WRAPADJ        D5D4 WRAPDNE        D5D5 CLOSE          D5DF CLOSALL
 D5E1 CLSALL1        D600 NXTCLOS        D617 C3             D618 CFERR
 D619 CLOSE1         D61E CLOSE2         D644 CLOSEND        D646 CLOSERR
 D649 FLUSH          D653 FLSHALL        D655 FLSHAL1        D664 NXFLUSH
 D674 FLUSHEND       D67B F3             D67C FLSHERR        D67F FLUSH2
 D687 FLUSH1         D691 FLUSH2A        D6A7 FLUSH2B        D6B3 FLUSH2C
 D6C0 FLUSH3         D6CD FLUSH4         D6CF OWNRMOV        D700 FLSHEBLK
 D707 FLUSH5         D71C EOFUPDTE       D765 BMAPUP         D776 FLSHEND1
 D778 GLBERR         D778 FLUSHERR       D780 GLBERR1        D781 GFCBADR
 D78C SETERR         D78F EOFRETN        D790 SETEOF         D7A9 SETSAVE
 D7B9 NEOFPOS        D7C4 NEOFTST        D7D0 SETEOF0        D7D5 SETEOF1
 D7D9 SETEOF2        D7E6 PURTEST        D7F2 PURTEST1       D7F5 TRELEAS1
 D7F8 PURGE          D818 PUR2           D819 PUR1           D839 PURHI
 D83B EOFOUT         D83D PURLBLKS       D841 PURLOOP        D878 PURLRTS
 D879 PURUSE         D87A PURPLACE       D87B TRELEASE       D87E GETEOF
 D882 OUTEOF         D893 NEWLINE        D8A7 OFFNEWL        D8AD OFFRTS
 D8AF GETINFO        D8EB GTINFO1        D8ED GTINFO2        D903 GTINFO3
 D907 GTINFO4        D90E GINFOEND       D90F GINFOERR       D910 SETINFO
 D925 SETINF1        D92B SETINF1X       D93A SINFEND        D93D SETINF2
 D945 SETINF3        D954 SINFEND1       D957 BKBITFLG       D958 INFTABL
 D968 RENAME         D984 SINFOERR       D985 RNAMEVOL       D996 RNMEVOL
 D99F RNAME0         D9A9 TSTSMROT       D9B9 RNAMERR        D9BB TSTFNF1
 D9C1 SVENEWID       D9E5 RNAME1         D9E7 SAMOWNR        D9EE RNBADPTH
 D9F2 RNAME2         D9FE RNAME3         DA30 RNAMDONE       DA33 MVROTNAM
 DA36 MVHEDNAM       DA41 RENPATH        DA52 DEALBLK        DA54 DALBLK1
 DA62 DALBLK2        DA6C DALBLK3        DA70 DALBLKERR      DA71 DESTROY
 DA81 DSTROY1        DA91 DSTROY2        DA9D DSTROY3        DAAE DSTREE
 DAC1 DESTERR        DAC2 DSTSAP         DAD9 DSTRE2         DADB DSTNXT
 DAE9 DSTRE3         DAF0 DSTRE4         DB10 DSTNXT1        DB13 DSTLAST
 DB2B DST1           DB4F DSTDIR         DB58 DSDIR1         DB76 DSDIRACC
 DB7B DSDIR2         DB87 DSDIR3         DB9E DSDIRERR      NDB9F WORKSPC
 DB9F V.STATUS       DBA0 H.CREDT        DBA6 H.ATTR         DBA7 H.ENTLN
 DBA8 H.MAXENT       DBA9 H.FCNT         DBB4 D.DEV          DBB5 D.HEAD
 DBB7 D.ENTBLK       DBB9 D.ENTNUM       DBBA DFIL           DBE1 CMDADR
 DBE3 SCRTCH         DBF0 OLDEOF         DBF3 OLDMARK        DBF6 SWAPOUT
 DBFC SWAPOUTX       DC18 FDIRBM         DC1C GETVOL         DC2E VONLINE
 DC33 MARKSWAP       DC45 SORTS          DC4A UNLOG          DC51 SWAPIN
 DC6D SI1            DC7F UNMARK         DC92 USRTS          DC97 SAVEPTRS
 DC9C SAVECBS        DCB6 RESTCBS        DCD0 FCBSCAN        DCDC FCBIN
 DCEC FCBIN1         DCF4 FCBOUT         DD02 FCBOUT1        DD0A FCBRTS
 DD0B FCBFETCH       DD1B NEXTFCB        DD2D NEXTEND        DD2F USRREQ
 DD39 UR1            DD78 UR2            DD83 URDU           DD90 URDU1
 DD9B CLOSEU         DDBA VFCBLOP        DDE1 VFCBNXT        DDF4 FCBUSED
 DE03 URDERR         DE04 UMB            DE13 MVOL           DE32 MDEV
 DE66 ORGPATCH       DE66 ORGOMSG       ?DE66 ZZEND          DFC0 ORGIPL
 E48B ORGUMGR        E899 ORGDISK3       EE04 ORGSERR        EED9 ORGDMGR
 F05E ORGSCMGR       F2F4 ORGFMGR        F355 ORGCFM         F552 ORGBUFMG
 F86E ORGMEMMG       FFBF ORGEND

** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 30-APR-85 22:46
** TOTAL LINES ASSEMBLED  5522
** FREE SPACE PAGE COUNT    9
```