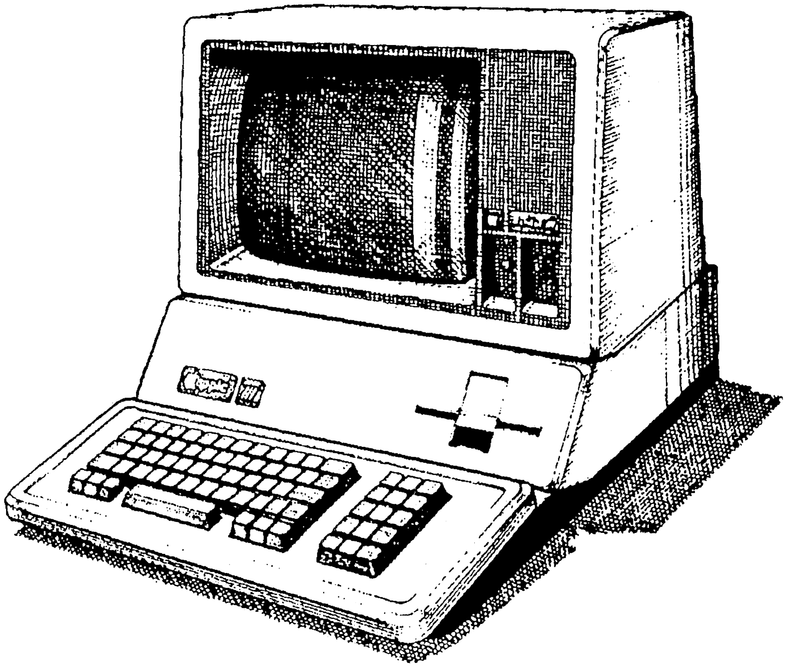
Apple /// Computer Technical Information

Apple ///
Sophisticated Operating System (SOS)
Version 1.3
Source Code Listing



How This Document Was Created

David T. Craig - 29 August 2001

This document was created from a set of text files containing the source code for SOS 1.3. These files were obtained around 1989 or 1990. These text files were originally on Apple II computer DOS 3.3 5.25" floppy diskettes. These files were transferred from the Apple II to either an Apple Lisa or Macintosh computer (I forget which one I used but suspect the Lisa). These files were then formatted in the Lisa Workshop (or Macintosh MPW Shell?) to have headers and footers.

Several years ago I turned these files into a PDF document since PDF was a more universal document format than many other formats. This file was included on my Apple /// Info CD from 1999 (or 1998?).

In August 2001, I created a new PDF document by taking the original text files that were on Macintosh disks. I used various computer tools which I had created over the years to do this.

DTCCatTextDocsRecursive
This Macintosh MPW Shell tool created a single text document from all the SOS source files. As this tool's name implies, it created its output document by recursively traversing the folder on my Mac with the SOS folders and subfolders. Resulting document stored in text document SOS.SOURCE.ALL.FILES.TEXT.

DTCEnTabNOT
This Macintosh MPW Shell converted all the tabs in the document to spaces. The original text files were based on tab stops every 8 positions which my Mac word processor (Microsoft Word 5.1) could not handle correctly. Here's the MPW command line and output for this tool:

```
DTCEnTabNOT 8 SOS.SOURCE.ALL.FILES.TEXT > SOS.SOURCE.ALL.FILES2.TEXT

Apple Macintosh EnTab NOT 1.0.0
David T. Craig / 71533.606@compuserve.com
December 4, 1997

Tab width = 8

Processing file "SOS.SOURCE.ALL.FILES.TEXT" ...

That's all folks!
```

Microsoft Word 5.1a
This Macintosh tool was used to put the text document (SOS.SOURCE.ALL.FILES2.TEXT) into a good format for printing (the document was saved as SOS.SOURCE.ALL.FILES.MSW). I added a nice header and footer to the document and also a nice introductory page complete with a scanned image of the Apple /// to make the document stand out as a /// document. I also used landscrape page orientation since parts of the listing have very long lines (e.g. the SOS loader files have some rather wide diagrams).

PDFWriter
This Macintosh printer driver was used to create the actual PDF document. I just told my Macintosh to use this printer driver and then printed from Microsoft Word. The end result was a PDF document which I named "Apple 3 SOS 1.3 Source Listing.pdf".

==

FILE: "SOS.SOURCE.ALL.FILES.TEXT"

==

==

DOCUMENT :SOS.SOURCE.INFO:SOS.AAA.1.README.TEXT

==

===

READ ME FILE FOR SOS SOURCE CODE DISK

Publicus / David T Craig -- March 1993

===

This Macintosh 800K HFS disk contains the complete source code listing for the Apple /// computer's operating system, SOS. This source listing is for version 1.3 of SOS, the last released SOS. Note that Apple had (to my knowledge) 3 SOS releases: 1.0, 1.1, 1.3 (version 1.2 appeares to have not been released to the public). Version 1.3's release date is February 1982.

SOS may be read as "Sophisticated Operating System" or "Sara's Operating System" since the Apple /// computer was code-named "SARA" by Apple Computer.

The Apple /// was Apple's premier business computer system for the time period 1980 to 1983.

This source listing is written in 6502 assembly language. The assembler used by Apple was an Apple][computer assembler which ran on a networked collection of Apple][computers. I have been told by knowledgable /// owners that the SOS source code was never ported to an Apple /// even though the /// had a nice assembler (as part of the ///'s Pascal development system).

For a detailed discussion of SOS see Apple Computer's well-written "SOS Reference Manual" series (two volumes).

From a historical perspective this source code is of no real use today since it is for a discontinued computer system. From a technical perspective this source is interesting since it provides a "real world" example of an operating system for a microcomputer. From a legal perspective this source is rather sensitive since parts of it may be used by Apple in its ProDOS operating system for the Apple][series (includes the //e and //GS).

Due to the legal ramifications of the SOS source code the author of this READ ME file shall remain anonymous.

This author would very much like to learn a little about how Apple developed

SOS. If any former /// development team members ever read this file, I hope that one of them will write a short "SOS History" and place it in a publically accessable area (e.g. CompuServe Information System).

Enjoy ...

==

```
==============================================================================
DOCUMENT :SOS.SOURCE.INFO:SOS.AAA.2.CATALOG.TEXT
==============================================================================

][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][
           APPLE /// SOS 1.3 A][ SOURCE CODE DISK CATALOG LISTINGS
][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][

/SOS1.3.ONE           Size   Modified  Time  File type  Eof Phys
  SYSGLOB.SRC           17  31-Dec-89 18:17  Asciifile  304   18
  OPRMSG.SRC            11  31-Dec-89 18:18  Asciifile  260   12
  IPL.SRC1              25  31-Dec-89 18:19  Asciifile  237   26
  IPL.SRC2              19  31-Dec-89 18:20  Asciifile  208   20
  SOSLDR.SRC             9  31-Dec-89 19:56  Asciifile  455   10
  BFM.INIT2.SRC          9  31-Dec-89 18:21  Asciifile  371   10
  INIT.SRC              16  31-Dec-89 18:22  Asciifile  194   17
  SOSLDR.A.SRC          12  31-Dec-89 18:34  Asciifile  424   13
  SOSLDR.B.SRC           9  13-Jan-90 22:17  Asciifile  113   10
  SOSLDR.C.SRC          13  31-Dec-89 18:23  Asciifile  440   14
  SOSLDR.D.SRC          29  31-Dec-89 18:24  Asciifile   28   30
  SOSLDR.E.SRC          16  31-Dec-89 18:25  Asciifile  334   17
  SOSLDR.F.SRC          21  31-Dec-89 18:27  Asciifile  419   22
13 files listed, 52 blocks available

/SOS1.3.TWO           Size   Modified  Time  File type  Eof Phys
  DISK3.SRC              9  31-Dec-89 18:53  Asciifile  496   10
  DISK3.DATA.SRC         3  31-Dec-89 18:54  Asciifile  403    4
  DISK3.SUBS.SRC        13  31-Dec-89 18:55  Asciifile  136   14
  DISK3.USEL.SRC         8  31-Dec-89 18:56  Asciifile  108    9
  DISK3.SIO.SRC         11  31-Dec-89 18:57  Asciifile  249   12
  DISK3.WRT.SRC          7  31-Dec-89 18:57  Asciifile   43    8
  DISK3.MAIN.SRC        10  31-Dec-89 18:58  Asciifile  175   11
  SYSERR.SRC             7  31-Dec-89 19:00  Asciifile  355    8
  SCMGR.SRC             22  31-Dec-89 19:01  Asciifile  500   23
  FMGR.SRC               4  31-Dec-89 19:01  Asciifile   95    5
  CFMGR.SRC             18  31-Dec-89 19:02  Asciifile   44   19
  BUFMGR.SRC            29  31-Dec-89 19:03  Asciifile   41   30
  MEMMGR.A.SRC          18  31-Dec-89 19:04  Asciifile  357   19
  MEMMGR.B.SRC          18  31-Dec-89 19:05  Asciifile   65   19
  MEMMGR.C.SRC          14  31-Dec-89 19:09  Asciifile  376   15
  DEVMGR.SRC            11  31-Dec-89 19:10  Asciifile  281   12
16 files listed, 55 blocks available

/SOS1.3.THREE         Size   Modified  Time  File type  Eof Phys
  UMGR.SRC              27  31-Dec-89 19:22  Asciifile  350   28
  ALLOC                 23  31-Dec-89 19:24  Asciifile  345   24
  EQUATES               19  31-Dec-89 20:55  Asciifile  186   20
  FNDFIL                42  31-Dec-89 19:27  Asciifile  263   43
  PRINT                  1  31-Dec-89 19:29  Asciifile  435    1
  PATH                  33  31-Dec-89 19:30  Asciifile  497   34
  VOLUME                 9  31-Dec-89 19:31  Asciifile  369   10
  CREATE                30  12-Jan-89 22:30  Asciifile  441   31
8 files listed, 82 blocks available

/SOS1.3.FOUR          Size   Modified  Time  File type  Eof Phys
  SWAPOUT.IN            21  31-Dec-89 19:47  Asciifile  303   22
  CLOSE.EOF             23  31-Dec-89 19:48  Asciifile   87   24
  READ.WRITE            38  31-Dec-89 19:49  Asciifile   86   39
  DESTROY               28  31-Dec-89 19:50  Asciifile  242   29
  POSN.OPEN             44  31-Dec-89 19:52  Asciifile  243   45
5 files listed, 114 blocks available

/SOS1.3.FIVE          Size   Modified  Time  File type  Eof Phys
  LCHK                   1  31-Dec-89 19:59  Asciifile  147    1
  LC                     1  31-Dec-89 20:00  Asciifile   74    1
  COMPILE.BFM            1  31-Dec-89 20:00  Asciifile   68    1
  COMPILE.SOS            2  31-Dec-89 20:01  Asciifile   97    3
  SOS.BLOAD              1  31-Dec-89 20:02  Asciifile  450    1
  SOS.LINK               1  31-Dec-89 20:03  Asciifile  170    1
  SOS.RENAME             2  31-Dec-89 20:03  Asciifile   11    3
  FEB01.1982             2  31-Dec-89 20:04  Asciifile   64    3
  PUBLICRELEASE          1  31-Dec-89 20:05  Asciifile   71    1
  COMP.SOS.NOLIST        2  31-Dec-89 20:05  Asciifile   79    3
  TCOMP.SOS              1  31-Dec-89 20:06  Asciifile  388    1
  SOSORG                 5  31-Dec-89 20:07  Asciifile  428    6
  C.S                    1  31-Dec-89 20:08  Asciifile  116    1
  C.BI2                  1  31-Dec-89 20:09  Asciifile   76    1
  C3                     1  31-Dec-89 20:09  Asciifile  155    1
  COMP.OPR.IPL           1  31-Dec-89 20:10  Asciifile  124    1
16 files listed, 244 blocks available

                     <<< END OF CATALOG LISTING >>>
```

```
==============================================================================
DOCUMENT :SOS.SOURCE.INFO:SOS.AAA.3.OPCODEFREQS.TEXT
==============================================================================

APPLE /// SOS 1.3 OPCODE INFORMATION

########################################## OPCODE LISTS

    Sorted by Name:

      Opcode count :    86
      Min frequency:     1
      Max frequency:  1864

        #    Opcode Name   Freq   Histogram
       ---   ------------  -----  ----------------------------------------
      [  1]  .PAGE             2  **
      [  2]  ADC             125  ****
      [  3]  AND             147  *****
      [  4]  ASC              32  **
      [  5]  ASL              46  **
      [  6]  BCC             240  *******
      [  7]  BCS             274  *******
      [  8]  BEQ             291  ********
      [  9]  BIT              58  ***
      [ 10]  BMI              59  ***
      [ 11]  BNE             397  **********
      [ 12]  BPL             121  ****
      [ 13]  BRK              12  **
      [ 14]  BVC              12  **
      [ 15]  BVS               7  **
      [ 16]  CHN              28  **
      [ 17]  CHR               1  *
      [ 18]  CLC             181  *****
      [ 19]  CLD               3  **
      [ 20]  CLI               5  **
      [ 21]  CLV               4  **
      [ 22]  CMP             286  *******
      [ 23]  CPX              40  **
      [ 24]  CPY              56  ***
      [ 25]  DEC              73  ***
      [ 26]  DEND              4  **
      [ 27]  DEX              75  ***
      [ 28]  DEY             126  ****
      [ 29]  DFB             188  *****
      [ 30]  DO               10  **
      [ 31]  DS              210  ******
      [ 32]  DSECT             4  **
      [ 33]  DW               83  ***
```

```
000187        [ 34]  ELSE           5  **
000188        [ 35]  ENTRY        190  *****
000189        [ 36]  EOR           21  **
000190        [ 37]  EQU         1134  *************************
000191        [ 38]  ERROR          1  *
000192        [ 39]  EXTRN        242  *******
000193        [ 40]  FAIL          17  **
000194        [ 41]  FIN           27  **
000195        [ 42]  IBUFSIZ        1  *
000196        [ 43]  IFNE          16  **
000197        [ 44]  INC          131  ****
000198        [ 45]  INCLUDE       21  **
000199        [ 46]  INX           33  **
000200        [ 47]  INY          143  ****
000201        [ 48]  JMP          199  ******
000202        [ 49]  JSR          546  *************
000203        [ 50]  LDA         1864  ****************************************
000204        [ 51]  LDX          266  *******
000205        [ 52]  LDY          450  ***********
000206        [ 53]  LSR           93  ***
000207        [ 54]  LST           13  **
000208        [ 55]  MSB           22  **
000209        [ 56]  NOP            2  **
000210        [ 57]  ORA          100  ****
000211        [ 58]  ORG           24  **
000212        [ 59]  PAGE         246  *******
000213        [ 60]  PHA           90  ***
000214        [ 61]  PHP           23  **
000215        [ 62]  PLA           91  ***
000216        [ 63]  PLP           20  **
000217        [ 64]  REL           16  **
000218        [ 65]  REP          417  **********
000219        [ 66]  ROL           14  **
000220        [ 67]  ROR           23  **
000221        [ 68]  RTI            3  **
000222        [ 69]  RTS          324  ********
000223        [ 70]  SBC          111  ****
000224        [ 71]  SBTL          32  **
000225        [ 72]  SBUFSIZ        1  *
000226        [ 73]  SEC          136  ****
000227        [ 74]  SEI           23  **
000228        [ 75]  SKP            3  **
000229        [ 76]  STA         1406  *******************************
000230        [ 77]  STX           84  ***
000231        [ 78]  STY           78  ***
000232        [ 79]  TAX           93  ***
000233        [ 80]  TAY           76  ***
000234        [ 81]  TSTERR         1  *
000235        [ 82]  TSX            7  **
000236        [ 83]  TXA           82  ***
```

```
    [ 84]  TXS               6  **
    [ 85]  TYA              58  ***
    [ 86]  ZZLEN-LENLODR     1  *

  Sorted by Static Frequency:

    Opcode count :    86
    Min frequency:     1
    Max frequency:  1864

      #    Opcode Name   Freq   Histogram
     ---   ------------  -----  ----------------------------------------
    [  1]  LDA            1864  ****************************************
    [  2]  STA            1406  *******************************
    [  3]  EQU            1134  *************************
    [  4]  JSR             546  *************
    [  5]  LDY             450  ***********
    [  6]  REP             417  **********
    [  7]  BNE             397  **********
    [  8]  RTS             324  ********
    [  9]  BEQ             291  ********
    [ 10]  CMP             286  *******
    [ 11]  BCS             274  *******
    [ 12]  LDX             266  *******
    [ 13]  PAGE            246  *******
    [ 14]  EXTRN           242  *******
    [ 15]  BCC             240  *******
    [ 16]  DS              210  ******
    [ 17]  JMP             199  ******
    [ 18]  ENTRY           190  *****
    [ 19]  DFB             188  *****
    [ 20]  CLC             181  *****
    [ 21]  AND             147  *****
    [ 22]  INY             143  ****
    [ 23]  SEC             136  ****
    [ 24]  INC             131  ****
    [ 25]  DEY             126  ****
    [ 26]  ADC             125  ****
    [ 27]  BPL             121  ****
    [ 28]  SBC             111  ****
    [ 29]  ORA             100  ****
    [ 30]  LSR              93  ***
    [ 31]  TAX              93  ***
    [ 32]  PLA              91  ***
    [ 33]  PHA              90  ***
    [ 34]  STX              84  ***
    [ 35]  DW               83  ***
    [ 36]  TXA              82  ***
    [ 37]  STY              78  ***
    [ 38]  TAY              76  ***
```

```
[ 39]  DEX                75  ***
[ 40]  DEC                73  ***
[ 41]  BMI                59  ***
[ 42]  BIT                58  ***
[ 43]  TYA                58  ***
[ 44]  CPY                56  ***
[ 45]  ASL                46  **
[ 46]  CPX                40  **
[ 47]  INX                33  **
[ 48]  ASC                32  **
[ 49]  SBTL               32  **
[ 50]  CHN                28  **
[ 51]  FIN                27  **
[ 52]  ORG                24  **
[ 53]  PHP                23  **
[ 54]  ROR                23  **
[ 55]  SEI                23  **
[ 56]  MSB                22  **
[ 57]  EOR                21  **
[ 58]  INCLUDE            21  **
[ 59]  PLP                20  **
[ 60]  FAIL               17  **
[ 61]  IFNE               16  **
[ 62]  REL                16  **
[ 63]  ROL                14  **
[ 64]  LST                13  **
[ 65]  BRK                12  **
[ 66]  BVC                12  **
[ 67]  DO                 10  **
[ 68]  BVS                 7  **
[ 69]  TSX                 7  **
[ 70]  TXS                 6  **
[ 71]  CLI                 5  **
[ 72]  ELSE                5  **
[ 73]  CLV                 4  **
[ 74]  DEND                4  **
[ 75]  DSECT               4  **
[ 76]  CLD                 3  **
[ 77]  RTI                 3  **
[ 78]  SKP                 3  **
[ 79]  .PAGE               2  **
[ 80]  NOP                 2  **
[ 81]  CHR                 1  *
[ 82]  ERROR               1  *
[ 83]  IBUFSIZ             1  *
[ 84]  SBUFSIZ             1  *
[ 85]  TSTERR              1  *
[ 86]  ZZLEN-LENLODR       1  *
```

```
### FINIS: Assembly Source Code File Beautifier  [0.8]  28-Mar-93


```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.BFM.INIT2.SRC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: BFM.INIT2.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1  BFM.INIT2"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGBFMI
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC.  1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* BLOCK FILE MANAGER INIT2
*
*  SECONDARY INITIALIZATION ROUTINE FOR BLOCK FILE MANAGER
*
* MODIFIED: 03/25/81 TO UTILIZE NEW
*   DISK DRIVER'S SEEKDSK3 ROUTINE.
*  CHANGES MARKED BY 'D3RRA81084'
*
* MODIFIED: 08/19/81 TO WORK WITH NEW
*   SOSLDR MODULE.
                REP        60
*
                ENTRY      BFM.INIT2
*
*EXTRN I.BASE.P ; ENTRY IN SOSLDR
                EXTRN      SYSBANK
                EXTRN      SXPAGE
                EXTRN      CZPAGE
                EXTRN      SEEKDSK3             ;IN DISKDH/D3RRA81084
                EXTRN      NMIDSBL              ;/D3RRA81084
I.BASE.P        EQU        $2
                PAGE
*
* CONSTANTS
*
KERNEL.BASE     EQU        $B800                ; BASE ADDRESS OF SOS KERNEL
ROMID           EQU        $A0                  ;$F1B9 OF NEW ROM/D3RRA81084
SLOT            EQU        $60
BEGTRK          EQU        $9
BEGSECT         EQU        $2
```

```
ENDSECT         EQU        $6
*
* ZERO PAGE
*
TRACK           EQU        $99
SECTOR          EQU        $98
VOLUME          EQU        $9A
KEY             EQU        $E0                     ; THRU $E7
PREV.K          EQU        KEY+$8
XIDX            EQU        KEY+$9
I               EQU        KEY+$A                  ; & $B
*
* ROM ROUTINES
*
RDADR           EQU        $F1B9                   ;REV1
RDADRX          EQU        $F1BD                   ;REV0
*
* HARDWARE LOCATIONS
*
E.REG           EQU        $FFDF
B.REG           EQU        $FFEF
MOTORON         EQU        $C089
MOTOROFF        EQU        $C088
                PAGE
                REP        60
*
* BFM.INIT2 ENTRY POINT
*
                REP        60
*
STATE           DFB        $FE                     ; FF=1ST ENTRY, 0=2ND ENTRY, 1=PROT
*
BFM.INIT2       EQU        *
                INC        STATE
                BMI        BFMI050
                JSR        GETK
                LDA        RETRY
                BEQ        BADNEWS
                BCC        BFMI050
                JSR        NMIDSBL
                JSR        DC
                INC        STATE
BFMI050         CLC
                RTS
BADNEWS         SEC                                ; I/O ERROR
                RTS
                PAGE
                REP        60
*
* DECODE SUBROUTINE
```

```
*
* TO ENCODE:
*    E0.E8:        - INIT KEY  & PREV.K
*    B84E:4C 64 B8 - JUMPS AROUND INTERP'S 3 BYTE OVERWRITE
*    1A02.1A03:    - NEW INTERP'S LOAD ADR (LO,HII)
*    B81DG:        - JSR FROM MONITOR
*
                REP        60
DC              EQU        *
                LDA        B.REG                ; SAVE BANK REGISTER
                PHA
                LDA        SYSBANK              ;    AND SWITCH TO SYSTEM BANK
                STA        B.REG
                CLC                             ; FETCH LOADER'S INTERPRETER POINTER
                LDA        CZPAGE+I.BASE.P
                ADC        #3
                STA        I
                PHA
                LDA        CZPAGE+I.BASE.P+1
                ADC        #0
                STA        I+1
                PHA
                LDA        #0
                STA        SXPAGE+I+1
*
                LDY        I                    ; ALIGN I PTR TO PAGE BOUNDARY
                LDA        #0
                STA        I
                STA        PREV.K
*
                JSR        DCLOOP               ; DECODE
*
                PLA                             ; RETRIEVE LOADER'S INTERPRETER POINTER
                STA        I+1
                PLA
                STA        I
*
                LDY        #1                   ; REPOSITION LOADER'S INTERPRETER POINTER (PUT ENCODE JMP HERE)
                LDA        (I),Y
                STA        CZPAGE+I.BASE.P
                INY
                LDA        (I),Y
                STA        CZPAGE+I.BASE.P+1
*
                LDY        #2                   ; WALK ON INTERPRETER'S FIRST INSTRUCTION (3 BYTES)
                LDA        #0
DCA             STA        (I),Y
                DEY
                BPL        DCA
                PLA                             ; RESTORE BANK REGISTER (ENCODE JMP JUMPS TO HERE)
                STA        B.REG
                RTS
                PAGE
                REP        60
*
* DECODE LOOP SUBROUTINE
*
                REP        60
DCLOOP          EQU        *
                LDX        #7                    ; SHIFT LEFT ONE BIT
                CLC
                LDA        KEY
                BPL        DC1
                SEC
DC1             ROL        KEY,X
                DEX
                BPL        DC1
*
DC2             TYA
                AND        #7
                EOR        #2
                TAX
                LDA        KEY,X
                PHA
                AND        #7
                TAX
                PLA
                CLC
                ADC        PREV.K
                CLC
                ADC        KEY,X
                STA        PREV.K
                EOR        (I),Y                 ; DECODE BYTE
                STA        (I),Y                 ; AND PUT IT BACK
                INY
                BNE        DC2
                INC        I+1
                LDA        I+1
                CMP        #<KERNEL.BASE
                BCC        DCLOOP
                RTS
                PAGE
                REP        60
*
* GETKEY SUBROUTINE
*
                REP        60
*
RETRY           DFB        10+1                  ;TEN RETRIES
OURTRACK        DS         1                     ;CURRENT TRACK/D3RRA81084
```

```
*
GETK            EQU        *
                LDX        #7
                STX        XIDX
                LDX        #SLOT
                LDA        MOTORON,X            ;ENSURE MOTOR STAYS ON
                LDA        E.REG                ; SELECT 1MHZ, ROM
                ORA        #$83
                STA        E.REG
*
* NOTE: THE SEEKDSK3 ROUTINE HAS THESE /D3RRA81084
*  CAVEATS: 1MHZ MODE, MOTOR IS ON, /D3RRA81084
*  DRIVE CURRENTLY SELECTED, ROM+I/O ENABLED! /D3RRA81084
*
GETK010         LDA        #BEGTRK
                STA        OURTRACK             ;WHERE WE SEEK TO /D3RRA81084
                JSR        SEEKDSK3             ;HAVE DISKDH SEEK FOR US /D3RRA81084
GETK020         LDX        #SLOT
                JSR        DOREAD               ;FIND A SECTOR HEADER
                BCS        IOERROR              ;=>RETRY IF BAD
                LDA        SECTOR               ;WHERE ARE WE?
                CMP        #BEGSECT             ;AT THE RIGHT PLACE?
                BNE        GETK020              ;=>NO, GET THERE
*
GETK100         LDX        #1
                JSR        WAIT                 ; (X * 1284) + 15 MILISECONDS
                LDX        XIDX
                LDA        VOLUME
                STA        KEY,X
                DEC        XIDX
                BMI        ENUFF
                INC        OURTRACK             ;BUMP FOR NEXT TRACK /D3RRA81084
                LDA        OURTRACK             ;WHERE TO GO /D3RRA81084
                LDX        #SLOT
                JSR        SEEKDSK3             ;DISKDH, PLEASE SEEK ME /D3RRA81084
                LDX        #SLOT
                JSR        DOREAD
                BCC        GETK100
                BCS        IOERROR
*
ENUFF           LDX        #SLOT
                LDA        MOTOROFF,X
                LDA        E.REG                ; SELECT 2MHZ, RAM
                AND        #$7C
                STA        E.REG
                PAGE
                LDA        SECTOR
                CMP        #ENDSECT             ;TRACKS SYNC'ED?
                BNE        NOTPROT
                LDA        KEY
                EOR        KEY+1
                BEQ        NOTPROT              ;IF FIRST 2 VOLS ARE EQUAL
                SEC
                RTS
*
NOTPROT         LDA        #0
                CLC
                RTS
*
*
DOREAD          JSR        WHICHROM
                BCS        OLDREAD
                JMP        RDADR
OLDREAD         JMP        RDADRX
*
*
WHICHROM        LDA        RDADR
                CMP        #ROMID
                CLC
                BEQ        NEWROM
                SEC
NEWROM          RTS
*
*
IOERROR         DEC        RETRY
                BEQ        ERR1
                JMP        GETK                 ; TRY, TRY AGAIN
ERR1            JMP        ENUFF                ; I/O ERROR, CLEANUP AND EXIT
*
*
WAIT            LDY        #0
W1              DEY
                BNE        W1
                DEX
                BNE        W1
                RTS

ZZLEN           EQU        $400
                IFNE       ZZLEN-LENBFMI
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR BFM.INIT2"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: BFM.INIT2.SRC
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.INIT.SRC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: INIT.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1 INITIALIZATION"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGINIT
ZZORG           EQU        *
                MSB        OFF
                REP        100
*          COPYRIGHT (C) APPLE COMPUTER INC.  1981
*                    ALL RIGHTS RESERVED
                REP        100
*
* SOS INIT MODULE (VERSION = 1.10   )
*                 (DATE    = 8/04/81)
*
                REP        100
*
                ENTRY      INT.INIT
                ENTRY      EVQ.INIT
                ENTRY      CLK.INIT
                ENTRY      MMGR.INIT
                ENTRY      BMGR.INIT
                ENTRY      DMGR.INIT
                ENTRY      CFMGR.INIT
                ENTRY      BFM.INIT
*
*  EXTERNAL SUBROUTINES & DATA
*
                EXTRN      SXPAGE
                EXTRN      SYSDEATH
*
*  INTERRUPT SYSTEM INITIALIZATION
*
                EXTRN      COLDSTRT
                EXTRN      IRQ.RCVR
                EXTRN      NMI.RCVR
                EXTRN      NMIFLAG
                EXTRN      SIRTABLE
                EXTRN      SIRTBLSIZ
                EXTRN      ZPGSTACK
                EXTRN      ZPGSTART
```

```
*
*  EVENT QUEUE INITIALIZATION
*
                EXTRN      EV.QUEUE
                EXTRN      EVQ.LEN
                EXTRN      EVQ.CNT
                EXTRN      EVQ.SIZ
                EXTRN      EVQ.FREE
                EXTRN      EVQ.LINK
*
*  CLOCK INITIALIZATION
*
                EXTRN      PCLOCK
*
*  CHARACTER FILE MANAGER INITIALIZATION
*
                EXTRN      CFCB.MAX
                EXTRN      CFCB.DEV
*
*  DEVICE MANAGER INITIALIZATION
*
                EXTRN      DMGR
                EXTRN      MAX.DNUM
*
*  BUFFER MANAGER INITIALIZATION
*
                EXTRN      BUF.CNT
                EXTRN      PGCT.T
                EXTRN      XBYTE.T
                EXTRN      BUFREF
*
*  MEMORY MANAGER INITIALIZATION
*
                EXTRN      ST.CNT
                EXTRN      ST.ENTRY
                EXTRN      ST.FREE
                EXTRN      ST.FLINK
                EXTRN      VRT.LIM
                EXTRN      MEMSIZE
                EXTRN      MEM2SML
*
*  BLOCK FILE MANAGER INITIALIZATION
*
                EXTRN      FCBZPP
                EXTRN      PATHBUF
                EXTRN      VCB
                EXTRN      WORKSPC
                EXTRN      PFIXPTR
                EXTRN      FCBADDRH
                EXTRN      BMAPAGE
                EXTRN      BMBPAGE
                EXTRN      BMAMADR
                EXTRN      BMBMADR
                EXTRN      BFMFCB1
                EXTRN      BFMFCB2
*
*  CONSTANT DECLARATIONS
*
TRUE            EQU        $80
FALSE           EQU        $00
BITON6          EQU        $40
BITON7          EQU        $80
*
*  SYSTEM CONTROL REGISTERS
*
E.REG           EQU        $FFDF                ;ENVIRONMENT REGISTER
Z.REG           EQU        $FFD0                ;ZERO PAGE REGISTER
                SBTL       "INTERRUPT SYSTEM INITIALIZATION"
*
*  6522 REGISTERS
*
D.DDRB          EQU        $FFD2
D.DDRA          EQU        $FFD3
D.ACR           EQU        $FFDB
D.PCR           EQU        $FFDC
D.IFR           EQU        $FFDD
D.IER           EQU        $FFDE
E.IORB          EQU        $FFE0
E.DDRB          EQU        $FFE2
E.DDRA          EQU        $FFE3
E.ACR           EQU        $FFEB
E.PCR           EQU        $FFEC
E.IFR           EQU        $FFED
E.IER           EQU        $FFEE
ACIASTAT        EQU        $C0F1
*
*
                REP        60
*
*  THIS SUBROUTINE INITIALIZES THE INTERRUPT SYSTEM.
*  ALL HARDWARE INTERRUPTS ARE MASKED AND THE
*  INTERRUPT ALLOCATION TABLE IS CLEARED.
*
                REP        60
*
*
INT.INIT        EQU        *
                SEI                             ;DISABLE INTERRUPTS
                LDA        #>ZPGSTART           ;SET UP MIH
                STA        ZPGSTACK             ;  ZERO PAGE STACK POINTER
```

```
*
                LDA        E.REG                ;SELECT $C000 I/O SPACE
                PHA                             ;  AND SET 1 MHZ
                ORA        #BITON7+BITON6
                STA        E.REG
*
                STA        ACIASTAT             ;RESET ACIA
*
                LDA        #$FF                 ;SET UP 6522 D
                STA        D.DDRB
                STA        D.DDRA
                LDA        #$00
                STA        D.ACR
                LDA        #$76
                STA        D.PCR
                LDA        #$7F
                STA        D.IFR
                STA        D.IER
                LDA        #$82
                STA        D.IER
*
                LDA        #$3F                 ;SET UP 6522 E
                STA        E.DDRB
                LDA        #$0F
                STA        E.DDRA
                LDA        #$00
                STA        E.ACR
                LDA        #$63
                STA        E.PCR
                LDA        #$7F
                STA        E.IFR
                STA        E.IER
*
                LDA        #$FF
                STA        E.IORB               ;SOUND PORT
                BIT        $C0D8                ;DISABLE GRAPHICS SCROLL
                BIT        $C0DA                ;DISABLE CHARACTER DOWNLOAD
                BIT        $C0DC                ;DISABLE ENSEL
                BIT        $C0DE                ;SET ENSIO FOR INPUT
*
                PLA                             ;RESTORE E REGISTER
                STA        E.REG
*
                LDA        #FALSE
                STA        NMIFLAG              ;CLEAR NMI WAIT FLAG
                LDY        #>SIRTBLSIZ-1
INTI010         STA        SIRTABLE,Y           ;  ALLOCATION TABLE
                DEY
                BPL        INTI010
                LDA        #TRUE
                STA        SIRTABLE+$0A         ;LOCK DOWN ANY SLOT SIR
*
                LDX        #$05
INTI020         LDA        RAMVECT,X            ;SET UP VECTORS
                STA        $FFFA,X              ;  AT $FFFA - $FFFF
                LDA        RAMJMPS,X            ;SET UP JMP INSTRUCTIONS
                STA        $FFCA,X              ;  AT $FFCA - $FFCF
                DEX
                BPL        INTI020
                RTS
*
RAMVECT         DW         NMI.RCVR
                DW         COLDSTRT
                DW         IRQ.RCVR
RAMJMPS         JMP        NMI.RCVR
                JMP        IRQ.RCVR
                SBTL       "EVENT QUEUE INITIALIZATION"
                REP        60
*
*  THIS SUBROUTINE INITIALIZES THE EVENT QUEUE.  ALL ENTRIES
*  ARE CLEARED AND LINKED INTO THE FREE LIST.  THE ACTIVE
*  LIST IS EMPTY.
*
                REP        60
*
*
EVQ.INIT        EQU        *
*
*  CLEAR ALL ENTRIES
*
                LDY        #>EVQ.LEN
                LDA        #0
EVQI010         STA        EV.QUEUE-1,Y
                DEY
                BNE        EVQI010
*
*  SET UP FREE LIST
*
                LDX        #>EVQ.CNT-2
                LDA        #>EVQ.SIZ
                STA        EVQ.FREE
EVQI020         TAY
                CLC
                ADC        #>EVQ.SIZ
                STA        EVQ.LINK,Y
                DEX
                BNE        EVQI020
                RTS
                SBTL       "PSEUDO CLOCK INITIALIZATION"
                REP        60
```

```
*
*  THIS SUBROUTINE INITIALIZES THE PSEUDO CLOCK.  IF THE
*  RAM BEHIND THE "D" 6522 HAS THE PROPER CHECKSUM, IT
*  IS USED TO INITIALIZE THE PSEUDO CLOCK.  OTHERWISE,
*  THE PSEUDO CLOCK IS SET TO ZERO.
*
* (ADDED 23 OCT 81)
* BOTH THE CLOCK AND PSEUDO CLOCK ARE
* ARE NOW INITIALIZED
*
                REP        60
*
PCLK            EQU        $F0
CKSUM           EQU        $F2
CLKICR          EQU        $11                 ; CLOCK INTERRUPT CONTROL REG
CLKSTBY         EQU        $16                 ; CLOCK STANDBY INTERRUPT
CLOCK           EQU        $C070
*
CLK.INIT        EQU        *
                LDA        #$D0
                STA        PCLK                ;POINT (PCLK) TO 8F:FFD0
                LDA        #$FF
                STA        PCLK+1
                LDA        #$8F
                STA        SXPAGE+PCLK+1
                LDA        #$A5
                STA        CKSUM               ;INITIALIZE CHECKSUM
*
                LDY        #$00
CLK010          LDA        (PCLK),Y            ;COPY SAVED CLOCK DATA
                STA        PCLOCK,Y            ;  TO PSEUDO CLOCK
                EOR        CKSUM
                STA        CKSUM               ;UPDATE CHECKSUM
                INY
                CPY        #$0A
                BCC        CLK010
*
                CMP        (PCLK),Y            ;TEST CHECKSUM
                BEQ        CLK030
*
                LDA        #$00
CLK020          DEY
                STA        PCLOCK,Y            ;ZERO PSEUDO CLOCK
                BNE        CLK020
CLK030          LDA        E.REG
                PHA
                ORA        #$80                ; SET 1 MHZ
                STA        E.REG
                LDA        #$00
                LDY        Z.REG
                LDX        #CLKICR
                STX        Z.REG
                STA        CLOCK                ; DISABLE CLOCK INTERRUPTS
                LDX        #CLKSTBY
                STX        Z.REG
                STA        CLOCK                ; DISABLE STANDBY INTERRUPT
                STY        Z.REG
                PLA
                STA        E.REG
                RTS
                SBTL       "CHARACTER FILE MANAGER INITIALIZATION"
                REP        60
*
* CHAR FILE MANAGER INITIALIZATION ROUTINE
*
* CFMGR.INIT INITIALIZES ALL ENTRIES IN THE CFCB TABLE TO
* THE "FREE" STATE.
*
                REP        60
*
CFMGR.INIT      EQU        *
                LDA        #$80
                LDX        #CFCB.MAX-1
CFINIT010       STA        CFCB.DEV,X
                DEX
                BPL        CFINIT010
                RTS
                SBTL       "DEVICE MANAGER INITIALIZATION"
                REP        60
*
* DEVICE MANAGER INITIALIZATION ROUTINE
*
* INITIALIZES THE SYSTEM DEVICE TABLE (SDT) BY WALKING THE
* DEVICE INFORMATION BLOCK (DIB) LINKS.  CALLED BY SYSLDR.
*
                REP        60
*
D.TPARMX        EQU        $C0
REQCODE         EQU        D.TPARMX+$00
DNUM            EQU        D.TPARMX+$01
DNUM.TEMP       DS         1
*
*
DMGR.INIT       EQU        *
                LDX        MAX.DNUM
                INC        MAX.DNUM             ; MAX.DNUM:=MAX DEV NUMBER IN SYSTEM+1
                STX        DNUM.TEMP
DMI110          LDA        #8                   ; INITIALIZE ALL DEVICES IN SYSTEM (D.INIT)
                STA        REQCODE
                LDA        DNUM.TEMP
                STA        DNUM
                JSR        DMGR
                DEC        DNUM.TEMP
                BNE        DMI110
                RTS                             ; NORMAL EXIT
                SBTL       "BUFFER MANAGER INITIALIZATION"
                REP        60
*
* BMGR.INIT
*
* THIS ROUTINE INITIALIZES THE BUFFER TABLE'S ENTRIES TO "FREE".
* CALLED DURING SYSTEM BOOT.
*
                REP        60
*
BMGR.INIT       EQU        *
                LDA        #$FF                 ; USED WHEN FINDING LOWEST BUFFER IN TBL (BUFCOMPACT)
                STA        XBYTE.T
*
                LDX        #BUF.CNT-1
                LDA        #$80
BUFI010         STA        PGCT.T,X             ;SET ALL ENTRIES "FREE"
                DEX
                BNE        BUFI010
*
                STX        BUFREF               ;ZERO COUNT BYTE IN BUFFER REFERENCE TABLE
*
                CLC
                RTS
                SBTL       "MEMORY MANAGER INITIALIZATION"
                REP        60
*
* MMGR.INIT
*
* THIS ROUTINE INITIALIZES THE MEMORY MANAGER'S SEGMENT TABLE
* TO FREE ENTRIES, AND DETERMINES THE MEMORY SIZE OF THE
* MACHINE (96K,128K,160K,192K,224K,256K,..,512K IN 32K STEPS).
*
                REP        60
*
MMGR.INIT       EQU        *
*
* INIT SEGMENT TABLE
*
                LDA        #0
                STA        ST.ENTRY
                LDA        #$81
                STA        ST.FREE
*
                LDY        #ST.CNT-1
                LDA        #$80                 ; SET LAST LINK TO NULL
                STA        ST.FLINK,Y
MEMI010         TYA
                ORA        #$80
                DEY
                STA        ST.FLINK,Y
                BNE        MEMI010
*
* COMPUTE VIRTUAL LIMIT FROM MEMORY SIZE
* VRT.LIM := NUMBER OF PAGES IN BANK SWITCHED MEMORY - 1
*         := (MEMSIZ-2)*64 - 1
*         := (MEMSIZ-4)*64 + 127
*
                SEC
                LDA        MEMSIZE
                SBC        #4
                BCC        MEMI.ERR
                LSR        A
                LSR        A
                STA        VRT.LIM+1
                LDA        #$FE
                ROR        A
                STA        VRT.LIM
                CLC
                RTS                             ; NORMAL EXIT
*
MEMI.ERR        LDA        #MEM2SML             ; FATAL ERR - MEM < 64K
                JSR        SYSDEATH
                PAGE
                REP        60
*
*  BLOCK FILE MANAGER INITIALIZATION
*
                REP        60
*
SISTER          EQU        $1400                ;BFM XPAGE
BFM.INIT        EQU        *
                LDA        #BFMFCB1             ; ADDRESS OF PAGE 1 OF FCB
                STA        >FCBZPP+1
                LDA        #BFMFCB2             ; AND PAGE 2
                STA        >FCBZPP+3
                LDA        #0
                STA        >FCBZPP              ; FCB PAGE ALIGNED
                STA        >FCBZPP+2
                STA        SISTER+FCBZPP+1      ; PREPARE PART OF EXTEND BYTE
                STA        SISTER+FCBZPP+3
                TAY                             ; MAKE ZERO INTO INDEX
CLRBUFFS        EQU        *
                STA        PATHBUF,Y            ; PATHNAME BUFFER PAGE
                STA        VCB,Y                ; VOLUME CONTROL BLOCK PAGE
                STA        (>FCBZPP),Y          ; BOTH FILE CONTROL BLOCK PAGES
                STA        (>FCBZPP+2),Y
                INY
                BNE        CLRBUFFS
                LDX        #$3F                 ; SIZE OF MY ZERO PAGE STUFF
CLRZWRK         STA        0,X                  ; ZERO PAGE ZEROED
                STA        WORKSPC,X
                DEX
                BPL        CLRZWRK
                LDA        #<PATHBUF
                STA        PFIXPTR+1
                LDA        #BFMFCB1
                STA        FCBADDRH
                LDA        #BMAPAGE             ; BIT MAP A PAGE NUMBER
                STA        BMAMADR
                LDA        #BMBPAGE             ; BIT MAP B PAGE NUMBER
                STA        BMBMADR
                CLC
                RTS
*
                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENINIT
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR INIT"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: INIT.SRC
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.IPL.SRC1.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC1
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1  INTRPTS. & PROC. LAUNCH"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGIPL
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
*  THIS MODULE IS RESPONSIBLE FOR FIELDING ALL INTERRUPTS
*  AND RELAUNCHING THE INTERRUPTED CODE AFTER THE INTERRUPTS
*  HAVE BEEN PROCESSED.  THE MAJOR FUNCTIONAL AREAS ARE:
*
*       GENERAL INTERRUPT RECEIVER
*       NMI INTERRUPT RECEIVER
*       DISPATCHER
*       INTERRUPT ALLOCATION & DEALLOCATION
*       EVENT QUEUE MANAGER
*       TABLE INITIALIZATION
*
                REP        60
*
*  SUBROUTINE ENTRY POINTS
*
                ENTRY      IRQ.RCVR             ;GENERAL INTERRUPT RECEIVER
                ENTRY      NMI.RCVR             ;NON-MASKABLE INTRPT RCVR
                ENTRY      DISPATCH             ;DISPATCHER
                ENTRY      ALLOCSIR             ;SIR ALLOCATION
                ENTRY      DEALCSIR             ;SIR DEALLOCATION
                ENTRY      SELC800              ;SELECT I/O EXPANSION ROM
                ENTRY      NMIDSBL              ;DISABLE NMI
                ENTRY      NMIENBL              ;ENABLE NMI
                ENTRY      NMIDBUG              ;NMI DEBUG ENTRY
                ENTRY      NMICONT              ;NMI DEBUG CONTINUATION
                ENTRY      QUEEVENT             ;QUEUE AN EVENT
*
*  EXTERNAL SUBROUTINES & DATA
*
```

```
                EXTRN      SCMGR
                EXTRN      CHKBUF
*
*  SYSTEM DEATH ERRORS
*
                EXTRN      SYSDEATH
                EXTRN      BADBRK
                EXTRN      BADINT1
                EXTRN      BADINT2
                EXTRN      NMIHANG
                EXTRN      EVQOVFL
                EXTRN      STKOVFL
*
*  LINKAGE DATA FOR INITIALIZATION ROUTINES
*
                ENTRY      EV.QUEUE
                ENTRY      EVQ.CNT
                ENTRY      EVQ.SIZ
                ENTRY      EVQ.LEN
                ENTRY      EVQ.FREE
                ENTRY      EVQ.LINK
                ENTRY      SIRTABLE
                ENTRY      SIRTBLSIZ
                ENTRY      ZPGSTACK
                ENTRY      ZPGSTART
*
*  SYSGLOB DATA
*
                EXTRN      SERR
                EXTRN      CEVPRI               ;CALLER'S EVENT PRIORITY
                EXTRN      SYSBANK              ;SYSTEM BANK
                EXTRN      KYBDNMI
                EXTRN      NMISPSV
                EXTRN      NMIFLAG              ;NMI PENDING FLAG
                EXTRN      SCRNMODE             ;CURRENT SCREEN MODE
                EXTRN      SIRTEMP              ;FOR ALLOCSIR & DEALCSIR
                EXTRN      SIRARGSIZ
                EXTRN      IRQCNTR              ;FLASE IRQ COUNTER
                EXTRN      NMICNTR              ;TWO BYTE COUNTER
                EXTRN      QEVTEMP
                EXTRN      QEV.THIS
                EXTRN      QEV.LAST
                EXTRN      BACKMASK
*
*  CONSTANT DECLARATIONS
*
FALSE           EQU        $00
BITON0          EQU        $01
BITON1          EQU        $02
BITON2          EQU        $04
```

```
BITON4          EQU        $10
BITON5          EQU        $20
BITON6          EQU        $40
BITON7          EQU        $80
BITOFF3         EQU        $F7
BITOFF4         EQU        $EF
BITOFF5         EQU        $DF
BITOFF6         EQU        $BF
BITOFF7         EQU        $7F
BACKBIT         EQU        $20                  ; BACKUP BIT MASK
*
*  SYSTEM CONTROL REGISTERS
*
B.REG           EQU        $FFEF                ;BANK REGISTER
E.REG           EQU        $FFDF                ;ENVIRONMENT REGISTER
Z.REG           EQU        $FFD0                ;ZERO PAGE REGISTER
*
*  6522 REGISTERS
*
D.IFR           EQU        $FFDD
D.IER           EQU        $FFDE
E.IORB          EQU        $FFE0
E.IFR           EQU        $FFED
E.IER           EQU        $FFEE
E.IORA          EQU        $FFEF
                PAGE
*
*  REGISTER PRESERVATION EQUATES
*  FOR USE DURING INTERRUPT PROCESSING
*
A.SAVE          EQU        $103
S.SAVE          EQU        $104
SP.SAVE         EQU        $1FF
E.SAVE          EQU        $1FE
Z.SAVE          EQU        $1FD
B.SAVE          EQU        $1FC
EXPNSLOT        DFB        $00                  ;CURRENT I/O EXPANSION SLOT
*
*  STATUS LOCATIONS FOR INTERRUPT POLLING
*
ACIASTAT        EQU        $C0F1
ANYSLOT         DFB        BITON1
SLOT1           EQU        $C065
SLOT2           EQU        $C064
SLOT3           DFB        BITON5
SLOT4           DFB        BITON4
*
*  INTERRUPT ZERO PAGE STORAGE & EQUATES
*
SIRARGS         EQU        $F9                  ;AND $FA
```

```
QEVARGS         EQU        $FB                     ;AND $FC
IRQADDR         EQU        $FD                     ;AND $FE
ZPGSP           EQU        $FF
ZPGSTART        EQU        $F8
ZPGSTOP         EQU        $28
ZPGSPACE        EQU        $20
ZPGSTACK        DFB        ZPGSTART
*
*  SYSTEM INTERNAL RESOURCE
*  TABLE STORAGE AND EQUATES
*
SIRTBLSIZ       EQU        $18
SIRTABLE        DS         SIRTBLSIZ
SIRADR.L        DS         SIRTBLSIZ
NMIADR.L        DS         1                       ;MUST PRECEED SIRADR.H
SIRADR.H        DS         SIRTBLSIZ
SIRADR.B        DS         SIRTBLSIZ
*
*  EVENT QUEUE STORAGE AND EQUATES
*
EVQ.SIZ         EQU        6                       ;ENTRY SIZE
EVQ.CNT         EQU        $07                     ;ENTRY COUNT
EVQ.LEN         EQU        $2A                     ;(EVQ.SIZ*EVQ.CNT)
EV.QUEUE        DS         EVQ.LEN
EVQ.FREE        EQU        EV.QUEUE+2              ;FIRST FREE ENTRY INDEX
EVQ.LINK        EQU        EV.QUEUE+0              ;NEXT ACTIVE ENTRY INDEX
EVQ.PRI         EQU        EV.QUEUE+1              ;EVENT PRIORITY
EVQ.ID          EQU        EV.QUEUE+2              ;EVENT IDENTIFICATION
EVQ.ADRL        EQU        EV.QUEUE+3              ;EVENT ADDRESS:  LOW BYTE
EVQ.ADRH        EQU        EV.QUEUE+4              ;EVENT ADDRESS:  HIGH BYTE
EVQ.BANK        EQU        EV.QUEUE+5              ;EVENT ADDRESS:  BANK
                SBTL       "GENERAL INTERRUPT RECEIVER"
                REP        60
*
*  THIS IS THE GENERAL INTERRUPT RECEIVER.  WHEN AN
*  INTERRUPT OCCURS, THE CPU PASSES CONTROL TO THE GIR
*  THROUGH THE IRQ VECTOR.  THE GIR IS RESPONSIBLE FOR
*  SAVING THE CURRENT ENVIRONMENT, SETTING UP THE SOS
*  ENVIRONMENT, AND CALLING THE APPROPRIATE CODE MODULE.
*  IF THE INTERRUPT WAS CAUSED BY A BRK, THE GIR CALLS
*  THE SYSTEM CALL MANAGER.  OTHERWISE, THE GIR POLLS THE
*  I/O DEVICES AND CALLS THE APPROPRIATE MASTER INTERRUPT
*  HANDLER.  WHEN THE SCM OR MIH RETURNS, THE GIR PASSES
*  CONTROL TO THE DISPATCHER.
*
                REP        60
*
IRQ.RCVR        EQU        *
*
*  SAVE CPU REGISTERS A, X, & Y ON CURRENT STACK
```

```
*
                PHA
                TXA
                PHA
                TYA
                PHA
*
*  CHECK FOR STACK OVERFLOW AND
*  SAVE INTERRUPTED STATUS IN Y REGISTER.
*
                TSX
                CPX        #$FA
                BCC        GIR005
                LDA        #>STKOVFL
                JSR        SYSDEATH
GIR005          LDY        S.SAVE,X
*
*  SET UP INTERRUPT ENVIRONMENT:
*    BINARY ARITHMETIC, 2 MHZ, I/O ENABLED,
*    RAM WRITE ENABLED, PRIMARY STACK,
*    AND $F000 RAM SELECTED.  PRESERVE
*    USER STATE OF SCREEN AND RESET LOCK.
*
                CLD
                LDA        E.REG
                TAX
                AND        #BITON5+BITON4
                ORA        #BITON6+BITON2
                STA        E.REG
*
*  IF NOT ALREADY ON PRIMARY STACK, SAVE USER'S STACK
*  POINTER AND SET UP SOS STACK POINTER.
*
                TXA
                AND        #BITON2
                BNE        GIR010
                TXA
                TSX
                STX        SP.SAVE
                LDX        #>E.SAVE
                TXS
                TAX
*
*  SAVE E, Z, B, & I/O EXPANSION SLOT ON SOS STACK
*  IF BRK THEN CALL SCMGR ELSE POLL I/O DEVICES
*
GIR010          TXA
                PHA
                LDA        Z.REG
                PHA
                LDA        B.REG
                PHA
                LDA        EXPNSLOT
                PHA
                BIT        $CFFF
                BIT        $C020                ;RESET I/O SPACE
                LDA        #$00
                STA        EXPNSLOT
                TYA
                AND        #BITON4
                BEQ        POLL.IO
*
*  CALL SYSTEM CALL MANAGER; ON RETURN, PUT ERROR CODE IN
*  USER'S A REGISTER AND SET RETURN STATUS, THEN DISPATCH.
*
                TSX                             ;CHECK FOR
                CPX        #>B.SAVE-2           ;  REENTRANT
                BEQ        GIR020               ;  SYSTEM CALL
                LDA        #>BADBRK
                JSR        SYSDEATH
GIR020          LDA        E.REG                ;SELECT $C000 RAM
                AND        #BITOFF6
                STA        E.REG
                CLI                             ;ENABLE INTERRUPTS
                JSR        SCMGR                ;CALL THE SYSTEM CALL MGR
                LDA        #BACKBIT             ; GET THE MASK
                STA        BACKMASK             ; SET IT IN SYSGLOB
                JSR        CHKBUF
                SEI
                LDX        SP.SAVE
                LDA        Z.SAVE
                EOR        #BITON0              ;SET ZERO PAGE TO
                STA        Z.REG                ;  CALLER'S STACK
                LDA        SERR
                STA        >A.SAVE,X
                PHP
                LDA        >S.SAVE,X
                AND        #$7D
                STA        >S.SAVE,X
                PLA
                AND        #$82
                ORA        >S.SAVE,X
                STA        >S.SAVE,X
                JMP        DISPATCH
                PAGE
*
*  SET INTERRUPT ZERO PAGE AND SOS BANK
*    THEN POLL I/O DEVICES
*
POLL.IO         BIT        E.IORA               ;VERIFY THAT 'IRQ IS LOW
```

```
                BPL        PIO006
                INC        IRQCNTR              ;BUMP FALSE IRQ COUNTER
                BNE        PIO004
                INC        IRQCNTR+1
PIO004          JMP        DISPATCH
PIO006          LDA        #0                   ;SET INTERRUPT ZERO PAGE
                STA        Z.REG
                LDA        E.REG
                ORA        #BITON7              ;FORCE 1 MHZ FOR
                STA        E.REG                ;  READING ACIA STATUS
                AND        #BITOFF7
                LDX        #$01
                LDY        ACIASTAT             ;ANY INTERRUPT ON ACIA?
                STA        E.REG
                BMI        PIO070
                LDA        E.IFR                ;ANY INTERRUPT ON E-6522?
                BPL        PIO020               ;  NO
                AND        E.IER
                LDY        #7
                LDX        #$02
PIO010          LSR        A                    ;CHECK FLAG BITS
                BCS        PIO070
                INX
                DEY
                BNE        PIO010
                BEQ        PIO035
PIO020          LDA        D.IFR                ;ANY INTERRUPT ON D-6522?
                BPL        PIO035
                AND        D.IER
                BIT        ANYSLOT              ;ANY SLOT INTERRUPT?
                BNE        PIO040               ;  YES
                LDY        #7
                LDX        #$09
PIO030          LSR        A                    ;CHECK FLAG BITS
                BCS        PIO070
                INX
                DEY
                BNE        PIO030
PIO035          LDX        #$10                 ;INTERRUPT NOT FOUND
                BNE        PIO050
PIO040          LDX        #$11
                BIT        SLOT1                ;SLOT 1?
                BPL        PIO070
                INX
                BIT        SLOT2                ;SLOT 2?
                BPL        PIO070
                LDA        E.IORA
                INX
                BIT        SLOT3                ;SLOT 3?
                BEQ        PIO070
                INX
                BIT        SLOT4                ;SLOT 4?
                BEQ        PIO070
                LDX        #$0A
*
*  BAD INTERRUPT -- SYSTEM DEATH
*
PIO050          LDA        #>BADINT1            ;INTERRUPT NOT FOUND
                JSR        SYSDEATH
PIO060          LDA        #>BADINT2            ;BAD ZERO PAGE ALLOCATION
                JSR        SYSDEATH
*
*  INTERRUPTING DEVICE FOUND
*    ALLOCATE ZERO PAGE AND CALL MASTER INTERRUPT HANDLER
*
*  NOTE:
*    SINCE READING THE ACIA'S STATUS REGISTER RESETS THE
*    DSR AND DCD BITS, THE STATUS READ BY THE POLLING
*    ROUTINE MUST BE PASSED TO THE INTERRUPT HANDLER;
*    THE Y REGISTER HAS BEEN SELECTED FOR THIS PURPOSE.
*    THE CURRENT IMPLEMENTATION DOES NOT USE Y IN CALLING
*    THE INTERRUPT HANDLER.  IF SUBSEQUENT REVISIONS
*    NEED TO USE Y, THE STATUS MUST BE PRESERVED AND
*    RESTORED BEFORE CALLING THE INTERRUPT HANDLER.
*
CALLMIH         JMP        (IRQADDR)
*
PIO070          LDA        SIRTABLE,X           ;INTERRUPT ALLOCATED?
                BPL        PIO050               ;  NO
                LDA        SIRADR.L,X           ;GET INTERRUPT ADDRESS
                STA        IRQADDR
                ORA        SIRADR.H,X           ;CHECK FOR ADDRESS = $00
                BEQ        PIO050               ;  BAD ADDRESS
                LDA        SIRADR.H,X
                STA        IRQADDR+1
                LDA        SIRADR.B,X
                STA        B.REG
                LDA        ZPGSTACK             ;ALLOCATE MIH ZERO PAGE
                CMP        #ZPGSTOP+ZPGSPACE
                BCC        PIO060               ;TOO MANY NESTED INTERRUPTS
                SBC        #ZPGSPACE
                STA        ZPGSTACK
                STA        ZPGSP
                TAX
                JSR        CALLMIH              ;CALL INTERRUPT HANDLER
                SEI
                LDA        #$00
                STA        Z.REG
                CLC
                LDA        ZPGSTACK             ;DEALLOCATE MIH ZERO PAGE
                ADC        #ZPGSPACE
                STA        ZPGSTACK
                STA        ZPGSP
                LDA        #BITON1
                STA        D.IFR                ;CLEAR ANY SLOT INTERRUPT
                JMP        DISPATCH
                SBTL       "NON-MASKABLE INTERRUPT RECEIVER"
                REP        60
*
*  THIS IS THE NON-MASKABLE INTERRUPT RECEIVER.  WHEN AN
*  NMI OCCURS, THE CPU PASSES CONTROL TO THE NMI RECEIVER
*  THROUGH THE NMI VECTOR.  THE OPERATION OF THE NMI
*  RECEIVER IS ESSENTIALLY THE SAME AS THE GIR EXCEPT
*  THAT IT IS NOT CONCERNED WITH BRK, AND THE ONLY VALID
*  SOURCE OF AN NMI IS THE KEYBOARD OR THE I/O DEVICE THAT
*  HAS ALLOCATED THE NMI RESOURCE.
*
                REP        60
*
*
NMI.RCVR        EQU        *
*
*  SAVE CPU REGISTERS A, X, & Y ON CURRENT STACK
*
                PHA
                TXA
                PHA
                TYA
                PHA
*
*  CHECK FOR STACK OVERFLOW
*
                TSX
                CPX        #$FA
                BCC        NMI005
                LDA        #>STKOVFL
                JSR        SYSDEATH
*
*  SET UP INTERRUPT ENVIRONMENT:
*    BINARY ARITHMETIC, 2 MHZ, I/O ENABLED,
*    RAM WRITE ENABLED, PRIMARY STACK,
*    AND $F000 RAM SELECTED.  PRESERVE
*    USER STATE OF SCREEN AND RESET LOCK.
*
NMI005          CLD
                LDA        E.REG
                TAX
                AND        #BITON5+BITON4
                ORA        #BITON6+BITON2
                STA        E.REG
```

```
*
*  IF NOT ALREADY ON PRIMARY STACK, SAVE USER'S
*  STACK POINTER AND SET UP SOS STACK POINTER.
*
                TXA
                AND        #BITON2
                BNE        NMI010
                TXA
                TSX
                STX        SP.SAVE
                LDX        #>E.SAVE
                TXS
                TAX
*
*  SAVE SYSTEM CONTROL REGISTERS E, Z, & B ON SOS STACK
*
NMI010          TXA
                PHA
                LDA        Z.REG
                PHA
                LDA        B.REG
                PHA
                LDA        EXPNSLOT
                PHA
                BIT        $CFFF
                BIT        $C020                 ;RESET I/O SPACE
                LDA        #$00
                STA        EXPNSLOT
*
*  SET INTERRUPT ZERO PAGE
*
                LDA        #0
                STA        Z.REG
*
*  SEE IF NMI IS FROM KEYBOARD OR I/O DEVICE
*
                LDA        E.IORB
                BMI        NMI030
*
*  NMI IS FROM I/O DEVICE
*
                LDA        SIRTABLE              ;NMI ALLOCATED?
                BPL        NMI020
                JSR        CALLNMI
                SEI
                JMP        DISPATCH
CALLNMI         LDA        SIRADR.L
                STA        NMIADR.L
                LDA        SIRADR.B
                STA        B.REG
                JMP        (NMIADR.L)
*
*  BAD INTERRUPT -- SYSTEM DEATH
*
NMI020          LDA        #>BADINT1            ;NMI NOT ALLOCATED
                JSR        SYSDEATH
*
*  NMI IS FROM THE KEYBOARD
*
NMI030          LDA        SYSBANK
                STA        B.REG
                JSR        KYBDNMI
                SEI
                JMP        DISPATCH
                SBTL       "DISPATCHER"
                REP        60
*
*  THIS IS THE DISPATCHER.  UPON COMPLETION, ALL SOS CALLS
*  AND INTERRUPT HANDLERS RETURN CONTROL TO THE DISPATCHER.
*  ITS PURPOSE IS TO SET UP THE APPROPRIATE ENVIRONMENT AND
*  PASS CONTROL TO WHATEVER CODE SHOULD RUN NEXT.
*
*  WHEN SOS IS INTERRUPTED, CONTROL ALWAYS RETURNS TO THE
*  INTERRUPTED CODE.  HOWEVER, WHEN THE USER IS INTERRUPTED,
*  BY EITHER A SOS CALL OR AN INTERRUPT, THE DISPATCHER
*  MUST CHECK THE EVENT QUEUE.  IF THERE IS AN ACTIVE EVENT
*  WITH A PRIORITY HIGHER THAN THE CURRENT EVENT FENCE,
*  CONTROL IS PASSED TO THE EVENT CODE.  OTHERWISE, CONTROL
*  RETURNS TO THE INTERRUPTED CODE.
*
                REP        60
*
DISPATCH        EQU        *
*
*  DISABLE INTERRUPTS AND RESTORE
*  SYSTEM CONTROL REGISTERS B & Z
*
                SEI
                LDA        E.REG
                ORA        #BITON6              ;ENABLE I/O
                STA        E.REG
                PLA
                JSR        SELC800              ;RESTORE I/O SPACE
                PLA
                STA        B.REG
                PLA
                STA        Z.REG
*
*  CHECK SAVED ENVIRONMENT REGISTER
*  IF RETURNING TO PRIMARY STACK
*    THEN RESTORE E REG AND RELAUNCH SOS
*    ELSE RESET STACK POINTER & RESTORE E REG
*
                PLA
                ORA        #BITON5              ;SET SCREEN STATE TO
                BIT        SCRNMODE             ;  CURRENT SCREEN MODE
                BMI        DSP005
                AND        #BITOFF5
DSP005          TAY
                AND        #BITON2
                BEQ        DSP010
                STY        E.REG
                BNE        DSP030
DSP010          PLA
                TAX
                TXS
                STY        E.REG
*
*  CHECK FOR ACTIVE EVENT WITH PRIORITY > FENCE
*
DSP020          LDA        CEVPRI
                LDX        EVQ.LINK
                CMP        EVQ.PRI,X
                BCS        DSP030
*
*  PROCESS ACTIVE EVENT TRAP
*  SAVE E, Z, B, & CALLER'S PRIORITY ON STACK THEN CALL
*  EVENT.  UPON RETURN, RESTORE PRIORITY, B, Z, & E THEN
*  CHECK FOR MORE EVENTS.
*
                LDA        E.REG
                PHA
                LDA        Z.REG
                PHA
                LDA        B.REG
                PHA
                LDA        CEVPRI
                PHA
                JSR        DO.EVENT
                SEI
                PLA
                STA        CEVPRI
                PLA
                STA        B.REG
                PLA
                STA        Z.REG
                PLA
                ORA        #BITON5              ;SET SCREEN STATE TO
                BIT        SCRNMODE             ;  CURRENT SCREEN MODE
                BMI        DSP025
                AND        #BITOFF5
DSP025          STA        E.REG
                JMP        DSP020
*
*  RESTORE CPU REGISTERS Y, X, & A AND LAUNCH
*
DSP030          PLA
                TAY
                PLA
                TAX
                PLA
                RTI
                PAGE
                REP        60
*
*  THIS SUBROUTINE CALLS THE HIGHEST PRIORITY ACTIVE EVENT.
*  FIRST, IT DELINKS THE FIRST ENTRY ON THE ACTIVE LIST AND
*  LINKS IT TO THE FREE LIST.  THEN, IT SETS UP THE BANK,
*  ADDRESS, ID, & STATUS AND CALLS THE EVENT VIA AN RTI.
*
                REP        60
*
DO.EVENT        EQU        *
*
*  WRITE ENABLE RAM
*
                LDY        E.REG
                TYA
                AND        #BITOFF3
                STA        E.REG
*
*  DELINK ENTRY FROM ACTIVE LIST AND RELINK IT TO FREE LIST
*
                LDX        EVQ.LINK
                LDA        EVQ.LINK,X
                STA        EVQ.LINK
                LDA        EVQ.FREE
                STA        EVQ.LINK,X
                STX        EVQ.FREE
*
*  SET FENCE TO EVENT PRIORITY THEN RESTORE E REG
*
                LDA        EVQ.PRI,X
                STA        CEVPRI
                STY        E.REG
*
*  SET UP B, Z, E, ADDRESS, ID, & STATUS
*
                LDA        EVQ.BANK,X
                STA        B.REG
                LDA        EVQ.ADRH,X
                PHA
                LDA        EVQ.ADRL,X
                PHA
                LDY        EVQ.ID,X
                PHP
                PLA
                AND        #$82
                PHA
                TYA
                RTI

                CHN        IPL.SRC2

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC1
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.IPL.SRC2.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC2
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SYSTEM INTERNAL RESOURCES"
                REP        60
*
*  SYSTEM INTERNAL RESOURCE NUMBERS
*
*
*  SIR  RESOURCE
*
*   0   SOUND PORT / I/O NMI
*   1   ACIA
*   2   E.CA2 -- KEYBOARD
*   3   E.CA1 -- CLOCK
*   4   E.SR
*   5   E.CB2 -- VBL +
*   6   E.CB1 -- VBL -
*   7   E.T2
*   8   E.T1
*   9   D.CA2 -- CSP INPUT FLAG / INPUT SWITCH 1
*   A   D.CA1 -- ANY SLOT (RESERVED FOR SOS)
*   B   D.SR  -- CSP DATA REGISTER
*   C   D.CB2 -- CSP DATA I/O / ENSIO
*   D   D.CB1 -- CSP CLOCK / ENSEL / A/D SELECT / INPUT SW3
*   E   D.T2
*   F   D.T1
*  10   DISK STEPPER / GRAPHICS SCROLL / CHARACTER DOWNLOAD
*  11   SLOT 1
*  12   SLOT 2
*  13   SLOT 3
*  14   SLOT 4
*  15   (UNASSIGNED)
*  16   (UNASSIGNED)
*  17   (UNASSIGNED)
*
                REP        60
                SBTL       "RESOURCE ALLOCATION & DEALLOCATION"
                REP        60
*
*  RESOURCE ALLOCATION AND DEALLOCATION
*
*  SIRS ARE ALLOCATED AND DEALLOCATED BY THE SUBROUTINES
*  'ALLOCSIR' AND 'DEALCSIR'.  THE RESOURCE PARAMETERS ARE
*  PASSED IN A TABLE THAT CONTAINS ONE FIVE-BYTE ENTRY FOR
*  EACH SIR THAT IS TO BE ALLOCATED OR DEALLOCATED.  THE
*  TABLE ENTRY FORMAT IS SHOWN BELOW:
*
*           0       1       2       3       4
*       +-------+-------+-------+-------+-------+
*       | SIR # |  ID   | ADR.L | ADR.H | ADR.B |
*       +-------+-------+-------+-------+-------+
*
*  SIR # -- SYSTEM INTERNAL RESOURCE NUMBER
*  ID    -- IDENTIFICATION BYTE
*           SUPPLIED BY ALLOCSIR, CHECKED BY DEALCSIR
*  ADR   -- INTERRUPT ADDRESS (LOW, HIGH, BANK)
*           ZERO IF NO INTERRUPT HANDLER
*
*
*  ALLOCSIR -- ALLOCATE SYSTEM INTERNAL RESOURCES
*
*    PARAMETERS:
*      A:  NUMBER OF BYTES IN TABLE
*      X:  TABLE ADDRESS (LOW BYTE)
*      Y:  TABLE ADDRESS (HIGH BYTE)
*
*    NORMAL EXIT -- SIRS ALLOCATED
*      CARRY:  CLEAR
*      A, X, Y:  UNDEFINED
*
*    ERROR EXIT -- SIRS NOT ALLOCATED
*      CARRY:  SET
*      X:  SIR NUMBER
*      A, Y:  UNDEFINED
*
*
*  DEALCSIR -- DEALLOCATE SYSTEM INTERNAL RESOURCES
*
*    PARAMETERS:
*      A:  NUMBER OF BYTES IN TABLE
*      X:  TABLE ADDRESS (LOW BYTE)
*      Y:  TABLE ADDRESS (HIGH BYTE)
*
*    NORMAL EXIT -- SIRS DEALLOCATED
*      CARRY:  CLEAR
*      A, X, Y:  UNDEFINED
*
*    ERROR EXIT -- SIRS NOT DEALLOCATED
*      CARRY:  SET
*      X:  SIR NUMBER
*      A, Y:  UNDEFINED
*
```

```
                REP        60
                PAGE
*
IDBYTE          DFB        $81
*
ALLOCSIR        EQU        *
                CLC
                PHP
                SEI
                STA        SIRARGSIZ            ;SAVE TABLE SIZE
                LDA        E.REG
                STA        SIRTEMP
                ORA        #BITON2              ;FORCE PRIMARY STACK
                AND        #BITOFF3             ;  AND WRITE ENABLE
                STA        E.REG
                LDA        SIRTEMP
                PHA
                LDA        Z.REG
                PHA
                LDA        #$00
                STA        Z.REG                ;SET ZERO PAGE := $00
                STX        SIRARGS
                STY        SIRARGS+1            ;SET POINTER TO TABLE
*
                LDY        #$00
ASIR010         LDA        (SIRARGS),Y          ;GET SIR NUMBER
                CMP        #SIRTBLSIZ
                TAX
                BCS        ASIR020
                LDA        SIRTABLE,X           ;CHECK ALLOCATION
                BMI        ASIR020
                LDA        IDBYTE
                STA        SIRTABLE,X           ;ALLOCATE SIR
                INY
                STA        (SIRARGS),Y          ;RETURN ID BYTE
                INY
                LDA        (SIRARGS),Y
                STA        SIRADR.L,X           ;SAVE INTERRUPT ADDRESS
                INY
                LDA        (SIRARGS),Y
                STA        SIRADR.H,X
                INY
                LDA        (SIRARGS),Y
                STA        SIRADR.B,X
                INY
                CPY        SIRARGSIZ
                BCC        ASIR010
*
                CLC
                INC        IDBYTE               ;BUMP ID BYTE
                BMI        SIREXIT
                LDA        #$81
                STA        IDBYTE
                BMI        SIREXIT
*
ASIR020         STX        SIRTEMP              ;SAVE BAD SIR NUMBER
ASIR030         SEC
                TYA
                SBC        #5
                TAY
                BCC        ASIR040
                LDA        (SIRARGS),Y          ;GET SIR NUMBER
                TAX
                LDA        #FALSE
                STA        SIRTABLE,X           ;RELEASE ALLOCATED SIRS
                BEQ        ASIR030
*
ASIR040         LDX        SIRTEMP              ;RETURN BAD SIR
                SEC
*
*
*
SIREXIT         PLA
                STA        Z.REG                ;RESTORE Z REGISTER
                PLA
                STA        E.REG                ;RESTORE E REGISTER
                BCC        SIREXIT1
                PLA
                ORA        #BITON0
                PHA
SIREXIT1        PLP
                RTS
*
*
*
DEALCSIR        EQU        *
                CLC
                PHP
                SEI
                STA        SIRARGSIZ            ;SAVE TABLE SIZE
                LDA        E.REG
                STA        SIRTEMP
                ORA        #BITON2              ;FORCE PRIMARY STACK
                AND        #BITOFF3             ;  AND WRITE ENABLE
                STA        E.REG
                LDA        SIRTEMP
                PHA
                LDA        Z.REG
                PHA
                LDA        #$00
                STA        Z.REG                ;SET ZERO PAGE := $00
                STX        SIRARGS
                STY        SIRARGS+1            ;SET POINTER TO TABLE
*
                LDY        #$00
DSIR010         LDA        (SIRARGS),Y          ;GET SIR NUMBER
                TAX
                CPX        #SIRTBLSIZ
                BCS        DSIR030
                INY
                LDA        SIRTABLE,X
                BPL        DSIR030              ;VERIFY ALLOCATION
                CMP        (SIRARGS),Y
                BNE        DSIR030
                INY
                INY
                INY
                INY
                CPY        SIRARGSIZ
                BCC        DSIR010
*
                LDY        SIRARGSIZ
DSIR020         SEC
                TYA
                SBC        #5
                TAY
                BCC        SIREXIT
                LDA        (SIRARGS),Y          ;GET SIR NUMBER
                TAX
                LDA        #FALSE
                STA        SIRTABLE,X
                BEQ        DSIR020
*
DSIR030         SEC
                BCS        SIREXIT
                SBTL       "SELECT I/O EXPANSION ROM"
                REP        60
*
*  SUBROUTINE 'SELC800' IS CALLED TO SELECT THE C800 I/O EX-
*  PANSION ADDRESS SPACE FOR A PERIPHERAL SLOT.  ON ENTRY,
*  THE SLOT NUMBER IS PASSED IN THE ACCUMULATOR.  IF NO
*  ERROR OCCURS, CARRY IS CLEARED; OTHERWISE, CARRY IS SET
*  AND THE PREVIOUS SLOT REMAINS SELECTED.
*
*  PARAMETERS:
*    A:  SLOT NUMBER
*
*  NORMAL EXIT -- NEW SLOT SELECTED
*    CARRY:  CLEAR
*    A:  UNDEFINED
*    X, Y:  UNCHANGED
*
*  ERROR EXIT -- SLOT NOT CHANGED
*    CARRY:  SET
*    A, X, Y:  UNCHANGED
*
*  WARNING !!!
*    'SELC800' USES SELF-MODIFYING CODE!
*
                REP        60
*
SELC800         EQU        *
                CMP        #$05                 ;CHECK SLOT NUMBER
                BCS        SC8EXIT              ;  INVALID
                PHP
                SEI
                STA        EXPNSLOT
                ORA        #$C0                 ;MAKE SLOT INTO $CN00
                STA        CNADDR+2             ;  AND MODIFY BIT ADDRESS
                BIT        $C020
                BIT        $CFFF                ;DESELECT PREVIOUS SLOT
CNADDR          BIT        $C0FF                ;  AND SELECT CURRENT SLOT
                PLP
SC8EXIT         RTS
                SBTL       "NMI DISABLE / ENABLE"
                REP        60
*
*  THE SUBROUTINES NMIDSBL AND NMIENBL ARE CALLED TO
*  DISABLE AND ENABLE NMI, RESPECTIVELY.  THERE ARE NO
*  INPUT PARAMETERS.  ON EXIT, THE REGISTERS ARE UN-
*  DEFINED.  NMIDSBL CLEARS THE CARRY FLAG IF NMI WAS
*  SUCCESSFULLY DISABLED; OTHERWISE, CARRY IS SET.
*
                REP        60
*
NMIDSBL         EQU        *
                LDX        E.REG
                BIT        NMIFLAG
                BPL        NDS020
                TXA
                ORA        #BITON7
                STA        E.REG                ;SET 1MHZ
                LDA        #$00
                STA        NMICNTR
                STA        NMICNTR+1
NDS010          BIT        NMIFLAG              ;NMI PENDING?
                BPL        NDS020               ;  NO
                INC        NMICNTR              ;BUMP NMI COUNTER
                BNE        NDS010               ;  AND RECHECK NMI FLAG
                INC        NMICNTR+1
                BNE        NDS010
                LDA        #>NMIHANG            ;CAN'T LOCK NMI
                JSR        SYSDEATH
NDS020          TXA                             ;GET E.REG
                AND        #BITOFF4             ;DISABLE NMI
                STA        E.REG
                RTS
*
*
*
NMIENBL         EQU        *
                LDA        E.REG
                ORA        #BITON4              ;ENABLE NMI
                STA        E.REG
                RTS
                SBTL       "KEYBOARD NMI HANDLER"
                REP        60
*
*  BY DEFAULT, KEYBOARD NMI IS IGNORED.  THE USER MAY
*  PROCESS NMI BY CHANGING THE ADDRESS IN SYSTEM GLOBAL.
*
                REP        60
*
NMIDBUG         EQU        *
                TSX                             ;SAVE THE STACK POINTER
                STX        NMISPSV
                LDA        #$03                 ;SELECT MONITOR'S ZERO PAGE
                STA        Z.REG
                LDA        E.REG
                ORA        #$03                 ;SELECT MONITOR ROM
                STA        E.REG
                JSR        $F901                ;CALL THE MONITOR
*
NMICONT         EQU        *
                LDA        E.REG
                ORA        #BITON2              ;FORCE PRIMARY STACK
                STA        E.REG
                LDX        NMISPSV
                TXS                             ;RESTORE STACK POINTER
                RTS
                SBTL       "EVENT QUEUE MANAGER"
                REP        60
*
*  THE EVENT QUEUE IS USED TO HOLD THE PARAMETERS OF EVENTS
*  THAT HAVE BEEN DETECTED BUT NOT YET RECOGNIZED.  EVENT
*  QUEUE ENTRIES ARE ORGANIZED INTO TWO LINKED LISTS; A FREE
*  LIST AND AN ACTIVE LIST.  EACH ENTRY IS SIX BYTES LONG,
*  WITH THE FIRST BYTE (BYTE 0) USED AS A LINK.  THE LINK
*  BYTE CONTAINS THE TABLE INDEX OF THE NEXT ENTRY IN THE
*  LIST.  BECAUSE OF THE INDEXING METHOD, THE EVENT QUEUE
*  MUST NOT EXCEED 256 BYTES.
*
*  ENTRY ZERO IS A SPECIAL ENTRY.  BYTE 0 IS THE INDEX OF
*  THE FIRST ACTIVE ENTRY; BYTE 1 CONTAINS A ZERO, ALLOWING
*  ENTRY 0 TO BE USED AS THE ACTIVE EVENT LIST TERMINATER;
*  BYTE 2 CONTAINS THE INDEX OF THE FIRST FREE ENTRY; AND
*  BYTES 4 THROUGH 6 ARE UNUSED.
*
*  THE FREE LIST IS LINKED LIFO.  THE ONLY VALID BYTE IN A
*  FREE ENTRY IS THE LINK BYTE; THE REMAINING BYTES ARE
*  UNDEFINED.  THE FREE LIST IS TERMINATED BY A LINK BYTE
*  CONTAINING A ZERO.
*
*  THE ACTIVE LIST IS LINKED IN DECREASING PRIORITY ORDER
*  WITH ENTRIES OF EQUAL PRIORITY LINKED FIFO.  BYTES 1
*  THROUGH 5 CONTAIN THE EVENT PRIORITY, EVENT ID, LOW BYTE
*  OF THE EVENT ADDRESS, HIGH BYTE OF THE EVENT ADDRESS, AND
*  THE ADDRESS BANK.  THE ACTIVE LIST IS TERMINATED BY AN
*  ENTRY WITH AN EVENT PRIORITY OF ZERO.
*
                REP        60
                PAGE
                REP        60
*
*  SUBROUTINE 'QUEEVENT' IS USED TO ENTER AN EVENT INTO THE
*  EVENT QUEUE.  ACTIVE EVENTS ARE LINKED IN DECREASING
*  PRIORITY ORDER WITH EVENTS OF EQUAL PRIORITY LINKED FIFO.
*  EVENTS ARE REMOVED FROM THE QUEUE AS THEY ARE RECOGNIZED
*  BY THE DISPATCHER.
*
*  PARAMETERS:
*    X:  EVENT PARAMETER ADDRESS (LOW BYTE)
*    Y:  EVENT PARAMETER ADDRESS (HIGH BYTE)
*
*    EVENT       0       1       2       3       4
*    PARMS:  +-------+-------+-------+-------+-------+
*            |  PRI  |   ID  | ADR.L | ADR.H | ADR.B |
*            +-------+-------+-------+-------+-------+
*            PRI:  EVENT PRIORITY
*            ID:   EVENT ID BYTE
*            ADR:  EVENT ADDRESS (LOW, HIGH, BANK)
*
*  EXIT CONDITIONS:
*    CARRY:  CLEAR
*    A, X, Y:  UNDEFINED
*
                REP        60
*
QUEEVENT        EQU        *
                CLC
                PHP
                SEI
                LDA        E.REG
                STA        QEVTEMP
                ORA        #BITON2              ;FORCE PRIMARY STACK
                AND        #BITOFF3             ;  AND WRITE ENABLE
                STA        E.REG
                LDA        QEVTEMP
                PHA
                LDA        Z.REG
                PHA
                LDA        #0
                STA        Z.REG                ;SET ZERO PAGE := 0
*
                STX        QEVARGS
                STY        QEVARGS+1            ;SET ARGUMENT POINTER
                LDY        #0
                LDA        (QEVARGS),Y          ;GET PRIORITY
                BEQ        Q.EXIT               ;  IGNORE IF ZERO
*
                LDX        EVQ.FREE
                BEQ        Q.FULL
                STX        QEV.THIS             ;GET FIRST FREE ENTRY
                LDA        EVQ.LINK,X           ;  AND DELINK IT
                STA        EVQ.FREE
*
                LDY        #EVQ.SIZ-2
QEV010          LDA        (QEVARGS),Y          ;COPY ARGUMENTS
                STA        EVQ.BANK,X           ;  INTO NEW ENTRY
                DEX
                DEY
                BPL        QEV010
*
                LDX        QEV.THIS
                LDY        #0
QEV020          STY        QEV.LAST
                LDA        EVQ.LINK,Y
                TAY
                LDA        EVQ.PRI,Y            ;SCAN EVENT QUEUE
                CMP        EVQ.PRI,X            ;  FOR PROPER POSITION
                BCS        QEV020
*
                TYA
                STA        EVQ.LINK,X           ;RELINK EVENT INTO QUEUE
                TXA
                LDY        QEV.LAST
                STA        EVQ.LINK,Y
*
Q.EXIT          PLA
                STA        Z.REG                ;RESTORE Z REGISTER
                PLA
                STA        E.REG                ;RESTORE E REGISTER
                PLP
                RTS
*
Q.FULL          LDA        #>EVQOVFL            ;EVENT QUEUE OVERFLOW
                JSR        SYSDEATH
                LST        ON

ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENIPL
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR IPL"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC2
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.OPRMSG.SRC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: OPRMSG.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1  OPERATOR MESSAGE/REPLY"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGOMSG
ZZORG           EQU        *
                MSB        OFF
                REP        60
*
*          COPYRIGHT (C) APPLE COMPUTER INC. 1981
*                   ALL RIGHTS RESERVED
*
                REP        60
*
*  THIS MODULE CONTAINS THE BLOCK FILE MANAGERS'S OPERATOR
*  INTERFACE.  IT DISPLAYS A MESSAGE IN A FOUR LINE BY
*  FOURTY COLUMN WINDOW, THEN WAITS FOR THE USER TO TOGGLE
*  THE ALPHA-LOCK KEY BEFORE RETURNING.
*
*  THE VERTICAL BLANKING FLAGS AND COMPOSITE BLANKING
*  TIMER ARE USED TO MAINTAIN THE DISPLAY.  MEMORY PAGE
*  $02 IS USED FOR TEMPORARY STORAGE.  ON EXIT, ALL
*  RESOURCES ARE RESTORED TO THEIR PREVIOUS STATES.
*
*  ENTRY POINT:  OPMSGRPLY
*
*  PARAMETERS:  X -- MESSAGE ADDRESS (LOW BYTE)
*               Y -- MESSAGE ADDRESS (HIGH BYTE)
*     (THE MESSAGE MUST RESIDE IN THE CURRENT BANK)
*
*  RESULT:  A -- RESPONSE KEYSTROKE
*           X, Y -- UNDEFINED
*
                REP        60
*
*
                ENTRY      OPMSGRPLY
*
                EXTRN      SCRNMODE
                PAGE
*
```

```
*  HARDWARE EQUATES
*
Z.REG           EQU        $FFD0
E.REG           EQU        $FFDF
*
KBPORT          EQU        $C008
*
BELL            EQU        $C040
*
VM0             EQU        $C050
VM1             EQU        $C052
VM2             EQU        $C054
VM3             EQU        $C056
*
E.T2            EQU        $FFE8
E.ACR           EQU        $FFEB
E.PCR           EQU        $FFEC
E.IFR           EQU        $FFED
E.IER           EQU        $FFEE
*
*  ZERO PAGE DECLARATIONS
*
                DSECT
ZPBASE          EQU        $200
                ORG        $0000                 ;ZERO PAGE DECLARATIONS
MSGPTR          DS         2                     ;MESSAGE POINTER
MSGIDX          DS         1
*
SCRNIDX         DS         1
SCRNPTR         DS         2
DATAPTR         DS         2
DATABUF         DS         160
*
SV.ZREG         DS         1
SV.EREG         DS         1
SV.SMODE        DS         1
SV.EACR         DS         1
SV.EPCR         DS         1
SV.EIER         DS         1
*
FLAG            DS         1
                DEND
                PAGE
OPMSGRPLY       EQU        *
*
*
*  SAVE CURRENT VALUES AND SET UP ZERO PAGE,
*  ENVIRONMENT, SCREEN MODE, AND E.6522 REGISTERS.
*
                PHP
                SEI
                LDA     Z.REG
                STA     ZPBASE+SV.ZREG      ;SAVE ZERO PAGE
                LDA     #<ZPBASE
                STA     Z.REG
                STX     MSGPTR              ;SAVE MESSAGE ADDRESS
                STY     MSGPTR+1
                LDA     E.REG
                STA     SV.EREG             ;SAVE ENVIRONMENT
                AND     #$5F
                ORA     #$40
                STA     E.REG               ;SCREEN OFF, I/O SPACE ON
                LDA     SCRNMODE
                STA     SV.SMODE            ;SAVE SCREEN MODE
                LDA     #$00
                STA     SCRNMODE
                BIT     VM0                 ;SET 40 COLUMN
                BIT     VM1                 ;  BLACK & WHITE TEXT
                BIT     VM2
                BIT     VM3
                LDX     E.ACR
                TXA
                AND     #$20
                STA     SV.EACR             ;SAVE AUXILIARY CONTROL REG
                TXA
                ORA     #$20
                STA     E.ACR               ;SET UP BL TIMER
                LDX     E.PCR
                TXA
                AND     #$F0
                STA     SV.EPCR             ;SAVE PERIPHERAL CONTROL REG
                TXA
                AND     #$0F
                ORA     #$60
                STA     E.PCR               ;SET UP VBL FLAGS
                LDA     E.IER
                AND     #$38
                STA     E.IER               ;MASK VBL & BL INTERRUPTS
                STA     SV.EIER             ;SAVE INTERRUPT MASKS
                PLP
*
*
*  SAVE SCREEN DATA AND CLEAR MESSAGE WINDOW
*
                LDX     #3
OPR010          JSR     SETPTRS
                LDY     #39
OPR020          LDA     (SCRNPTR),Y         ;SAVE SCREEN DATA
                STA     (DATAPTR),Y
                LDA     #$A0
                STA        (SCRNPTR),Y          ;BLANK SCREEN
                DEY
                BPL        OPR020
                DEX
                BPL        OPR010
*
*
*  MOVE MESSAGE TO WINDOW
*
                BIT        BELL
                LDX        #$00
                STX        MSGIDX
OPR100          JSR        SETPTRS
                LDY        #$00
                STY        SCRNIDX
OPR110          LDY        MSGIDX
                INC        MSGIDX
                LDA        (MSGPTR),Y           ;SET UP MESSAGE
                BEQ        OPR110
                BMI        OPR200
                CMP        #$0D
                BEQ        OPR120
                LDY        SCRNIDX
                INC        SCRNIDX
                ORA        #$80
                STA        (SCRNPTR),Y
                CPY        #39
                BCC        OPR110
OPR120          INX
                CPX        #4
                BCC        OPR100
*
*
*  DISPLAY MESSAGE UNTIL ALPHA-LOCK KEY TOGGLES
*
OPR200          LDY        #2
                LDA        KBPORT
                AND        #$08
                STA        FLAG
OPR210          JSR        VIDEO
                LDA        KBPORT
                AND        #$08
                CMP        FLAG
                BEQ        OPR210
                STA        FLAG
                DEY
                BNE        OPR210
*
*
*  RESTORE PREVIOUS CONTENTS OF WINDOW
```

```
*
                LDX        #3
OPR400          JSR        SETPTRS
                LDY        #39
OPR410          LDA        (DATAPTR),Y
                STA        (SCRNPTR),Y
                DEY
                BPL        OPR410
                DEX
                BPL        OPR400
*
*
*  RESTORE E.6522, SCREEN MODE, ENVIRONMENT, & ZERO PAGE
*  THEN RETURN TO CALLER
*
                PHP
                SEI
                LDA        E.ACR
                AND        #$DF
                ORA        SV.EACR              ;RESTORE AUXILIARY CONTROL REG
                STA        E.ACR
                LDA        E.PCR
                AND        #$0F
                ORA        SV.EPCR              ;RESTORE PERIPHERAL CONTROL REG
                STA        E.PCR
                LDA        SV.EIER              ;RESTORE INTERRUPT ENABLE REG
                ORA        #$80
                STA        E.IER
                LDA        SV.SMODE             ;RESTORE SCREEN MODE
                STA        SCRNMODE
                LSR        A
                BCC        OPR500
                BIT        VM0+1                ;RESTORE VIDEO MODE
OPR500          LSR        A
                BCC        OPR510
                BIT        VM1+1
OPR510          LSR        A
                BCC        OPR520
                BIT        VM2+1
OPR520          BIT        SCRNMODE
                BVC        OPR530
                BIT        VM3+1
OPR530          LDA        SV.EREG              ;RESTORE ENVIRONMENT
                STA        E.REG
                LDA        SV.ZREG              ;RESTORE ZERO PAGE
                STA        Z.REG
                PLP
                RTS
                PAGE
                REP        60
```

```
*
*  SUBROUTINE VIDEO
*
*  THIS SUBROUTINE POLLS THE VERTICAL-BLANKING AND
*  COMPOSITE-BLANKING-TIMER FLAGS AND TURNS THE SCREEN
*  OFF AND ON SO THAT ONLY THE MESSAGE WINDOW WILL BE
*  DISPLAYED.
*
*  THE E.6522 MUST BE INITIALIZED SO THAT E.CB2 FLAGS THE
*  POSITIVE EDGE OF VBL AND E.T2 COUNTS BL PULSES.  THE
*  INTERRUPTS MUST BE MASKED AND THE PROPER COUNT MUST
*  ALREADY BE STORED IN THE LOW ORDER BYTE OF E.T2.
*
*  ENTRY:  VIDEO
*
*  PARAMETERS:  INTERRUPT SYSTEM DISABLED
*
*  EXIT:  A -- UNDEFINED
*         X, Y -- PRESERVED
*
                REP        60
*
VIDEO           EQU        *
                LDA        E.IFR
                AND        #$28                 ;GET VBL & BL FLAGS
                BEQ        VID030
                STA        E.IFR                ;CLEAR FLAGS
                AND        #$20                 ;WHICH FLAG?
                BNE        VID010               ;  BL
*
                LDA        #$1F
                STA        E.T2                 ;SET UP BL TIMER
                LDA        #$00
                STA        E.T2+1
                LDA        E.REG
                ORA        #$20                 ;SET UP FOR SCREEN ON
                SEC
                BCS        VID020
*
VID010          LDA        E.REG
                AND        #$DF                 ;SET UP FOR SCREEN OFF
                CLC
*
VID020          STA        E.REG
                LDA        #$00
                ROR        A
                STA        SCRNMODE
VID030          RTS
                PAGE
                REP        60
```

```
*
*  SUBROUTINE SETPTRS
*
*  THIS SUBROUTINE SETS UP THE POINTERS TO THE MESSAGE
*  WINDOW AND DATA SAVE AREA.
*
*  ENTRY:  SETPTRS
*
*  PARAMETERS:  X -- LINE NUMBER [0..3]
*
*  EXIT:  A -- UNDEFINED
*         X, Y -- PRESERVED
*
                REP        60
*
SETPTRS         EQU        *
                TXA
                LSR        A
                ORA        #$04
                STA        SCRNPTR+1
                LDA        #$00
                ROR        A
                STA        SCRNPTR
                LDA        #<DATABUF
                STA        DATAPTR+1
                LDA        DBUFADR,X
                STA        DATAPTR
                RTS
*
DBUFADR         EQU        *
                DFB        >0*40+DATABUF
                DFB        >1*40+DATABUF
                DFB        >2*40+DATABUF
                DFB        >3*40+DATABUF
                LST        ON

ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENOMSG
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR OPRMSG"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: OPRMSG.SRC
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.A.SRC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.A.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        110
*
*   $1E00 +---------------+
*         !    SOSLDR     !<-ENTRY        SOS MEMORY MAP
*   $1FFF +---------------+  -----       (128K APPLE ///)
*
*              BANK 0               BANK 1               BANK 2
*   $2000 +---------------+    +---------------+    +---------------+
*         !               !    !               !    !               !
*         !               !    !               !    !    SOSLDR     !
*         !               !    !               !    !      &        !
*         !               !    !               !    !  INIT MODULE  !
*         !               !    !               !    !               !
*         !               !    !               !    ! - - - - - - - !
*         !               !    !               !    !    GLOBALS    !
*         !               !    !               !    ! - - - - - - - !
*         !               !    !               !    !               !
*         !               !    !               !    !               !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !  KERNEL !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !-- EOF --!
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*   $9FFF +---------------+    +---------------+    +---------------+
*
*
*   $A000 +---------------+
*     .   !    SOSBOOT    !
*     .   +---------------+
*
*
*  FIGURE 1.  SOS KERNEL FILE READ INTO $2:1E00..9FFF BY SOS BOOT IN BLOCKS 0,1.
*             SOS LOADER BEGINS EXECUTION AT THIS POINT.
*
*
*
*
                REP        110
                PAGE
                REP        110
*
*   $1E00 +---------------+
*         !    SOSLDR     !          SOS MEMORY MAP
*   $1FFF +---------------+          (128K APPLE ///)
*
*               BANK 0               BANK 1               BANK 2
*   $2000 +---------------+    +---------------+    +---------------+
*         !               !    !               !    !               !         !
*         !    SOSLDR     !    !               !    !               !
*         !      &        !    !               !    !               !
*         !  INIT MODULE  !    !               !    !               !
*         !               !    !               !    !               !         !
*  LDREND ! - - - - - - - !    !               !    !               !
*         !  FILE BUFFER  !    !               !    !               !
*         ! - - - - - - - !    !               !    !               !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !  INTERPRETER  !    !  INTERPRETER  !    !               !
*         !     FILE      !    !     FILE      !    !               !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !               !    !               !         !
*         !               !    !- - - EOF - - -!    !               !
*   $9FFF +---------------+    +---------------+    +---------------+
*
*
*
*
*  FIGURE 2.  SOS INTERPRETER FILE READ INTO BANKS 0 AND 1
*             USING EXTENDED ADDRESSING (X=$80).
*
*
*
*
*
                REP        110

                CHN        SOSLDR.B.SRC

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.A.SRC
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.C.SRC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.C.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        100
*
* SUBROUTINES:
*
* SOSLDR                "MAIN PROGRAM"
*
*    SOSLDR1            "PROCESSES KERNEL/INTERPRETER/DRIVER FILES"
*
* (1)   MOVE            "MOVES SRC.P..SRC.P+CNT-1 TO DST.P..DST.P+CNT-1"
*
*       INIT.KRNL       "CALLS KERNEL INITIALIZATION MODULES"
*
*       WELCOME         "PRINTS WELCOME MESSAGE ("APPLE ///", VERSION, DATE/TIME, COPYRIGHT)
*
*       ADVANCE         "ADVANCES WRK.PTR TO NEXT INTERP/KERNEL MODULE.  INITS SRC.P, DST.P, CNT FOR MOVE"
*
*       REVERSE         "REVERSES TITLE/CODE/RELOC COUNTS TO ALLOW DRIVER FILE TO BE PROCESSED FM BACK TO FRONT"
*
*       DADVANCE        "ADVANCES WORK.P TO NEXT DRIVER MODULE.  INITS SRC.P, CNT, REL.P FOR MOVE"
*
*          DADD         "ADVANCES WORK.P TO NEXT DRIVER FIELD"
*
*       FLAGS           "PROCESSES "INACTIVE" & "PAGE ALIGN" FLAGS IN DRIVER MODULE'S DIBS"
*
*          NEXT.DIB     "ADVANCES TO NEXT DIB IN DRIVER MODULE"
*
*       GETMEM          "COMPUTES DESTINATION BASE ADDRESS FOR NEXT DRIVER MODULE"
*
*          NEWDST       "COMPUTES DESTINATION BASE ADDRESS, ALIGNING ON PAGE BOUNDARY IF REQUESTED"
*
*          BUILD.DSEG   "COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT"
*
*       RELOC           "RELOCATES DRIVER MODULE'S CODE FIELD USING RELOCATION FIELD"
*
* (1)   LINK            "LINKS FIRST DIB TO PREVIOUS DRIVER'S LAST "ACTIVE" DIB, AND ADDS SDT ENTRY"
*
*          SET.DRIVES   "INITIALIZES DIB LINKS IN KERNEL'S FLOPPY DRIVER"
*
* (1)      ALLOC.DEV    "ADDS A NEW ENTRY TO THE DEVICE MANAGER'S SYSTEM DEVICE TABLE (SDT)"
```

```
*
*       ALLOC.SEG       "ALLOCATES SEGMENTS FOR KERNEL, INTERPRETER AND SYSTEM WORK AREA"
*
*          RSEG         "CALLS MEMORY MANAGER TO ALLOCATE SEGMENTS FOR THE KERNEL AND INTERPRTER"
*
*       ALLOC.DSEG      "ALLOCATES SEGMENTS FOR DRIVER MODULES"
*
*    ERROR              "DISPLAYS ERROR MESSAGE, SOUNDS BELL AND LOOPS UNTIL CONTROL/RESET PRESSED"
*
* (1) - INDICATES THAT THE ROUTINE PERFORMS BANK SWITCHING AND MUST(!) BE OUTSIDE THE 32K RAM BANKS.
                REP        100
                PAGE
                REP        100
*
* SOS.KERNEL FILE FORMAT
*
*  (8)   LABEL                    <---+
*          = "SOS KRNL"               !
*                                     !
*  (2)   HEADER COUNT                 !
*        HEADER                       !
*          = # OF FLOPPY DRIVES       !    CONTAINED IN THIS LISTING
*          = INTERPRETER PATHNAME     !
*          = DRIVER PATHNAME          !
*                                     !
*  (4)   ADR & COUNT                  !
*        SOSLDR CODE              <---+
*
*  (4)   ADR & COUNT
*        GLOBALS
*
*  (4)   ADR & COUNT
*        KERNEL CODE
*
                REP        100
*
* SOS.INTERP FILE FORMAT
*
*  (8)   LABEL
*           = "SOS NTRP"
*
*  (2)   HEADER COUNT
*
*  (4)   ADR & COUNT
*        INTERPRETER CODE
*
                REP        100
*
* SOS.DRIVER FILE FORMAT
*
```

```
*  (8)   LABEL
            = "SOS DRVR"
*
*  (2)   HEADER COUNT
            = # OF FLOPPY DRIVES
            = CHARACTER SET TABLE
            = KEYBOARD TABLE
         ...
*                                                               +---------------------------------------+
*  (2)   DM #N TITLE COUNT        <---+                         !       RELOCATION FIELD FORMAT         !
*              TITLE FIELD            !                         !       -----------------------         !
*  (2)   DM #N CODE  COUNT            !    DRIVER MODULE #N     ! CONSISTS OF A LIST OF 2 BYTE POINTERS !
*              CODE  FIELD            !                         ! WHICH POINT TO THE LOW BYTE OF A TWO  !
*  (2)   DM #N RELOC COUNT            !                         ! BYTE QUANTITY TO BE RELOCATED.        !
*              RELOC FIELD        <---+                         +---------------------------------------+
*        ...
*
*        $FFFF = THE END
*
                REP        100
                PAGE
                REP        100
*
* SOSLDR - EXTERNAL DECLARATIONS
*
                REP        100
                EXTRN      SYSBANK
                EXTRN      MEMSIZE
                EXTRN      SCRNMODE
                EXTRN      SOSVER
                EXTRN      SOSVERL
*
                EXTRN      INT.INIT             ; (IPL) INTERRUPT INIT
                EXTRN      EVQ.INIT             ; (IPL) EVENT QUEUE INIT
                EXTRN      DMGR.INIT            ; DEVICE MANAGER INIT
                EXTRN      MAX.DNUM             ;        "
                EXTRN      SDT.SIZE
                EXTRN      SDT.DIBL
                EXTRN      SDT.DIBH
                EXTRN      SDT.ADRL
                EXTRN      SDT.ADRH
                EXTRN      SDT.BANK
                EXTRN      SDT.UNIT
                EXTRN      BLKD.SIZE
                EXTRN      BLKDLST
                EXTRN      CFMGR.INIT           ; CHAR FILE MANAGER INIT
                EXTRN      MMGR.INIT            ; MEMORY MANAGER INIT
                EXTRN      BMGR.INIT            ; BUFFER FILE MANAGER INIT
                EXTRN      BFM.INIT             ; BLOCK FILE MANAGER INIT
                EXTRN      BFM.INIT2            ; BLOCK FILE MANAGER INIT2
                EXTRN      CLK.INIT             ; CLOCK SYSTEM CALL INIT
*
                EXTRN      DIB1                 ; ON BOARD DISK DRIVER'S DIBS (1-4)
                EXTRN      DIB2
                EXTRN      DIB3
                EXTRN      DIB4
*
*ENTRY I.BASE.P ; USED BY BFM.INIT2  (HARDWIRED!)
                PAGE
                REP        100
*
* FILE DATA DECLARATIONS
*
                REP        100
* KERNEL FILE
                REP        100
K.FILE          ASC        "SOS KRNL"
K.HDR.CNT       DW         LDR.ADR-K.DRIVES
K.DRIVES        DFB        $1
K.FLAGS         DFB        $0                   ; RESERVED FOR FUTURE USE
I.PATH          DFB        $E
                ASC        ".D1/SOS.INTERP"
                DS         $30-$F
D.PATH          DFB        $E
                ASC        ".D1/SOS.DRIVER"
                DS         $30-$F
LDR.ADR         DW         $0
LDR.CNT         DW         ZZEND-SOSLDR
                REP        100
* INTERPRETER/DRIVER FILES    <--+
* ERROR MESSAGES                 !    DEFINED IN BACK OF THIS LISTING
* WELCOME MESSAGES            <--+
                REP        100
                PAGE
                REP        100
*
* SOSLDR - DATA DECLARATIONS (1)
*
                REP        100
TRUE            EQU        $80
FALSE           EQU        $0
*
Z.REG           EQU        $FFD0
E.REG           EQU        $FFDF
B.REG           EQU        $FFEF
*
CZPAGE          EQU        $1A00
CSPAGE          EQU        $1B00
CXPAGE          EQU        $1600
SZPAGE          EQU        $1800
```

```
SXPAGE          EQU        $1400
SSPAGE          EQU        $0100
*
ROM.ADR         EQU        $F1B9
ROM.ID          EQU        $A0
                PAGE
                REP        100
*
* SOSLDR - DATA DECLARATIONS (2)
*
                REP        100
ZPAGE           EQU        $00
*
K.BASE          EQU        ZPAGE+$0             ; SOSLDR1 SUBROUTINE    +--------------------------------------+
I.BASE.P        EQU        ZPAGE+$2             ;                       ! <VARNAME>.P ::= 3 BYTE ZPAGE POINTER !
RDBUF.P         EQU        ZPAGE+$4             ;                       +--------------------------------------+
SYSBUF.P        EQU        ZPAGE+$6
TEMP.BANK       EQU        ZPAGE+$8
TEMP.ADRH       EQU        ZPAGE+$9
WORK.P          EQU        ZPAGE+$A
*
REV.SAVE        EQU        ZPAGE+$C             ; REVERSE SUBROUTINE
*
FIRST.ADIB      EQU        ZPAGE+$10            ; FLAGS SUBROUTINE
PREV.ADIB.P     EQU        ZPAGE+$12
DIB.P           EQU        ZPAGE+$14
PG.ALIGN        EQU        ZPAGE+$16
DIB.FLAGS       EQU        $14
DIB.DCB         EQU        $20
*
PREVBANK        EQU        ZPAGE+$18            ; GETMEM SUBROUTINE
PREVDST         EQU        ZPAGE+$19
*
CODE.P          EQU        ZPAGE+$1C            ; RELOCATION SUBROUTINE
REL.P           EQU        ZPAGE+$1E
REL.END         EQU        ZPAGE+$20
*
SRC.P           EQU        ZPAGE+$22            ; MOVE SUBROUTINE
DST.P           EQU        ZPAGE+$24
CNT             EQU        ZPAGE+$26
*
DSTBANK         EQU        ZPAGE+$2A            ; LINK SUBROUTINE
LINK.P          EQU        ZPAGE+$2C
*
DIB.ENTRY       EQU        2                    ; ALLOC.DEV SUBROUTINE
DIB.UNIT        EQU        4+16+2
DIB.DTYPE       EQU        4+16+3
*
ETEMP           EQU        ZPAGE+$2E            ; ERROR SUBROUTINE
*
```

```
WTEMP           EQU        ZPAGE+$2F            ; WELCOME SUBROUTINE

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.C.SRC
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.D.SRC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.D.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        100
*
* SOS LOADER -
*
* (MAIN PROGRAM)
                REP        100
SOSLDR          EQU        *                    ;                                          +---------------+
                LDA        #0                   ; ZERO SOS/USER X, Z AND STACK PAGES       ! SEE FIGURE 1. !
                TAX                             ;                                          +---------------+
SLDR010         STA        CZPAGE,X
                STA        CXPAGE,X
                STA        CSPAGE,X
                STA        SZPAGE,X
                STA        SXPAGE,X
                STA        SSPAGE,X
                DEX
                BNE        SLDR010
*                                       ; SETUP SOS CALL ENVIRONMENT (WRITE PROTECT=OFF)
                LDA        #$30                 ; E:=( 0.0.1.1:0.0.0.0 )
                STA        E.REG                ;    ( 1.I.S.R:W.P.R.R )
*
                LDX        #$FB                 ; CONSOLE 1.0 MODIFIES STACK DURING D.INIT CALL
                TXS
                LDA        #<CZPAGE             ; ZREG:=CALLER'S Z PAGE
                STA        Z.REG
*                                       ; +--------------------------------+
                JSR        SOSLDR1              ; ! PROCESS KRNL/INTERP/DRVR FILES !
*                                       ; +--------------------------------+
                LDA        E.REG
                AND        #$10                 ; SETUP SOS CALL ENVIRONMENT (WRITE PROTECT=ON)
                ORA        #$28                 ; E:=( 0.0.1.X:1.0.0.0 )
                STA        E.REG                ;    ( 1.I.S.R:W.P.R.R )
*
                LDX        #$FF                 ; STACK.REG:=$FF
                TXS
                LDA        #<CZPAGE             ; ZREG:=CALLER'S Z PAGE
                STA        Z.REG
*                                                                                  +---------------+
                LDA        SYSBANK              ; BREG:=SYSBANK                            ! SEE FIGURE 4. !
                STA        B.REG                ;                                          +---------------+
                JMP        (I.BASE.P)           ; SOS LOAD COMPLETE - JMP TO INTERPRETER
*
*THE END.
                REP        100
                PAGE
                REP        100
*
* MOVE ( IN:   SRC.P
*        IN:   DST.P
*        IN:   A="BANK"
*        IN:   CNT      )
*
*        LOCAL:  END
* (MOVES SRC.P..SRC.P+CNT-1 TO DST.P..DST.P+CNT-1)                "CNT PARM IS DESTROYED"
                REP        100
MOVE            EQU        *
                TAX
                LDA        B.REG                ; SAVE BANK REGISTER
                PHA
                STX        B.REG                ; BREG:=A
                LDA        CNT+1                ; IF CNT <> 0
                ORA        CNT                  ;    THEN
                BEQ        MOVE.EXIT
                LDA        CNT                  ;       CNT:=CNT-1
                BNE        MOVE010
                DEC        CNT+1
MOVE010         DEC        CNT
                CLC                             ;       SRC.P:=SRC.P+PAGE.CNT
                LDA        SRC.P+1
                ADC        CNT+1
                STA        SRC.P+1
                LDA        DST.P+1              ;       DST.P:=DST.P+PAGE.CNT
                ADC        CNT+1
                STA        DST.P+1
                INC        CNT+1                ;       PAGE.CNT:=PAGE.CNT+1
                LDY        CNT                  ;       Y:=BYTE.CNT
                BEQ        MOVE020              ;       IF Y=0 THEN M2
*
MOVE.PAGE       LDA        (SRC.P),Y            ;M1:    DO
                STA        (DST.P),Y            ;          (DST.P),Y:=(SRC.P),Y
                DEY                             ;          Y:=Y-1
                BNE        MOVE.PAGE            ;       UNTIL  Y=0
MOVE020         LDA        (SRC.P),Y            ;M2:    (DST.P),Y:=(SRC.P),Y
                STA        (DST.P),Y
                DEY                             ;       Y:=Y-1
                DEC        SRC.P+1              ;       SRC.P:=SRC.P-256
                DEC        DST.P+1              ;       DST.P:=DST.P-256
                DEC        CNT+1                ;       PAGE.CNT:=PAGE.CNT-1
                BNE        MOVE.PAGE            ;       IF PAGE.CNT <> 0 THEN M1
```

```
*
                INC        SRC.P+1              ; RESTORE SRC.P
                INC        DST.P+1              ;    "    DST.P
*
MOVE.EXIT       PLA                             ; RESTORE BANK REGISTER
                STA        B.REG
                RTS
                PAGE
                REP        100
*
* LINK ( IN:   DST.P
*        IN:   DSTBANK
*        IN:   PREVBANK
*        IN:   FIRST.ADIB
*        I/O:  SDT.TBL
*        I/O:  BLKDLST
*        OUT:  LINKED DRIVER MODULE )
*
*        OWN:  LINK.P
* (LINKS FIRST DIB TO PREVIOUS DRIVER'S LAST "ACTIVE" DIB, AND ADDS SDT ENTRY)
                REP        100
LINK            EQU        *
                CLC                             ; FIRST.ADIB:=0:DST.P+FIRST.ADIB
                LDA        DST.P
                ADC        FIRST.ADIB
                STA        FIRST.ADIB
                LDA        DST.P+1
                ADC        FIRST.ADIB+1
                STA        FIRST.ADIB+1
                LDA        #0
                STA        CXPAGE+FIRST.ADIB+1
                LDA        PREVBANK             ; BREG:=PREVBANK
                STA        B.REG
                LDY        #0                   ; (LINK.P):=FIRST.ADIB
                LDA        FIRST.ADIB
                STA        (LINK.P),Y
                INY
                LDA        FIRST.ADIB+1
                STA        (LINK.P),Y
                LDA        DSTBANK              ; BREG:=DSTBANK
                STA        B.REG
                LDA        FIRST.ADIB           ; LINK.P:=FIRST.ADIB
                STA        LINK.P
                LDA        FIRST.ADIB+1
                STA        LINK.P+1
WALKLINKS       JSR        ALLOC.DEV            ; ALLOC.DEV(LINK.P BREG.IN, SDT.TBL BLKDLST.IO)
LINK010         LDY        #0                   ; WHILE (LINK.P) <> 0 AND (LINK.P) <> LINK.P
                LDA        (LINK.P),Y
                INY
                ORA        (LINK.P),Y
                BEQ        LINK100
                LDA        (LINK.P),Y
                CMP        LINK.P+1
                BNE        LINK030
                DEY
                LDA        (LINK.P),Y
                CMP        LINK.P
                BEQ        LINK100
LINK030         LDY        #0                   ;    DO  LINK.P:=(LINK.P)
                LDA        (LINK.P),Y
                TAX
                INY
                LDA        (LINK.P),Y
                STX        LINK.P
                STA        LINK.P+1
                JSR        ALLOC.DEV            ;     "  ALLOC.DEV(LINK.P BREG.IN, SDT.TBL BLKDLST.IO)
                JMP        LINK010
*
LINK100         LDY        #0                   ; (LINK.P):=0
                TYA
                STA        (LINK.P),Y
                INY
                STA        (LINK.P),Y
                DEY                             ; BREG:=0
                STY        B.REG
                RTS
*
*
*
*
* LINK.INIT ( IN:   A=# DRIVES
*             IN:   DIB1..4
*             I/O:  SDT.TBL
*             I/O:  BLKDLST    )
*
LINK.INIT       EQU        *
                JSR        SET.DRIVES           ; SET.DRIVES(A=#DRIVES.IN, DIB1..4.IN)
                LDA        #0
                STA        MAX.DNUM             ; MAXDNUM:=0
                STA        BLKDLST              ; BLKDLST:=0
                STA        CXPAGE+LINK.P+1      ; LINK.P:=0:DIB1
                LDA        #>DIB1
                STA        LINK.P
                LDA        #<DIB1
                STA        LINK.P+1
                JMP        WALKLINKS
                PAGE
                REP        100
*
* ALLOC.DEV ( IN:   LINK.P
*             IN:   B.REG
*             I/O:  SDT.TBL                                                   (SYSTEM DEVICE TABLE)
*                      IN:   SDT.SIZE  = CONSTANT
*                      IN:   DIB.ENTRY = CONSTANT                       DEV   DIB   ADR  BANK  UNIT
*                      IN:   DIB.UNIT  = CONSTANT                           !-----!-----!-----!-----!
*                      IN:   DIB.DTYPE = CONSTANT                        1  !     !     !     !     !
*                      I/O:  MAX.DNUM                                    2  !     !     !     !     !
*                      OUT:  SDT.BANK                                    .  !     !     !     !     !
*                      OUT:  SDT.DIB                                     .  !     !     !     !     !
*                      OUT:  SDT.ADR                                     .  !-----!-----!-----!-----!
*                      OUT:  SDT.UNIT                                 MAX.DNUM
*             I/O:  BLKDLST
*                      IN:   BLKD.SIZE = CONSTANT
* (ADDS A NEW ENTRY TO THE DEVICE MANAGER'S SYSTEM DEVICE TABLE (SDT))
                REP        100
ALLOC.DEV       EQU        *
                INC        MAX.DNUM             ; MAX.DNUM:=MAX.DNUM+1
                LDX        MAX.DNUM             ; IF MAX.DNUM >= SDT.SIZE
                CPX        #>SDT.SIZE           ;    THEN
                BCC        ADEV010
                LDX        #ERR8X               ;       ERROR("TOO MANY DEVICES")
                LDY        #ERR8L
                JSR        ERROR
ADEV010         LDA        B.REG                ; SDT.BANK,X:=BREG
                STA        SDT.BANK,X
                CLC                             ; SDT.DIB,X:=LINK.P+4
                LDA        LINK.P
                ADC        #4
                STA        SDT.DIBL,X
                LDA        LINK.P+1
                ADC        #0
                STA        SDT.DIBH,X
                SEC                             ; SDT.ADR,X:=(LINK.P),DIB.ENTRY-1
                LDY        #DIB.ENTRY
                LDA        (LINK.P),Y
                SBC        #1
                STA        SDT.ADRL,X
                INY
                LDA        (LINK.P),Y
                SBC        #0
                STA        SDT.ADRH,X
                LDY        #DIB.UNIT            ; SDT.UNIT,X:=(LINK.P),DIB.UNIT
                LDA        (LINK.P),Y
                STA        SDT.UNIT,X
                LDY        #DIB.DTYPE           ; IF (LINK.P),DIB.DTYPE = "BLOCK DEVICE"
                LDA        (LINK.P),Y
                BPL        ADEV.EXIT
                TXA                             ;    THEN
                INC        BLKDLST              ;       BLKDLST:=BLKDLST+1
                LDX        BLKDLST              ;       IF BLKDLST >= BLKD.SIZE
                CPX        #>BLKD.SIZE          ;          THEN
                BCC        ADEV020
                LDX        #ERR9X               ;             ERROR("TOO MANY BLOCK DEVICES")
                LDY        #ERR9L
                JSR        ERROR
ADEV020         STA        BLKDLST,X            ;       BLKDLST,X:=MAX.DNUM
ADEV.EXIT       RTS                             ; RETURN
                PAGE
                REP        100
*
* SOSLDR1 ()
*
* (PROCESSES KERNEL/INTERPRETER/DRIVER FILES)
                REP        100
SOSLDR1         EQU        *
                LDX        #$1F                 ; COPY ROM'S DISK CORE ROUTINE ZPAGE VARS TO SOS ZPAGE
LDR010          LDA        $380,X
                STA        SZPAGE,X
                DEX
                BPL        LDR010
                REP        100
* PROCESS KERNEL FILE
                REP        100
*
* MOVE AND INITIALIZE SOS GLOBALS
*
                LDA        #>LDR.ADR            ; WORK.P:=0:LDR.ADR
                STA        WORK.P
                LDA        #<LDR.ADR
                STA        WORK.P+1
                JSR        ADVANCE              ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
*
                LDA        B.REG                ; MOVE(SRC.P DST.P A=BREG CNT.IN)
                JSR        MOVE
*
                LDA        B.REG                ; SYSBANK:=BREG
                AND        #$0F
                STA        SYSBANK
                ASL        A                    ; MEMSIZ:=SYSBANK*2+4 "16K CHUNKS"
                CLC
                ADC        #4
                STA        MEMSIZE              ; AND, MEMSIZE (SIZE IN 16K BYTE "CHUNKS")
*
* MOVE KERNAL CODE
*
                JSR        ADVANCE              ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
*
                LDA        DST.P                ; K.BASE:=DST.P
                STA        K.BASE
                LDA        DST.P+1
                STA        K.BASE+1
                LDA        B.REG                ; MOVE(SRC.P DST.P A=BREG CNT.IN)
                JSR        MOVE
*
* MOVE LOADER TO BANK 0 AND SWITCH FROM SYSTEM BANK TO BANK 0
*
                LDA        #>$2000              ; MOVE(SRC.P=0:2000 DST.P=8F:2000 A=BREG CNT=LDR.END-$2000)
                STA        SRC.P
                STA        DST.P
                LDA        #<$2000
                STA        SRC.P+1
                STA        DST.P+1
                LDA        #$8F
                STA        CXPAGE+DST.P+1
                LDA        #>LDREND-$2000
                STA        CNT
                LDA        #<LDREND-$2000
                STA        CNT+1
                LDA        B.REG
                JSR        MOVE
                LDA        #0                   ; BREG:=0
                STA        B.REG
*
* INITIALIZE SDT TABLE, KERNEL AND PRINT WELCOME MESSAGE
*
                LDA        K.DRIVES             ; LINK.INIT(A=K.DRIVES DIB1..4.IN, SDT.TBL BLKDLST.IO)
                JSR        LINK.INIT
                JSR        INIT.KRNL            ; INIT.KRNL()
                JSR        WELCOME              ; WELCOME()
*
                LDA        E.REG                ; ENABLE ROM BANK
                ORA        #$03
                STA        E.REG
                LDA        ROM.ADR              ; IF MONITOR ROM <> NEW
                CMP        #ROM.ID              ;    THEN
                BEQ        LDR020
                LDX        #ERR7X               ;       ERROR("ROM ERROR:  PLEASE NOTIFY YOUR DEALER")
                LDY        #ERR7L
                JSR        ERROR
LDR020          LDA        E.REG                ; DISABLE ROM BANK
                AND        #$F6
                STA        E.REG
                REP        100
* PROCESS INTERPRETER FILE
                REP        100
*
* OPEN SOS INTERPRETER FILE (DEFAULT='SOS.INTERP')
*
                LDY        I.PATH               ; OPEN(PATHNAME:=I.PATH
LDR030          LDA        I.PATH,Y             ;      REFNUM=OPEN.REF
```

```
                STA        PATH,Y               ;      SYSBUF.P:=80:LDREND-2000 )
                DEY
                BPL        LDR030
*
                LDA        #>LDREND-$2000
                STA        SYSBUF.P
                LDA        #<LDREND-$2000
                STA        SYSBUF.P+1
                LDA        #$80
                STA        CXPAGE+SYSBUF.P+1
*
*
                BRK
                DFB        OPEN
                DW         OPEN.PARMS
                BEQ        LDR040
                LDX        #ERR1X               ; ERROR("INTERPRETER FILE NOT FOUND")
                LDY        #ERR1L
                JSR        ERROR
LDR040          LDA        OPEN.REF
                STA        READ.REF
                STA        CLOSE.REF
*
* READ IN ENTIRE INTERPRETER FILE
*
                LDA        #$80                 ; READ(REFNUM=READ.REF
                STA        CXPAGE+RDBUF.P+1     ;      RDBUF.P:=80:FILE
                LDA        #>FILE               ;      BYTES=$FFFF-FILE+1
                STA        RDBUF.P              ;      BYTESRD=I.BYTESRD )
                LDA        #<FILE
                STA        RDBUF.P+1
*
                BRK
                DFB        READ
                DW         READ.PARMS
                BEQ        LDR050
                LDX        #ERR0X               ; ERROR("I/O ERROR")
                LDY        #ERR0L
                JSR        ERROR
*                                                                                  +---------------+
* CLOSE INTERPRETER FILE AND CHECK LABEL                                           ! SEE FIGURE 2. !
*                                                                                  +---------------+
LDR050          BRK                             ; CLOSE(REFNUM=CLOSE.REF)
                DFB        CLOSE
                DW         CLOSE.PARMS
                LDY        #7                   ; CHECK LABEL
LDR051          LDA        (RDBUF.P),Y
                CMP        I.LABEL,Y
                BNE        LDR052
                DEY
                BPL        LDR051
                BMI        LDR053
LDR052          LDX        #ERR2X               ; ERROR("INVALID INTERPRETER FILE")
                LDY        #ERR2L
                JSR        ERROR
*
* MOVE INTERPRETER CODE
*
LDR053          LDA        #>I.HDR.CNT-2        ; WORK.P:=80:I.HDR.CNT-2
                STA        WORK.P
                LDA        #<I.HDR.CNT-2
                STA        WORK.P+1
                LDA        #$80
                STA        CXPAGE+WORK.P+1
*
                JSR        ADVANCE              ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
*
                LDA        DST.P                ; I.BASE.P:=0:DST.P
                STA        I.BASE.P
                LDA        DST.P+1
                STA        I.BASE.P+1
                LDA        #0
                STA        CXPAGE+I.BASE.P+1
*
                CLC                             ; IF DST.P+CNT > K.BASE THEN ERROR
                LDA        CNT
                ADC        DST.P
                TAX
                LDA        CNT+1
                ADC        DST.P+1
                CPX        K.BASE
                SBC        K.BASE+1
                BEQ        LDR070
                BCC        LDR070
                LDX        #ERR3X               ; ERROR("INCOMPATIBLE INTERPRETER")
                LDY        #ERR3L
                JSR        ERROR
*
LDR070          LDA        SYSBANK              ; MOVE(SRC.P=RDBUF.P DST.P A=SYSBANK CNT.IN)
                JSR        MOVE
                REP        100
* PROCESS DRIVER FILE
                REP        100
*
* OPEN SOS DRIVER FILE (DEFAULT='SOS.DRIVER')
*
                LDY        D.PATH               ; OPEN(PATHNAME:=D.PATH
LDR080          LDA        D.PATH,Y             ;      REFNUM=OPEN.REF
                STA        PATH,Y               ;      SYSBUF.P:=80:LDREND-2000 )
                DEY
                BPL        LDR080
*
                BRK
                DFB        OPEN
                DW         OPEN.PARMS
                BEQ        LDR090
                LDX        #ERR4X               ; ERROR("DRIVER FILE NOT FOUND")
                LDY        #ERR4L
                JSR        ERROR
LDR090          LDA        OPEN.REF
                STA        READ.REF
                STA        CLOSE.REF
*
* READ IN ENTIRE DRIVER FILE INTO BANK 0
*
                BRK                             ; READ(REFNUM=READ.REF
                DFB        READ                 ;      RDBUF.P:=80:FILE
                DW         READ.PARMS           ;      BYTES=$FFFF-FILE+1
*                                       ;      BYTESRD=D.BYTESRD )
                BEQ        LDR100
                LDX        #ERR0X               ; ERROR("I/O ERROR")
                LDY        #ERR0L
                JSR        ERROR
*                                                                                  +---------------+
* CLOSE THE DRIVER FILE AND CHECK LABEL                                            ! SEE FIGURE 3. !
*                                                                                  +---------------+
LDR100          BRK                             ; CLOSE(REFNUM=CLOSE.REF)
                DFB        CLOSE
                DW         CLOSE.PARMS
                LDY        #$7                  ; CHECK LABEL
LDR101          LDA        (RDBUF.P),Y
                CMP        D.LABEL,Y
                BNE        LDR102
                DEY
                BPL        LDR101
                BMI        LDR103
LDR102          LDX        #ERR5X               ; ERROR("INVALID DRIVER FILE")
                LDY        #ERR5L
                JSR        ERROR
*
* MOVE CHARACTER SET TABLE
*
LDR103          LDA        #>D.CHRSET           ; MOVE(SRC.P=D.CHRSET DST.P=$C00 A=0 CNT=$400)
                STA        SRC.P
                LDA        #<D.CHRSET
                STA        SRC.P+1
                LDA        #>$C00
                STA        DST.P
                LDA        #<$C00
                STA        DST.P+1
                LDA        #>$400
                STA        CNT
                LDA        #<$400
                STA        CNT+1
                LDA        #0
                JSR        MOVE
*
* MOVE KEYBOARD TABLE
*
                LDA        #>D.KYBD             ; MOVE(SRC.P=D.KYBD DST.P=$1700 A=0 CNT=$100.IN)
                STA        SRC.P
                LDA        #<D.KYBD
                STA        SRC.P+1
                LDA        #>$1700
                STA        DST.P
                LDA        #<$1700
                STA        DST.P+1
                LDA        #>$100
                STA        CNT
                LDA        #<$100
                STA        CNT+1
                LDA        #0
                JSR        MOVE
*
* RE-INITIALIZE SDT TABLE
*
                LDY        #>D.DRIVES-D.FILE    ; LINK.INIT(A=D.DRIVES DIB1..4.IN, SDT.TBL BLKDLST.IO)
                LDA        (RDBUF.P),Y
                JSR        LINK.INIT
*
                LDA        #0                   ; DST.P:=0:I.BASE.P/256*256
                STA        CXPAGE+DST.P+1
                STA        DST.P
                LDA        I.BASE.P+1
                STA        DST.P+1
                CMP        #$A0                 ; IF DST.P>=$A000 THEN DST.P:=$A000
                BCC        LDR105
                LDA        #$A0
                STA        DST.P+1
LDR105          LDA        SYSBANK              ; DSTBANK:=SYSBANK
                STA        DSTBANK
                JSR        REVERSE              ; REVERSE(D.HDR.CNT.IN, WORK.P.OUT)
*
* RELOCATE AND MOVE DRIVERS
*
NEXTDRIVER      JSR        DADVANCE             ; "NO DRIVERS LEFT":=DADVANCE(WORK.P.IO SRC.P CNT REL.P.OUT)
                BCS        LDR140
                JSR        FLAGS                ; "INACTIVE":=FLAGS(SRC.P.IN, PG.ALIGN FIRST.ADIB.OUT)
                BVS        NEXTDRIVER
                JSR        GETMEM               ; GETMEM(PG.ALIGN CNT.IN, DST.P DSTBANK DSEGLIST.IO, PREVBANK.OUT)
                JSR        RELOC                ; RELOC(SRC.P REL.P DST.P.IN)
*
                LDA        DSTBANK              ; IF DSTBANK < 0 OR DST.P < SRC.P THEN ERROR
                BMI        LDR120
                LDA        CXPAGE+SRC.P+1       ;    (CONVERT SRC.P TO BANK SWITCHED ADDRESS)
                AND        #$7F
                STA        TEMP.BANK
                LDA        SRC.P+1
                BPL        LDR110
                INC        TEMP.BANK
LDR110          AND        #$7F
                CLC
                ADC        #<$2000
                STA        TEMP.ADRH
                LDA        DST.P                ;    (NOW COMPARE)
                CMP        SRC.P
                LDA        DST.P+1
                SBC        TEMP.ADRH
                LDA        DSTBANK
                SBC        TEMP.BANK
                BCS        LDR130
LDR120          LDX        #ERR6X               ;    ERROR("DRIVER FILE TOO LARGE")
                LDY        #ERR6L
                JSR        ERROR
*
LDR130          LDA        DSTBANK              ; MOVE(SRC.P DST.P A=DSTBANK CNT.IN)
                JSR        MOVE
                JSR        LINK                 ; LINK(DST.P DSTBANK PREVBANK FIRST.ADIB.IN, SDT.TBL BLKDLST.IO)
                JMP        NEXTDRIVER
                REP        100
* SETUP USER ENVIRONMENT
                REP        100
*
* RE-INITIALIZE KERNEL/DRIVERS, ALLOCATE SYSTEM SEGMENTS
*
LDR140          JSR        INIT.KRNL            ; INIT.KRNL()
                JSR        ALLOC.SEG            ; ALLOC.SEG(K.BASE I.BASE.P SYSBANK.IN)
                JSR        ALLOC.DSEG           ; ALLOC.DSEG(DSEGLIST.IN)
*
* SET PREFIX TO THE BOOT VOLUME
*
                LDA        #0                   ; TURN VIDEO OFF - PREVENTS CHAR "GROWTH" DURING DOWNLOAD
                STA        SCRNMODE
                BRK                             ; SET.PREFIX(PREFIXPATH=".D1")
                DFB        SETPREFIX
                DW         PREFX.PARMS
*
* LAUNCH CHARACTER SET DOWNLOAD (CONSOLE) AND CLEAR SCREEN
*
                CLI                             ; BEGIN CHARACTER SET DOWNLOAD (CONSOLE)
```

```
*
                LDA        #0                     ; CLEAR TEXT SCREENS
                STA        CXPAGE+SRC.P+1
                STA        CXPAGE+DST.P+1
                LDA        #$04
                STA        SRC.P+1
                STA        DST.P+1
                LDA        #$00
                STA        SRC.P
                LDA        #$80
                STA        DST.P
                LDA        #$A0
                LDX        #8
CLEAR0          LDY        #$77
CLEAR1          STA        (SRC.P),Y
                STA        (DST.P),Y
                DEY
                BPL        CLEAR1
                INC        SRC.P+1                ; NEXT PAGE
                INC        DST.P+1                ; NEXT PAGE
                DEX
                BNE        CLEAR0
*
WAIT            INC        SRC.P                  ; WAIT FOR DOWNLOAD TO COMPLETE
                BNE        WAIT
                INX
                BNE        WAIT
*
                LDA        #$80                   ; TURN VIDEO ON
                STA        SCRNMODE
                RTS
                REP        100

                CHN        SOSLDR.E.SRC

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.D.SRC
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.E.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.E.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        100
*
* SET.DRIVES ( IN:   A=# DRIVES
*              IN:   DIB1..4    )
* (INITIALIZES DIB LINKS IN KERNEL'S FLOPPY DRIVER)
                REP        100
*
SET.DRIVES      EQU        *
                TAY                                 ; SAVE # OF DRIVES
                LDA        #>DIB2                   ; DIB1:=ADR(DIB2)
                STA        DIB1
                LDA        #<DIB2
                STA        DIB1+1
                LDA        #>DIB3                   ; DIB2:=ADR(DIB3)
                STA        DIB2
                LDA        #<DIB3
                STA        DIB2+1
                LDA        #>DIB4                   ; DIB3:=ADR(DIB4)
                STA        DIB3
                LDA        #<DIB4
                STA        DIB3+1
*
                LDA        #0                       ; CASE (Y=# OF DRIVES)
                CPY        #2
                BCC        STDR010
                BEQ        STDR020
                CPY        #4
                BCC        STDR030
                BCS        STDR040
*
STDR010         STA        DIB1                     ;    1:  DIB1:=0
                STA        DIB1+1
                RTS
*
STDR020         STA        DIB2                     ;    2:  DIB2:=0
                STA        DIB2+1
                RTS
*
STDR030         STA        DIB3                     ;    3:  DIB3:=0
```

```
                STA        DIB3+1
                RTS
*
STDR040         STA        DIB4                 ;    4:  DIB4:=0
                STA        DIB4+1
                RTS                             ; RETURN
                PAGE
                REP        100
*
* INIT.KRNL ()
*
* (CALLS KERNEL INITIALIZATION MODULES)
                REP        100
*
INIT.KRNL       EQU        *
                LDA        E.REG                ; SWITCH IN I/O BANK AND SELECT PRIMARY STACK
                ORA        #$44                 ; E:=( 0.1.1.X:0.1.0.0 )
                STA        E.REG                ;    ( 1.I.S.R:W.P.R.R )
*
                LDA        #<SZPAGE             ; SWITCH TO SOS ZPAGE
                STA        Z.REG
*
                JSR        INT.INIT             ; CALL KERNEL INITIALIZATION ROUTINES
                JSR        EVQ.INIT
                JSR        BFM.INIT2
                BCS        INITK.ERR
                JSR        DMGR.INIT
                JSR        CFMGR.INIT
                JSR        MMGR.INIT
                JSR        BMGR.INIT
                JSR        BFM.INIT
                JSR        CLK.INIT
*
                LDA        E.REG                ; SWITCH OUT I/O BANK AND RETURN TO ALTERNATE STACK
                AND        #$BB                 ; E:=( 0.0.1.X:0.0.0.0 )
                STA        E.REG                ;    ( 1.I.S.R:W.P.R.R )
*
                LDA        #<CZPAGE             ; SWITCH BACK TO USER ZPAGE
                STA        Z.REG
*
                RTS                             ; RETURN
*
*
INITK.ERR       LDX        #ERR0X               ; ERROR("I/O ERROR")
                LDY        #ERR0L
                JMP        ERROR
                PAGE
                REP        100
*
* ADVANCE ( I/O:  WORK.P
*           OUT:  SRC.P
*           OUT:  DST.P
*           OUT:  CNT     )
* (ADVANCES WORK.P TO NEXT INTERP.KERNEL MODULE.  INITS SRC.P, DST.P, CNT FOR MOVE)
                REP        100
*
ADVANCE         EQU        *
                CLC
                LDY        #2                   ; Y:=0
                LDA        WORK.P               ; WORK.P:=WORK.P+(WORK.P),Y + 4
                ADC        (WORK.P),Y
                TAX
                INY
                LDA        WORK.P+1
                ADC        (WORK.P),Y
                PHA
                TXA
                ADC        #4
                STA        WORK.P
                PLA
                ADC        #0
                STA        WORK.P+1
                CLC                             ; SRC.P:=X:WORK.P+4
                LDA        WORK.P
                ADC        #>$0004
                STA        SRC.P
                LDA        WORK.P+1
                ADC        #<$0004
                STA        SRC.P+1
                LDA        CXPAGE+WORK.P+1
                STA        CXPAGE+SRC.P+1
                LDY        #0                   ; DST.P:=0:(WORK.P)
                STY        CXPAGE+DST.P+1
                LDA        (WORK.P),Y
                STA        DST.P
                INY
                LDA        (WORK.P),Y
                STA        DST.P+1
                INY                             ; Y:=2
                LDA        (WORK.P),Y           ; CNT:=(WORK.P),Y
                STA        CNT
                INY
                LDA        (WORK.P),Y
                STA        CNT+1
                RTS                             ; RETURN
                PAGE
                REP        100
*
* REVERSE ( IN:   D.HDR.CNT
*           IN:   SDT.SIZE = CONSTANT
*           I/O:  DRIVER FILE,
*           OUT:  WORK.P      )      )
*
*            LOCAL:  REV.SAVE, REV.TEMP
* (REVERSES TITLE/CODE/RELOC COUNTS TO ALLOW DRIVER FILE TO BE PROCESSED FROM BACK TO FRONT)
                REP        100
REVERSE         EQU        *
                LDA        #>D.HDR.CNT          ; WORK.P:=80:D.HDR.CNT
                STA        WORK.P
                LDA        #<D.HDR.CNT
                STA        WORK.P+1
                LDA        #$80
                STA        CXPAGE+WORK.P+1
                CLC                             ; WORK.P:=WORK.P+(WORK.P)+2
                LDY        #0
                LDA        WORK.P
                ADC        (WORK.P),Y
                TAX
                INY
                LDA        WORK.P+1
                ADC        (WORK.P),Y
                PHA
                TXA
                ADC        #2
                STA        WORK.P
                PLA
                ADC        #0
                STA        WORK.P+1
                LDA        (WORK.P),Y           ; IF (WORK.P)=$FFFF
                DEY
                AND        (WORK.P),Y           ;    THEN
                CMP        #$FF
                BNE        REV010
                LDX        #ERR10X              ;       ERROR("EMPTY DRIVER FILE")
                LDY        #ERR10L
                JSR        ERROR
REV010          LDA        #$FF
                STA        REV.SAVE
                STA        REV.SAVE+1
*
REV020          LDA        REV.SAVE             ;R1: STACK:=REV.SAVE
                PHA
                LDA        REV.SAVE+1
                PHA
                LDY        #0                   ;    REV.SAVE:=(WORK.P)
                LDA        (WORK.P),Y
                STA        REV.SAVE
                INY
                LDA        (WORK.P),Y
                STA        REV.SAVE+1
                PLA                             ;    (WORK.P):=STACK
                STA        (WORK.P),Y
                DEY
                PLA
                STA        (WORK.P),Y
                LDA        REV.SAVE             ;    IF REV.SAVE = $FFFF THEN EXIT
                AND        REV.SAVE+1
                CMP        #$FF
                BEQ        REV.EXIT
REV030          BIT        REV.SAVE+1           ;    IF REV.SAVE >= $8000 THEN ERROR
                BMI        REV040
                CLC                             ;    WORK.P:=WORK.P+REV.SAVE+2
                LDA        WORK.P
                ADC        REV.SAVE
                TAX
                LDA        WORK.P+1
                ADC        REV.SAVE+1
                PHA
                BCS        REV040
                TXA
                ADC        #2
                STA        WORK.P
                PLA
                ADC        #0
                STA        WORK.P+1
                BCC        REV020               ;    IF C=FALSE THEN R1
REV040          LDX        #ERR5X               ;                ELSE ERROR("INVALID DRIVER FILE")
                LDY        #ERR5L
                JSR        ERROR
*
REV.EXIT        RTS                             ; RETURN
                PAGE
                REP        100
*
* DADVANCE ( I/O:  WORK.P
*            OUT:  C="NO DRIVERS LEFT"
*            OUT:  SRC.P
*            OUT:  CNT
*            OUT:  REL.P )
* (ADVANCES WORK.P TO NEXT DRIVER MODULE.  INITS SRC.P, CNT, REL.P FOR RELOCATION AND MOVE)
                REP        100
DADVANCE        EQU        *
                LDY        #0                   ; IF (WORK.P)=$FFFF THEN EXIT "NO DRIVERS LEFT IN FILE"
                LDA        (WORK.P),Y
                INY
                AND        (WORK.P),Y
                CMP        #$FF
                BNE        DADV010
                SEC                             ; C:="NO DRIVERS LEFT"
                RTS                             ; RETURN
```

```
*
*
DADV010         LDA        WORK.P               ; REL.P:=X:WORK.P
                STA        REL.P
                LDA        WORK.P+1
                STA        REL.P+1
                LDA        CXPAGE+WORK.P+1
                STA        CXPAGE+REL.P+1
*
                JSR        DADD                 ; ADVANCE TO CODE COUNT FIELD
*
                LDY        #0                   ; CNT:=(WORK.P)
                LDA        (WORK.P),Y
                STA        CNT
                INY
                LDA        (WORK.P),Y
                STA        CNT+1
*
                JSR        DADD                 ; ADVANCE TO TITLE CNT FIELD
*
                CLC                             ; SRC.P:=X:WORK.P+2
                LDA        WORK.P
                ADC        #2
                STA        SRC.P
                LDA        WORK.P+1
                ADC        #0
                STA        SRC.P+1
                LDA        CXPAGE+WORK.P+1
                STA        CXPAGE+SRC.P+1
*
                JSR        DADD                 ; ADVANCE TO RELOC FIELD OF NEXT DRIVER
                CLC                             ; C:="DRIVERS LEFT"
                RTS                             ; RETURN
                PAGE
                REP        100
*
* DADD ( I/O:   WORK.P )
*
* (ADVANCES WORK.P TO NEXT FIELD IN DRIVER MODULE)
                REP        100
DADD            EQU        *
                SEC                             ; WORK.P:=WORK.P-(WORK.P)-2
                LDY        #0
                LDA        WORK.P
                SBC        (WORK.P),Y
                TAX
                INY
                LDA        WORK.P+1
                SBC        (WORK.P),Y
                PHA
                TXA
                SBC        #2
                STA        WORK.P
                PLA
                SBC        #0
                STA        WORK.P+1
                RTS                             ; RETURN
                PAGE
                REP        100
*
* FLAGS ( IN:   SRC.P
*         OUT:  PG.ALIGN
*         OUT:  FIRST.ADIB
*         OUT:  OV="ALL DIBS INACTIVE" )
*
*         LOCAL:  PREV.ADIB.P, DIB.P
* (PROCESSES "INACTIVE" & "PAGE ALIGN" FLAGS IN DRIVER MODULE'S DIBS"
                REP        100
FLAGS           EQU        *
                SEC                             ; C="FIRST DIB"
FLAG010         JSR        NEXT.DIB             ; NEXT.DIB(SRC.P.IN, DIB.P PG.ALIGN C OV.OUT)
                BVC        FLAG015              ; IF OV <> "INACTIVE" THEN ACTIVE DIB FOUND
                BCC        FLAG010              ; IF C <> "LAST DIB" THEN CHECK NEXT DIB
                RTS                             ; RETURN  (OV:="ALL DIBS INACTIVE")
*
FLAG015         PHP                             ; PUSH STATUS
                SEC                             ; FIRST.ADIB:=DIB.P-SRC.P
                LDA        DIB.P
                SBC        SRC.P
                STA        FIRST.ADIB
                LDA        DIB.P+1
                SBC        SRC.P+1
                STA        FIRST.ADIB+1
                LDA        DIB.P                ; PREV.ADIB.P:=X:DIB.P
                STA        PREV.ADIB.P
                LDA        DIB.P+1
                STA        PREV.ADIB.P+1
                LDA        CXPAGE+DIB.P+1
                STA        CXPAGE+PREV.ADIB.P+1
                PLP                             ; PULL STATUS
                BCS        FLAG100              ; IF C="LAST DIB" THEN EXIT
*
FLAG020         JSR        NEXT.DIB             ; NEXT.DIB(SRC.P.IN, DIB.P PG.ALIGN C OV.OUT)
                PHP                             ; PUSH STATUS
                LDY        #0                   ; IF OV="INACTIVE DIB"
                BVC        FLAG025
                SEC                             ;    THEN
                LDA        PREV.ADIB.P          ;       (PREV.ADIB.P):=PREV.ADIB.P-SRC.P
                SBC        SRC.P
                STA        (PREV.ADIB.P),Y
                INY
                LDA        PREV.ADIB.P+1
                SBC        SRC.P+1
                STA        (PREV.ADIB.P),Y
                JMP        FLAG050
*
FLAG025         SEC                             ;    ELSE
                LDA        DIB.P                ;       (PREV.ADIB.P):=DIB.P-SRC.P
                SBC        SRC.P
                STA        (PREV.ADIB.P),Y
                INY
                LDA        DIB.P+1
                TAX
                SBC        SRC.P+1
                STA        (PREV.ADIB.P),Y
                STX        PREV.ADIB.P+1        ;       PREV.ADIB.P:=DIB.P
                LDA        DIB.P
                STA        PREV.ADIB.P
FLAG050         PLP                             ; PULL STATUS
                BCC        FLAG020              ; IF C <> "LAST DIB" THEN PROCESS NEXT DIB
*
FLAG100         CLV                             ; OV:="ACTIVE DIBS"
                RTS                             ; RETURN
                PAGE
                REP        100
*
* NEXT.DIB ( IN:   C="FIRST DIB"
*            IN:   SRC.P
*            OUT:  DIB.P
*            OUT:  PG.ALIGN
*            OUT:  C="LAST DIB"
*            OUT:  OV="INACTIVE DIB" )
*
*            LOCAL:  DIB.FLAGS, DIB.DCB = CONSTANT
* (ADVANCES TO NEXT DIB IN DRIVER MODULE)
                REP        100
NEXT.DIB        EQU        *
                LDY        #0
                BCC        NXTD010              ; IF C = "FIRST DIB"
                STY        PG.ALIGN             ;    THEN
                STY        PG.ALIGN+1           ;       PG.ALIGN:=0
                LDA        SRC.P                ;       DIB.P:=X:SRC.P
                STA        DIB.P
                LDA        SRC.P+1
                STA        DIB.P+1
                LDA        CXPAGE+SRC.P+1
                STA        CXPAGE+DIB.P+1
                JMP        NXTD020
NXTD010         LDA        SRC.P                ;    ELSE
                ADC        (DIB.P),Y            ;       DIB.P:=SRC.P+(DIB.P)
                TAX
                INY
                LDA        SRC.P+1
                ADC        (DIB.P),Y
                STA        DIB.P+1
                STX        DIB.P
*
NXTD020         LDY        #DIB.FLAGS           ; IF (DIB.P),DIB.FLAGS.BIT7 = "INACTIVE"
                LDA        (DIB.P),Y
                BMI        NXTD030
                BIT        NXTD999              ;    THEN
                BVS        NXTD040              ;       OV:="INACTIVE"
*                                            ELSE
NXTD030         AND        #$40                 ;       IF (DIB.P),DIB.FLAGS.BIT6 = "PAGE ALIGN"
                BEQ        NXTD040
                CLC                             ;          THEN
                LDA        #DIB.DCB+2           ;             PAGE.ALIGN:=DIB.DCB+2+(SRC.P),DIB.DCB
                TAY
                DEY
                DEY
                ADC        (SRC.P),Y
                STA        PG.ALIGN
                INY
                LDA        #0
                ADC        (SRC.P),Y
                STA        PG.ALIGN+1
                CLV                             ;       OV:="ACTIVE"
*
NXTD040         LDY        #0                   ; IF (DIB.P) = 0
                LDA        (DIB.P),Y
                INY
                ORA        (DIB.P),Y
                BNE        NXTD998
                SEC                             ;    THEN  C:="LAST DIB"
                BCS        NXTD999
NXTD998         CLC                             ;    ELSE  C:=NOT "LAST DIB"
NXTD999         RTS                             ; RETURN
                REP        100

                CHN        SOSLDR.F.SRC

                RTS                             ; RETURN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.E.SRC
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.F.SRC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.F.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        100
*
* GETMEM ( IN:   PG.ALIGN
*          IN:   CNT
*          I/O:  DST.P
*          I/O:  DSTBANK
*          I/O:  DSEGLIST
*          OUT:  PREVBANK )
*
*          LOCAL:  PREVDST
* (COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT)
                REP        100
GETMEM          EQU        *
                LDA        DSTBANK              ; PREVBANK:=DSTBANK
                STA        PREVBANK
                LDA        DST.P                ; PREVDST:=DST.P
                STA        PREVDST
                LDA        DST.P+1
                STA        PREVDST+1
                JSR        NEWDST               ; NEWDST(PG.ALIGN.IN, PREVDST.IN, CNT.IN, DST.P.OUT)
*
                LDA        DST.P+1              ; IF DST.P >= $2000
                CMP        #$20
                BCC        GETM010
                SEC                             ;    THEN
                LDA        PREVDST+1            ;       A=PAGES:=PREVDST-DST.P
                SBC        DST.P+1
                CLC
                JSR        BUILD.DSEG           ;       BUILD.DSEG(C="NEXT BANK".IN, A=PAGES.IN, DSEGLIST.IO)
                JMP        GETM.EXIT
*                                             ELSE
GETM010         DEC        DSTBANK              ;       DSTBANK:=DSTBANK-1
                LDA        #>$A000              ;       PREVDST:=$A000
                STA        PREVDST
                LDA        #<$A000
                STA        PREVDST+1
                JSR        NEWDST               ;       NEWDST(PG.ALIGN.IN, PREVDST.IN, CNT.IN, DST.P.OUT)
                SEC                             ;       A="PAGES":=PREVDST-DST.P
                LDA        PREVDST+1
                SBC        DST.P+1
                SEC
                JSR        BUILD.DSEG          ;        BUILD.DSEG(C="NEXTBANK".IN, A="PAGES".IN, DSEGLIST.IO)
*
GETM.EXIT       RTS                            ; RETURN
                PAGE
                REP        100
*
* NEWDST ( IN:   PG.ALIGN
*          IN:   PREVDST
*          IN:   CNT
*          I/O:  DST.P      )
* (COMPUTES DESTINATION BASE ADDRESS, ALIGNING ON PAGE BOUNDARY IF REQUESTED)
                REP        100
NEWDST          EQU        *
                SEC                            ; IF (PREVDST-$2000) < CNT
                LDA        PREVDST
                SBC        #>$2000
                TAX
                LDA        PREVDST+1
                SBC        #<$2000
                CPX        CNT
                SBC        CNT+1
                BCS        NEWD010
                LDA        #0                  ;     THEN
                STA        DST.P               ;       DST.P:=0
                STA        DST.P+1
                BEQ        NEWD.EXIT
NEWD010         SEC                            ;     ELSE
                LDA        PREVDST             ;        DST.P:=PREVDST-CNT
                SBC        CNT
                STA        DST.P
                LDA        PREVDST+1
                SBC        CNT+1
                STA        DST.P+1
                LDA        PG.ALIGN            ;        IF PG.ALIGN <> 0
                ORA        PG.ALIGN+1          ;           THEN
                BEQ        NEWD.EXIT
                SEC                            ;               DST.P:=(DST.P/256*256)-PG.ALIGN
                LDA        #0
                SBC        PG.ALIGN
                STA        DST.P
                LDA        DST.P+1
                SBC        PG.ALIGN+1
                STA        DST.P+1
NEWD.EXIT       RTS                            ; RETURN
                PAGE
                REP        100
*
* BUILD.DSEG ( IN:   C="NEXTBANK"
*              IN:   A="PAGES"
*              I/O:  DSEGLIST     )
* (COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT)
                REP        100
BUILD.DSEG      EQU        *
                PHA
                BCS        BLDS010              ; IF ("NEXTBANK"=TRUE OR DSEGX=$FF)
                LDA        DSEGX                ;    THEN
                BPL        BLDS020
BLDS010         INC        DSEGX                ;       DSEGX:=DSEGX+1
BLDS020         LDX        DSEGX
                CLC                             ; DSEGLIST(DSEGX):=DSEGLIST(DSEGX)+"PAGES"
                PLA
                ADC        DSEGLIST,X
                STA        DSEGLIST,X
                RTS                             ; RETURN
*
*
*
DSEGX           DFB        $FF                  ; DRIVER SEGMENT LIST TABLE
DSEGLIST        DFB        $0                   ; # PAGES FOR 1ST DRIVER SEGMENT   (BANK N  )
                DFB        $0                   ;        "    2ND       "          (BANK N-1)
                DFB        $0                   ;        "    3RD       "          (BANK N-2)
                DFB        $0                   ;        "    4TH       "          (BANK N-3)
                PAGE
                REP        100
*
* RELOC ( IN:   SRC.P
*         IN:   REL.P
*         IN:   DST.P
*         OUT:  RELOCATED DRIVER MODULE )
*
*         LOCAL:  REL.END, CODE.P
* (RELOCATES DRIVER MODULE'S CODE FIELD USING RELOCATION FIELD)
                REP        100
RELOC           EQU        *
                SEC                             ; REL.END:=REL.P-(REL.P)
                LDY        #0
                LDA        REL.P
                SBC        (REL.P),Y
                STA        REL.END
                INY
                LDA        REL.P+1
                SBC        (REL.P),Y
                STA        REL.END+1
REL.LOOP        SEC                             ; REL.P:=REL.P-2
                LDA        REL.P
                SBC        #2
                STA        REL.P
                LDA        REL.P+1
                SBC        #0
                STA        REL.P+1
                LDA        REL.P                ; IF REL.P < REL.END THEN EXIT
                CMP        REL.END
                LDA        REL.P+1
                SBC        REL.END+1
                BCC        REL.EXIT
                LDY        #0                   ; CODE.P:=X:SRC.P+(REL.P)
                CLC
                LDA        SRC.P
                ADC        (REL.P),Y
                STA        CODE.P
                INY
                LDA        SRC.P+1
                ADC        (REL.P),Y
                STA        CODE.P+1
                LDA        CXPAGE+SRC.P+1
                STA        CXPAGE+CODE.P+1
                LDY        #0                   ; (CODE.P):=(CODE.P)+DST.P
                CLC
                LDA        (CODE.P),Y
                ADC        DST.P
                STA        (CODE.P),Y
                INY
                LDA        (CODE.P),Y
                ADC        DST.P+1
                STA        (CODE.P),Y
                JMP        REL.LOOP             ; GOTO REL.LOOP
*
REL.EXIT        RTS                             ; RETURN
                PAGE
                REP        100
*
* ALLOC.SEG ( IN:   K.BASE
*             IN:   I.BASE.P
*             IN:   SYSBANK )
*         I.BASE.P
*         D.BASE.PG
* (ALLOCATES SEGMENTS FOR KERNEL, INTERPRETER AND SYSTEM WORK AREA)
                REP        100
ALLOC.SEG       EQU        *
                BRK                             ; REQ.SEG(BASE=(F,0), LIMIT=(F,1D), SEGID=0, SEGNUM)
                DFB        REQSEG
                DW         SEGMENT
*
                LDA        #$10                 ; SET BASE/LIMIT BANKS
                STA        SEGBASE
                STA        SEGLIM
                LDA        #0                   ; AND INIT BASE PAGE
                STA        SEGBASE+1
```

```
*
                LDX        K.BASE+1             ; KERNEL SEGMENT, ID=1
                JSR        RSEG
*
                LDX        I.BASE.P+1           ; INTERPRETER SEGMENT, ID=2
                JSR        RSEG
                RTS
                PAGE
                REP        100
*
* RSEG ( IN:  X=BASE.PAGE OF SEGMENT )
*
                REP        100
RSEG            EQU        *
                INC        SEGID                ; SEGID:=SEGID+1
                LDY        SEGBASE+1            ; LIMIT.PAGE:=BASE.PAGE-1
                DEY
                STY        SEGLIM+1
                STX        SEGBASE+1            ; BASE.PAGE:=X
*
                CPX        #$A0                 ; IF BASE>=$A0 OR LIMIT<$A0 THEN
                BCS        RSEG010              ;    THEN
                LDA        SEGLIM+1             ;       REQUEST ONLY ONE SEGMENT
                CMP        #$A0
                BCC        RSEG010
*
                TXA                             ;    ELSE
                PHA                             ;       REQUEST TWO SEGMENTS
                LDX        #$A0
                STX        SEGBASE+1
*
                BRK                             ;       REQ.SEG(BASE, LIMIT, SEGID, SEGNUM)
                DFB        REQSEG
                DW         SEGMENT
*
                PLA
                STA        SEGBASE+1
                LDA        #$9F
                STA        SEGLIM+1
                LDA        SYSBANK
                STA        SEGBASE
                STA        SEGLIM
*
*
RSEG010         BRK                             ; REQ.SEG(BASE, LIMIT, SEGID, SEGNUM)
                DFB        REQSEG
                DW         SEGMENT
*
                RTS                             ; RETURN
                PAGE
                REP        100
*
* ALLOC.DSEG ( IN:   DSEGLIST )
*
* (ALLOCATES SEGMENTS FOR DRIVER MODULES"
                REP        100
ALLOC.DSEG      EQU        *
                INC        DSEGX                ; DSEGX:=DSEGX+1
                BNE        ALDS010              ; IF DSEGX=0
                LDX        #ERR5X               ;    THEN ERROR("INVALID DRIVER FILE")
                LDY        #ERR5L
                JSR        ERROR
*
ALDS010         LDY        #$FF                 ; Y:=-1
ALDS020         INY                             ; WHILE (Y:=Y+1) < DSEGX
                CPY        DSEGX                ;    DO
                BCS        ALDS.EXIT
                LDA        DSEGLIST,Y           ;       PAGECT:=DSEGLIST(Y)
                STA        SEGPGCNT
                BRK                             ;       FINDSEG (SRCHMODE=0.IN, SEGID=3.IN
                DFB        FINDSEG              ;               PAGECT=DSEGLIST(Y)
                DW         SEGMENT1             ;               BASE.OUT, LIMIT.OUT)
                JMP        ALDS020
*
ALDS.EXIT       RTS                             ; RETURN
                PAGE
                REP        100
*
* ERROR (IN:  X=MESSAGE INDEX
*        IN:  Y=MESSAGE LENGTH
* (DISPLAYS ERROR MESSAGE, SOUNDS BELL AND LOOPS UNTIL CONTROL/RESET PRESSED)
                REP        100
ERROR           EQU        *
                STY        ETEMP                ; CENTER MSG (Y:=LEN/2+LEN)
                SEC
                LDA        #40
                SBC        ETEMP
                LSR        A
                CLC
                ADC        ETEMP
                TAY
*
PRNT010         LDA        ERR,X                ; MOVE MESSAGE TO SCREEN MEMORY
                STA        EMSGADR-1,Y
                DEX
                DEY
                DEC        ETEMP
                BNE        PRNT010
*
                LDA        #$73                 ; E:=( 0.1.1.1:0.0.1.1 )
                STA        E.REG                ;    ( 1.I.S.R:W.P.R.S )
                LDA        $C040                ; SOUND BELL
                JMP        *                    ; LOOP UNTIL REBOOT (CTRL/RESET)
                PAGE
                REP        100
*
* ERROR MESSAGES
*
                REP        100
EMSGADR         EQU        $7A8
*
ERR             EQU        *
ERR0            ASC        "I/O ERROR"
ERR0L           EQU        *-ERR0
ERR0X           EQU        *-ERR-1
ERR1            ASC        "INTERPRETER FILE NOT FOUND"
ERR1L           EQU        *-ERR1
ERR1X           EQU        *-ERR-1
ERR2            ASC        "INVALID INTERPRETER FILE"
ERR2L           EQU        *-ERR2
ERR2X           EQU        *-ERR-1
ERR3            ASC        "INCOMPATIBLE INTERPRETER"
ERR3L           EQU        *-ERR3
ERR3X           EQU        *-ERR-1
ERR4            ASC        "DRIVER FILE NOT FOUND"
ERR4L           EQU        *-ERR4
ERR4X           EQU        *-ERR-1
ERR5            ASC        "INVALID DRIVER FILE"
ERR5L           EQU        *-ERR5
ERR5X           EQU        *-ERR-1
ERR6            ASC        "DRIVER FILE TOO LARGE"
ERR6L           EQU        *-ERR6
ERR6X           EQU        *-ERR-1
ERR7            ASC        "ROM ERROR:  PLEASE NOTIFY YOUR DEALER"
ERR7L           EQU        *-ERR7
ERR7X           EQU        *-ERR-1
ERR8            ASC        "TOO MANY DEVICES"
ERR8L           EQU        *-ERR8
ERR8X           EQU        *-ERR-1
ERR9            ASC        "TOO MANY BLOCK DEVICES"
ERR9L           EQU        *-ERR9
ERR9X           EQU        *-ERR-1
ERR10           ASC        "EMPTY DRIVER FILE"
ERR10L          EQU        *-ERR10
ERR10X          EQU        *-ERR-1
                PAGE
                REP        100
*
* WELCOME ()
*
```

```
* (PRINTS WELCOME MESSAGE - "APPLE ///", VERSION, DATE/TIME, COPYRIGHT)
                REP        100
WELCOME         EQU        *
*
*  PRINT "APPLE III" MESSAGE
*
                LDY        #AMSGL
WAM010          LDA        AMSG-1,Y
                STA        AMSGADR-1,Y
                DEY
                BNE        WAM010
*
*  PRINT SOS VERSION MESSAGE
*
                CLC
                LDA        #40
                ADC        #>SOSVERL
                LSR        A
                TAX
                LDY        #>SOSVERL
WSM010          LDA        SOSVER-1,Y
                ORA        #$80
                STA        SMSGADR-1,X
                DEX
                DEY
                BNE        WSM010
*
*  PRINT DATE AND TIME MESSAGE
*
                BRK                             ; GET.TIME(TIME.OUT)
                DFB        GETTIME
                DW         DTPARMS
*
                LDA        DATETIME+8           ;SET UP WEEKDAY
                AND        #$0F
                BEQ        WDM040               ;NO CLOCK
                STA        WTEMP
                ASL        A
                ADC        WTEMP
                TAX
                LDY        #3
WDM010          LDA        DAYNAME-1,X
                STA        DMSG-1,Y
                DEX
                DEY
                BNE        WDM010
*
                LDA        DATETIME+7           ;SET UP DATE
                LDX        DATETIME+6
                STA        DMSG+6
                STX        DMSG+5
*
                LDA        DATETIME+5           ;SET UP MONTH
                AND        #$0F
                LDX        DATETIME+4
                CPX        #$31
                BCC        WDM020
                ADC        #9
WDM020          STA        WTEMP
                ASL        A
                ADC        WTEMP
                TAX
                LDY        #3
WDM030          LDA        MONNAME-1,X
                STA        DMSG+7,Y
                DEX
                DEY
                BNE        WDM030
*
                LDA        DATETIME+3           ;SET UP YEAR
                LDX        DATETIME+2
                STA        DMSG+13
                STX        DMSG+12
*
                LDA        DATETIME+10          ;SET UP HOUR
                LDX        DATETIME+09
                STA        DMSG+17
                STX        DMSG+16
*
                LDA        DATETIME+12          ;SET UP MINUTE
                LDX        DATETIME+11
                STA        DMSG+20
                STX        DMSG+19
*
                LDY        #DMSGL               ;PRINT DATE & TIME
WDM050          LDA        DMSG-1,Y
                ORA        #$80
                STA        DMSGADR-1,Y
                DEY
                BNE        WDM050
*
*  PRINT COPYRIGHT MESSAGE
*
WDM040          LDY        #CMSGL
WCM010          LDA        CMSG-1,Y
                STA        CMSGADR-1,Y
                DEY
                BNE        WCM010
                RTS
                PAGE
                REP        100
*
* WELCOME () - DATA DECLARATIONS
*
                REP        100
                MSB        ON
AMSG            ASC        "APPLE ///"
AMSGL           EQU        *-AMSG
AMSGADR         EQU        40-AMSGL/2+$4A8
                MSB        OFF
SMSGADR         EQU        $5A8
DMSG            ASC        "DAY, DD-MON-YY  HH:MM"
DMSGL           EQU        *-DMSG
DMSGADR         EQU        40-DMSGL/2+$6A8
DAYNAME         ASC        "SUNMONTUEWEDTHUFRISAT"
MONNAME         ASC        "JANFEBMARAPRMAYJUN"
                ASC        "JULAUGSEPOCTNOVDEC"
                MSB        ON
CMSG            ASC        "(C)1980,1981,1982 BY APPLE COMPUTER INC."
CMSGL           EQU        *-CMSG
CMSGADR         EQU        40-CMSGL/2+$7D0
                MSB        OFF
                PAGE
                REP        100
*
* SOS SYSTEM CALLS (1)
*
                REP        100
* OPEN (PATHNAME.IN, REFNUM.OUT, OPENLIST.IN, OPENCNT.IN)  ** (ACCESS.IN, PAGES.IN, SYSBUF.IN)
                REP        100
OPEN            EQU        $C8
*
OPEN.PARMS      DFB        $4
                DW         PATH
OPEN.REF        DFB        $0
                DW         OPEN.LIST
                DFB        $4
OPEN.LIST       DFB        $0,$4                 ; PAGES:=4
                DW         SYSBUF.P
PATH            DS         $40                   ; PATHNAME BUFFER
I.LABEL         ASC        "SOS NTRP"            ; FILE LABELS
D.LABEL         ASC        "SOS DRVR"
                REP        100
* READ (REFNUM.IN, BUFFER.IN, BYTES.IN, BYTESREAD.OUT)
                REP        100
READ            EQU        $CA
*
READ.PARMS      DFB        $4
READ.REF        DFB        $0
READ.BUF        DW         RDBUF.P
READ.BYT        DW         $FFFF-FILE+1
READ.BYTRD      DW         $0
                REP        100
* CLOSE (REFNUM.IN)
                REP        100
CLOSE           EQU        $CC
*
CLOSE.PARMS     DFB        $1
CLOSE.REF       DFB        $0
                REP        100
* FIND.SEG (SRCHMODE.IN, PAGES.IN, SEGID.IN, BASE.OUT, LIMIT.OUT, SEGNUM.OUT)
                REP        100
FINDSEG         EQU        $41
*
SEGMENT1        DFB        $6                   ; FIND.SEG(SRCHMODE, SEGID, PAGECT, BASE, LIMIT, SEGNUM)
SEGSRCH         DFB        $0,$3
SEGPGCNT        DW         $0000
                DW         $0
                DW         $0
                DFB        $0
                PAGE
                REP        100
*
* SOS SYSTEM CALLS (2)
*
                REP        100
                REP        100
* REQUEST.SEG (BASE.IN, LIMIT.IN, SEGID.IN, SEGNUM.OUT)
                REP        100
REQSEG          EQU        $40
*
SEGMENT         DFB        $4                   ; REQUEST SEG PARM LIST
SEGBASE         DFB        $F,$0
SEGLIM          DFB        $F,$1D
SEGID           DFB        $0,$0
                REP        100
* SET.PREFIX (PREFIXPATH.IN)
                REP        100
SETPREFIX       EQU        $C6
PREFX.PARMS     DFB        $1
                DW         PREFX.PATH
PREFX.PATH      DFB        $3
                ASC        '.D1'
                REP        100
* GETTIME (TIME.OUT)
                REP        100
GETTIME         EQU        $63
*
DTPARMS         DFB        1
                DW         DATETIME
```

```
DATETIME        ASC        "YYYYMMDDWHHMMSSMMM"
                PAGE
                REP        100
*
* END OF SOSLDR CODE
*
                REP        100
SLOP            EQU        >$F8-*
                DS         SLOP                 ; +-----------------------------------+
INITMODULE      DS         $200                 ; ! KERNEL'S INIT MODULE RESIDES HERE !
LDREND          EQU        *                    ; +-----------------------------------+
FILE            EQU        *-$2000+$400
                REP        100
* SOS INTERPRETER FILE
                REP        100
I.FILE          EQU        FILE
I.HDR.CNT       EQU        I.FILE+$8
                REP        100
* SOS DRIVER FILE
                REP        100
D.FILE          EQU        FILE
D.HDR.CNT       EQU        D.FILE+$8
D.DRIVES        EQU        D.HDR.CNT+$2
D.CHRSET        EQU        D.DRIVES+$2+$10
D.KYBD          EQU        D.CHRSET+$10+$400
                REP        100

                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
*
NE              ZZLEN-LENLODR
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR SOS LOADER"
                FIN
*

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.F.SRC
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.SRC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1 SOS LOADER"
                REL
                ORG        $1E00
ZZORG           EQU        *
                MSB        OFF
                REP        100
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        100
*
*        SOS KERNEL LOAD & MEMORY POINTS
*
*  MODULE      START  END   I/O  ROM  SOS BLOAD    SIZE
*------------------------------------------------------
*  SOSLDR      1E00 - 28F7             2000         0CF8
*  INIT        28F8 - 2AA9             2AF8        [01B2]
*  SYSGLOB     18FC - 1A03             2CF8
*
*  BFM.INIT2 + BITMAPS
*              B800 - BBFF             2E00         03FF
*  BFM         BC00 - DE62             3200         2263
*  <PATCH>     DE63 - DE6A             5463         0008
*
*  OPRMSG      DE6B - E48A   X         546B         015A
*  IPL         DFC5 - E48F   X    X    55C5         04CB
*  UMGR        E490 - E89D   X    X    5A8B         040E
*
*  DISK3       E899 - EE03   X    X    5E99         056B
*  SYSERR      EE04 - EED8   X         64D9         00D5
*  DEVMGR      EED9 - F05D             64D9         0185
*
*  SCMGR       F05E - F2F3             665E         0296
*  FMGR        F2F4 - F354             68F4         0061
*  CFMGGR      F355 - F551             6955         01FD
*
*  BUFMGR      F552 - F86D             6B52         031C
*  MEMMGR      F86E - FFBE             6E6E         0751
*  <END>       FFBE
*
                REP        100
```

```
* SOS LOADER  (VERSION = 1.1O   )
*             (DATE    = 8/04/81)
*
* SOURCE FILES:  SOSLDR.SRC,   SOSLDR.A.SRC, SOSLDR.B.SRC, SOSLDR.C.SRC,
*                              SOSLDR.D.SRC, SOSLDR.E.SRC, SOSLDR.F.SRC
*
* FUNCTION:
*    MOVES AND INITIALIZES SOS KERNEL, READS INTERPRETER FROM DISK, READS CHARACTER SET TABLE,
*    KEYBOARD TABLE AND DRIVERS FROM DISK, INITIALIZES ALL DRIVERS AND THEN JUMPS TO INTERPRETER
*    ENTRY POINT.
*
* CALLED BY:
*    SOSBOOT 7.0 WITH KERNEL FILE LOADED AT $I:1E00.9FFF(MAX)
*    WHERE: $I=INTERPRETER BANK (HIGHEST BANK IN SYSTEM)
*
* CALLS:
*    INTERPRETER ENTRY POINT (FIRST BYTE OF INTERPRETER CODE)
*
* DOCUMENTS:
*    SOS ERS APPENDICES - XX/XX/81
*    APPLE III I/O SYSTEM PROGRAMMERS GUIDE - DEC-15-80
*
* CONSTRAINTS:
*    INTERPRETER FILE:  READ INTO BANK 0 BEGINNING AT $80:LDREND+$400(=BUFSIZE).
*                       INTERPRETER CODE DOES NOT CONTAIN RELOCATION INFORMATION.
*                       MAX = 38K  ($I:2000..B7FF)
*                       MIN = .25K ($I:B700..B7FF)
*
*    DRIVER FILE:  READ INTO BANK 0 BEGINNING AT $80:LDREND+$400(=BUFSIZE).
*                  DRIVER MODULES ARE RELOCATED AND MOVED TO THE HIGHEST AVAILABLE 32K BANK USING
*                  A "FIRST FIT" ALGORITHM.  MODULES ARE REMOVED FROM THE FILE BEGINNING AT THE BACK
*                  AND WORKING TOWARD THE FRONT.  A DRIVER MODULE CANNOT SPAN A BANK BOUNDARY.
*
*                  DRIVER FILE:  MAX = 60K  (APPROX)       DRIVER MODULE:  MAX = 32K-1
*                                MIN = .25K                                MIN < .25K
*
*
* DATA STRUCTURES:
*    SOS.KERNEL FILE FORMAT
*    SOS.INTERP FILE FORMAT
*    SOS.DRIVER FILE FORMAT
*
                REP        100
                PAGE
                REP        100
*
* NOTATION:
*
*    A, X, Y           ::= 6502 REGISTERS
*
```

```
*    C, OV             ::= CARRY, OVERFLOW FLAGS IN 6502 STATUS (P) REGISTER
*    E, Z, B           ::= ENVIRONMENT, ZERO PAGE, BANK REGISTERS (SYSTEM CONTROL REGISTERS)
*
*    (1.I.S.R:W.P.R.R) ::= ENVIRONMENT REGISTER FLAGS.  FROM LEFT TO RIGHT BITS 7..0
*                          (1MHZ, I/O ENABLE, SCREEN ENABLE, RESET ENABLE,
*                           WRITE PROTECT, PRIMARY STACK, ROM1, ROM ENABLE)
*
*    "POSITIVE LOGIC"  ::= ALL LOGIC USED IS POSITIVE LOGIC.  FOR EXAMPLE, C="NO DRIVERS LEFT"
*                          INDICATES THAT NO DRIVERS ARE LEFT WHEN CARRY = SET, AND THAT ONE OR
*                          MORE DRIVERS ARE LEFT WHEN CARRY = CLEAR.
*
*    TRUE,FALSE        ::= TRUE = SET = ON, WHILE FALSE = CLEAR = OFF.
*
                REP        100
*
* ABBREVIATIONS:
*
*    DIB               ::= DEVICE INFORMATION BLOCK.  DEFINES A UNIQUE DEVICE THAT CAN BE LINKED
*                          INTO THE SYSTEM DEVICE TABLE.  EACH DRIVER MODULE CONTAINS ONE OR MORE
*                          DIBS (DEVICES) EACH OF WHICH CAN BE "ACTIVE" OR "INACTIVE".
*
*    ADIB              ::= "ACTIVE DIB"
*
*   <VARNAME>.P        ::=  POINTER.  A 3 BYTE ZERO PAGE POINTER.  DON'T FORGET THE X BYTE!
*
*    SDT               ::= SYSTEM DEVICE TABLE.  CONTAINS THE ENTRY POINT AND DIB ADDRESS OF EACH
*                          DEVICE CONFIGURED INTO THE SYSTEM, (USED BY THE DEVICE MANAGER).
                REP        100

                CHN        SOSLDR.A.SRC

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.SRC
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.1of5.ONE:SOS.SYSGLOB.SRC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SYSGLOB.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1  GLOBAL EQUATES"
                REL
                ORG        $18FC
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
*  SOS SYSTEM GLOBAL DATA & EQUATES
*
*  THIS MODULE CONTAINS THE SOS JUMP TABLE, AND ALL GLOBAL
*  DATA AND EQUATES.  THE JUMP TABLE, AND ALL DATA THAT IS
*  TO BE REFERENCED BY DEVICE HANDLERS, ARE ASSIGNED FIXED
*  ADDRESSES AT THE BEGINNING OF MEMORY PAGE $19.  DATA
*  THAT IS ONLY REFERENCED BY SOS BEGINS $1980, BUT MAY BE
*  MOVED WHENEVER SOS IS RELINKED.
*
                REP        60
*
                EXTRN      ALLOCSIR
                EXTRN      DEALCSIR
                EXTRN      NMIDSBL
                EXTRN      NMIENBL
                EXTRN      QUEEVENT
                EXTRN      SELC800
                EXTRN      SYSDEATH
                EXTRN      SYSERR
                EXTRN      REQBUF
                EXTRN      GETBUFADR
                EXTRN      RELBUF
                EXTRN      NMIDBUG
                EXTRN      NMICONT
                EXTRN      COLDSTRT
*
*
                ENTRY      MEMSIZE
                ENTRY      SYSBANK
                ENTRY      SUSPFLSH
                ENTRY      NMIFLAG
                ENTRY     SCRNMODE
                ENTRY     GRSIZE
*
                ENTRY     SERR
                ENTRY     DBUGBRK
                ENTRY     KYBDNMI
                ENTRY     NMISPSV
                ENTRY     SDEATH.REGS
*
                ENTRY     SOSVER
                ENTRY     SOSVERL
*
                ENTRY     SZPAGE
                ENTRY     SXPAGE
                ENTRY     SSPAGE
*
                ENTRY     CZPAGE
                ENTRY     CXPAGE
                ENTRY     CSPAGE
                ENTRY     CEVPRI
*
                ENTRY     SIRTEMP
                ENTRY     SIRARGSIZ
                ENTRY     IRQCNTR
                ENTRY     NMICNTR
                ENTRY     QEVTEMP
                ENTRY     QEV.THIS
                ENTRY     QEV.LAST
*
                ENTRY     BADBRK
                ENTRY     BADINT1
                ENTRY     BADINT2
                ENTRY     NMIHANG
                ENTRY     EVQOVFL
                ENTRY     STKOVFL
                ENTRY     BADSYSCALL
                ENTRY     DEV.OVFLOW
                ENTRY     MEM2SML
                ENTRY     VCBERR
                ENTRY     FCBERR
                ENTRY     ALCERR
                ENTRY     DIRERR
                ENTRY     TOOLONG
                ENTRY     BADBUFNUM
                ENTRY     BADBUFSIZ
                ENTRY     BITMAPADR
*
                ENTRY     BADSCNUM
                ENTRY     BADCZPAGE
                ENTRY     BADXBYTE
                ENTRY      BADSCPCNT
                ENTRY      BADSCBNDS
*
                ENTRY      NODNAME
                ENTRY      BADDNUM
*
                ENTRY      BADPATH
                ENTRY      CFCBFULL
                ENTRY      FCBFULL
                ENTRY      BADREFNUM
                ENTRY      PATHNOTFND
                ENTRY      VNFERR
                ENTRY      FNFERR
                ENTRY      DUPERR
                ENTRY      OVRERR
                ENTRY      DIRFULL
                ENTRY      CPTERR
                ENTRY      TYPERR
                ENTRY      EOFERR
                ENTRY      POSNERR
                ENTRY      ACCSERR
                ENTRY      BTSERR
                ENTRY      FILBUSY
                ENTRY      NOTSOS
                ENTRY      BADLSTCNT
                ENTRY      OUTOFMEM
                ENTRY      BUFTBLFULL
                ENTRY      BADSYSBUF
                ENTRY      DUPVOL
                ENTRY      NOTBLKDEV
                ENTRY      LVLERR
*
                ENTRY      BADJMODE
*
                ENTRY      BADBKPG
                ENTRY      SEGRQDN
                ENTRY      SEGTBLFULL
                ENTRY      BADSEGNUM
                ENTRY      SEGNOTFND
                ENTRY      BADSRCHMODE
                ENTRY      BADCHGMODE
                ENTRY      BADPGCNT
*
                ENTRY      XREQCODE
                ENTRY      XCTLCODE
                ENTRY      XCTLPARM
                ENTRY      XNOTOPEN
                ENTRY      XNOTAVAIL
                ENTRY      XNORESRC
                ENTRY      XBADOP
                ENTRY      XIOERROR
                ENTRY      XNODRIVE
                ENTRY      XNOWRITE
                ENTRY      XBYTECNT
                ENTRY      XBLKNUM
                ENTRY      XDISKSW
                ENTRY      BACKMASK             ; MASK BYTE FOR BACKUP BIT.
*
                ENTRY      E1908                ; DISK DRIVER IS READING/WRITING (SET) ELSE NOT (RESET)
*
                PAGE
                DW         SYSGLOB              ;SYSGLOB TARGET ADDRESS
                DW         $0100                ;  AND LENGTH
*
*  SYSTEM GLOBAL DATA
*    (ACCESSIBLE TO SOS AND DEVICE HANDLERS)
*
SYSGLOB         EQU        *
*
MEMSIZE         DFB        $08                  ;MEMORY SIZE = 128K
SYSBANK         DFB        $02                  ;SYSTEM BANK = 2
SUSPFLSH        DFB        $00                  ;SYSOUT SUSPEND/FLUSH FLAG
NMIFLAG         DFB        $00                  ;NMI PENDING FLAG
                DW         NMIEXIT              ;DEFAULT NMI VECTOR
SCRNMODE        DFB        $80                  ;CURRENT SCREEN MODE
GRSIZE          DFB        $00
*
*
*  SOS JUMP TABLE
*
                DS         SYSGLOB+$10-*,$00    ; USED BY THE MOUSE DRIVER
USERNMI         JMP        NMIEXIT              ;KEYBOARD NMI VECTOR
                JMP        ALLOCSIR             ;ALLOCATE A SIR
                JMP        DEALCSIR             ;DEALLOCATE A SIR
                JMP        NMIDSBL              ;DISABLE NMI
                JMP        NMIENBL              ;ENABLE NMI
                JMP        QUEEVENT             ;QUEUE AN EVENT
                JMP        SELC800              ;SELECT I/O EXPANSION ROM
                JMP        SYSDEATH             ;SYSTEM DEATH
                JMP        SYSERR               ;SOS ERROR
                JMP        REQBUF               ;REQUEST BUFFER
                JMP        GETBUFADR            ;GET BUFFER'S ADDRESS
                JMP        RELBUF               ;RELEASE BUFFER
                JMP        CLRBMASK             ;VECTOR TO CLRBMASK
                PAGE
*
*  SOS DATA AND EQUATES
*    (ACCESSIBLE ONLY TO SOS)
*
                DS         SYSGLOB+$80-*,$00
```

```
SERR            DFB        $00                  ;SYSTEM ERROR CODE
*
DBUGBRK         NOP                             ;TO ENABLE DEBUG BREAK POINTS,
                PLA                             ;  PATCH THESE BYTES TO
                PLA                             ;  JMP TO THE DEBUGGER
                RTS
*
KYBDNMI         JMP        USERNMI
                JMP        NMIDBUG
NMISPSV         DFB        0
                JMP        NMICONT
NMIEXIT         RTS
*
*
SOSVER          ASC        "SOS 1.3   01-DEC-82"
SOSVERL         EQU        *-SOSVER
*
                ASC        "(C) 1980, 1981 BY APPLE COMPUTER INC."
*
E1908           EQU        $1908                ; ALLOCATED TO STEPHEN SMITH (MOUSE DRIVER)
* ABOVE SET AND RESET IN DISK DRIVER
SZPAGE          EQU        $1800                ;SYSTEM ZERO PAGE
SXPAGE          EQU        $1400                ;SYSTEM EXTEND PAGE
SSPAGE          EQU        $0100                ;SYSTEM STACK PAGE
*
CZPAGE          EQU        $1A00                ;CALLER'S ZERO PAGE
CXPAGE          EQU        $1600                ;CALLER'S EXTEND PAGE
CSPAGE          EQU        $1B00                ;CALLER'S STACK PAGE
CEVPRI          DFB        $00                  ;CALLER'S EVENT PRIORITY
*
SIRTEMP         DFB        $00                  ;TEMP FOR ALLOCSIR & DEALCSIR
SIRARGSIZ       DFB        $00                  ;ARGUMENT COUNT FOR ALLOCSIR & DEALCSIR
IRQCNTR         DW         $0000                ;FALSE IRQ COUNTER
NMICNTR         DW         $0000                ;COUNTER FOR NMILOCK
QEVTEMP         DFB        $00                  ;TEMP FOR QUEEVENT
QEV.THIS        DFB        $00                  ;POINTER FOR QUEEVENT
QEV.LAST        DFB        $00                  ;POINTER FOR QUEEVENT
*
SOSQUIT         DS         COLDSTRT
BACKMASK        DFB        BACKBIT              ; MASK USED BY BFM TO UPDATE BACKUP BIT
*
* TO CLEAR THE BACKUP BIT, A PROGRAM MUST JSR TO CLRBMASK THRU 1934 THEN DO A
* SET-FILE-INFO WITH NO INTERVENING SOS CALLS.  ANY SOS CALL WILL
* SET BACKMASK AGAIN.  THIS FEATURE IS INTENTIONALLY LEFT UNDOCUMENTED.
*
CLRBMASK        AND        #BACKBIT             ; PURIFY
                STA        BACKMASK             ; SET THE MASK
                RTS                             ; AND BACK TO THE CALLER
                PAGE
*
```

```
*  SYSTEM DEATH REGISTER SAVE AREA
* (SYSTEM STACK MOVED TO $1700-$17FF)
*
                DS         SYSGLOB+$F6-*,$00
SDEATH.REGS     EQU        *
                DFB        $00                     ;BANK
                DFB        $00                     ;ZERO PAGE
                DFB        $00                     ;ENVIRONMENT
                DFB        $00                     ;Y
                DFB        $00                     ;X
                DFB        $00                     ;A
                DFB        $00                     ;STATUS
                DW         $00                     ;PROGRAM COUNTER
                DFB        $00                     ;STACK POINTER
*
*  SYSTEM DEATH ERROR NUMBERS
*
BADBRK          EQU        $01                     ;BRK FROM SOS
BADINT1         EQU        $02                     ;INTERRUPT NOT FOUND
BADINT2         EQU        $03                     ;BAD ZERO PAGE ALLOCATION
NMIHANG         EQU        $04                     ;UNABLE TO LOCK NMI
EVQOVFL         EQU        $05                     ;EVENT QUEUE OVERFLOW
STKOVFL         EQU        $06                     ;STACK OVERFLOW
*
BADSYSCALL      EQU        $07                     ;DMGR DETECTED INVALID REQUEST CODE
DEV.OVFLOW      EQU        $08                     ;DMGR - TOO MANY DEVICE HANDLERS
MEM2SML         EQU        $09                     ;MEMORY SIZE < 64K
VCBERR          EQU        $0A                     ;VOLUME CONTROL BLOCK NOT USABLE (BFMGR)
FCBERR          EQU        $0B                     ;FILE CONTROL BLOCK CRASHED
ALCERR          EQU        $0C                     ;ALLOCATION BLOCKS INVALID
TOOLONG         EQU        $0E                     ;PATHNAME BUFFER OVERFLOW
BADBUFNUM       EQU        $0F                     ;INVALID BUFFER NUMBER
BADBUFSIZ       EQU        $10                     ;INVALID BUFFER SIZE (=0 OR >16K)
                PAGE
*
*  SYSTEM ERROR NUMBERS
*
* - SYSTEM CALL MANAGER
*
BADSCNUM        EQU        $01                     ;BAD SYSTEM CALL NUMBER
BADCZPAGE       EQU        $02                     ;BAD CALLER'S ZPAGE (MUST=$1A)
BADXBYTE        EQU        $03                     ;BITS         6..4 <> 0
BADSCPCNT       EQU        $04                     ;BAD SYSTEM CALL PARM COUNT
BADSCBNDS       EQU        $05                     ;SYS CALL PARM ADR
*
* - DEVICE MANAGER
*
NODNAME         EQU        $10                     ;DEVICE NAME NOT FOUND
BADDNUM         EQU        $11                     ;INVALID DEV.NUM PARM
*
```

```
* - DEVICE HANDLERS (STANDARD ERRORS)
*
XREQCODE        EQU        $20                  ;INVALID REQUEST CODE
XCTLCODE        EQU        $21                  ;INVALID CONTROL/STATUS CODE
XCTLPARM        EQU        $22                  ;INVALID CONTROL/STATUS PARM
XNOTOPEN        EQU        $23                  ;DEVICE NOT OPEN
XNOTAVAIL       EQU        $24                  ;DEVICE NOT AVAILABLE
XNORESRC        EQU        $25                  ;UNABLE TO OBTAIN RESOURCE
XBADOP          EQU        $26                  ;INVALID OPERATION
XIOERROR        EQU        $27                  ;I/O ERROR
*
XNODRIVE        EQU        $28                  ;NO DRIVE CONNECTED
XNOWRITE        EQU        $2B                  ;DEVICE WRITE PROTECTED
XBYTECNT        EQU        $2C                  ;BYTE COUNT <> A MULTIPLE OF 512
XBLKNUM         EQU        $2D                  ;BLOCK NUMBER TOO LARGE
XDISKSW         EQU        $2E                  ;DISK MEDIA HAS BEEN SWITCHED
*
* - NOTE: ERROR CODES $30-$3F HAVE BEEN RESERVED FOR DEVICE
*   HANDLER SPECIFIC ERRORS
*
*
* - FILE MANAGER
*
BADPATH         EQU        $40                  ;PATHNAME, INVALID SYNTAX
CFCBFULL        EQU        $41                  ;CHAR FILE CTRL BLOCK TABLE FULL
FCBFULL         EQU        $42                  ;BLOCK FILE CTRL BLOCK TABLE FULL
BADREFNUM       EQU        $43                  ;INVALID REF.NUM PARM
PATHNOTFND      EQU        $44                  ;PATHNAME NOT FOUND
VNFERR          EQU        $45                  ;VOLUME NOT FOUND
FNFERR          EQU        $46                  ;FILE NOT FOUND
DUPERR          EQU        $47                  ;DUPLICATE FILE NAME ERROR
OVRERR          EQU        $48                  ;NOT ENOUGH DISK SPACE FOR PREALLOCATION
DIRFULL         EQU        $49                  ;DIRECTORY FULL ERROR
CPTERR          EQU        $4A                  ;FILE INCOMPATIBLE SOS VERSION
TYPERR          EQU        $4B                  ;NOT CURRENTLY SUPPORTED FILE TYPE
EOFERR          EQU        $4C                  ;POSITION ATTEMPTED BEYOND END OF FILE
POSNERR         EQU        $4D                  ;ILLEGAL POSITION (L.T. 0 OR G.T. $FFFFFF)
ACCSERR         EQU        $4E                  ;FILE ACCESS R/W REQUEST CONFLICTS WITH ATTRIBUTES
BTSERR          EQU        $4F                  ;USER SUPPLIED BUFFER TOO SMALL
FILBUSY         EQU        $50                  ;EITHER WRITE WAS REQUESTED OR WRITE ACCESS ALREADY ALLOCATED
DIRERR          EQU        $51                  ;DIRECTORY ERROR
NOTSOS          EQU        $52                  ;NOT A SOS DISKETTE
BADLSTCNT       EQU        $53                  ;INVALID VALUE IN LIST PARAMETER
OUTOFMEM        EQU        $54                  ;OUT OF FREE MEMORY FOR BUFFER
BUFTBLFULL      EQU        $55                  ;BUFFER TABLE FULL
BADSYSBUF       EQU        $56                  ;INVALID SYSBUF PARAMETER
DUPVOL          EQU        $57                  ;SON A BITCH GOT TWO VOLUMES OF SAME ROOT NAME!!!
NOTBLKDEV       EQU        $58
LVLERR          EQU        $59                  ;INVALID FILE LEVEL
BITMAPADR       EQU        $5A
```

```
BACKBIT         EQU        $20                  ; MASK FOR BACKUP BIT
*
* - UTILITY MANAGER
*
BADJMODE        EQU        $70                  ;INVALID JOYSTICK REQUEST
*
* - MEMORY MANAGER
*
BADBKPG         EQU        $E0                  ;INVALID BANK/PAGE PAIR
SEGRQDN         EQU        $E1                  ;SEGMENT REQUEST DENIED
SEGTBLFULL      EQU        $E2                  ;SEGMENT TABLE FULL
BADSEGNUM       EQU        $E3                  ;INVALID SEGMENT NUMBER
SEGNOTFND       EQU        $E4                  ;SEGMENT NOT FOUND
BADSRCHMODE     EQU        $E5                  ;INVALID SEARCH MODE PARM
BADCHGMODE      EQU        $E6                  ;INVALID CHANGE MODE PARM
BADPGCNT        EQU        $E7                  ;INVALID PAGE COUNT PARM
                ORG        SYSGLOB+$100
                DW         $B800                ;KERNEL TARGET ADDRESS
                DW         $47C0                ;  AND LENGTH

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SYSGLOB.SRC
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.BUGMGR.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: BUFMGR.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1  BUFFER MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
*ORGBUFMG EQU $F552
*LENBUFMG EQU $31C
                ORG        ORGBUFMG
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* BUFFER MANAGER (VERSION = 1.10   )
*                (DATE    = 8/04/81)
*
* THIS MODULE IS RESPONSIBLE FOR CREATING AND RELEASING BUFFERS
* FOR BOTH THE BLOCK FILE MANAGER AND, LATER, DEVICE HANDLERS
* THE BUFFER MANAGER CREATES BUFFERS BY REQUESTING MEMORY
* SEGMENTS FROM THE MEMORY MANAGER, AND RELEASES THEM VIA SAME.
* THE PRIMARY DATA STRUCTURE IN THIS MODULE IS THE BUFFER TABLE.
*
                REP        60
*
                ENTRY      REQBUF
                ENTRY      REQFXBUF
                ENTRY      GETBUFADR
                ENTRY      CHKBUF
                ENTRY      RELBUF
*
                EXTRN      MMGR
                EXTRN      SXPAGE
                EXTRN      CZPAGE
                EXTRN      CXPAGE
*
                EXTRN      SYSERR
                EXTRN      SERR
                EXTRN      OUTOFMEM
                EXTRN      BUFTBLFULL
                EXTRN      BADSYSBUF
```

```
*
                EXTRN      SYSDEATH
                EXTRN      BADBUFNUM
                EXTRN      BADBUFSIZ
*
                ENTRY      BUF.CNT
                ENTRY      PGCT.T
                ENTRY      XBYTE.T
                ENTRY      BUFREF
                PAGE
                REP        60
*
* DATA DECLARATIONS
*
                REP        60
*
Z.REG           EQU        $FFD0
*
* MEMORY MGMT CALL PARM LOCATIONS ON SOS ZPAGE
*
M.TPARMX        EQU        $60                  ; FIRST ADR OF MEM SYS CALL PARMS ON SOS ZPAGE
REQCODE         EQU        M.TPARMX+$0
*
FINDSEG         EQU        $1
SRCHMODE        EQU        M.TPARMX+$1
F.ID            EQU        M.TPARMX+$2
F.PGCT          EQU        M.TPARMX+$3
F.PGCTX         DS         2                    ; TEMP LOC FOR F.PGCT PARM
F.BASE          EQU        M.TPARMX+$5
F.BASEX         DS         2                    ; TEMP LOC FOR F.BASE PARM
F.LIM           EQU        M.TPARMX+$7
F.LIMX          DS         2                    ; TEMP LOC FOR F.LIM PARM
F.NUM           EQU        M.TPARMX+$9
F.NUMX          DS         1                    ; TEMP LOC FOR F.NUM PARM
*
RELSEG          EQU        $5
RLS.NUM         EQU        M.TPARMX+$1
*
* REQBUF DATA DECLARATIONS
*
RQB.PGCT        DS         1                    ; REQUESTED PAGE COUNT
RQB.BNUM        DS         1                    ; BUFFER NUMBER (FM GETFREE CALL)
*
* REQFXBUF DATA DECLARATIONS
*
RQFB.PGCT       DS         1                    ; REQUESTED PAGE COUNT
RQFB.BNUM       DS         1                    ; BUFFER NUMBER (FM GETFREE CALL)
MAXPGCT         EQU        64                   ; MAX BUFSIZE=16K
F.TPARMX        EQU        $A0                  ; FIRST ADR OF FILE SYS CALL PARMS ON SOS ZPAGE
OPEN.LIST       EQU        F.TPARMX+$5          ; LOC OF OPEN.LIST PARM (OPEN SYS CALL)
```

```
*
* BUFCOMPACT DATA DECLARATIONS (SOURCE ALSO USED BY CHKBUF)
*
BUFC.BNUM       DS         1                    ; BUF# OF LOWEST BUFFER IN BUF.TBL
SOURCE          EQU        M.TPARMX+$10         ; & $11
DEST            EQU        M.TPARMX+$12         ; & $13
                PAGE
                REP        60
*
* BUFFER TABLE
*
* THE BUFFER TABLE CONSISTS OF "CNT"-1 ENTRIES (1 TO "CNT"-1).
* EACH ENTRY IS "SIZ" BYTES IN LENGTH.  THE "PGCT" FIELD
* CONTAINS 3 SUBFIELDS.  BIT 7 IS THE "FREE" FLAG (0=ACTIVE,1=FREE)
* BIT 6 IS THE "FIXED" FLAG (0=FLOATING BUFFER,1=FIXED BUFFER)
* BITS 5 THRU 0 CONTAIN THE PAGE COUNT OF AN "ACTIVE" ENTRY
* (0=>1 PAGE,63=>64 PAGES DECIMAL).  THE "XBYTE" FIELD CONTAINS
* THE PROPER XBYTE OF AN "ACTIVE" ENTRY.  THE "ADRH" FIELD
* CONTAINS THE HIGH BYTE OF THE BUFFER ADDRESS.  IF THE
* BUFFER ENTRY IS "FLOATING", THEN THE "SEG" FIELD CONTAINS THE
* SEGMENT NUMBER AND THE LOW BYTE OF THE BUFFER ADDRESS IS
* ASSUMMED TO BE ZERO.
*
* THUS, THE FOLLOWING RESTRICTIONS APPLY TO BUFFERS:
*
* (1) MAXIMUM BUFFER LENGTH IS 64 PAGES (16K)
* (2) "FLOATING" BUFFERS ALWAYS BEGIN ON A PAGE BOUNDARY
*     "FIXED" BUFFERS DO NOT.
* (3) BUFFERS ARE ALWAYS AN INTEGRAL NUMBER OF PAGES IN LENGTH
* (4) BUFFERS ALWAYS RESIDE IN THE 32K BANK MEMORY REGION,
*     A LIMITATION OF FIND.SEG (MEMORY MANAGER)
* (5) MAXIMUM NUMBER OF BUFFERS = 16; ENTRY 0 IS NOT USED.
*
                REP        60
*
* BUFFER TABLE
*
BUF.SIZ         EQU        5
BUF.CNT         EQU        17
BUF.TBL         DS         BUF.SIZ*BUF.CNT
PGCT.T          EQU        BUF.TBL
XBYTE.T         EQU        PGCT.T+BUF.CNT
ADRH.T          EQU        XBYTE.T+BUF.CNT
SEG.T           EQU        ADRH.T+BUF.CNT
ADRL.T          EQU        SEG.T
CHK.T           EQU        ADRL.T+BUF.CNT
ISFIXED         EQU        $40
ISFREE          EQU        $80
*
* BUFFER REFERENCE TABLE
```

```
*
* FIRST BYTE IS COUNT, FOLLOWED BY "COUNT" BUFFER #S.
* THIS TABLE IS A LIST OF ALL BUFFERS REFERENCED DURING ONE
* SOS SYSTEM CALL.  BUFFER #S ARE ADDED TO THIS LIST BY
* GETBUFADR AND REMOVED BY CHKSUM.
*
BUFREF.CNT      EQU        17
BUFREF          DS         BUFREF.CNT
ZPAGEX          DS         1
                PAGE
                REP        60
*
* REQBUF
*
* INPUT:  PAGE.CNT (A)
* OUTPUT: BUFNUM   (A)
* ERROR:  "BUFFER TABLE FULL" - SYSERR
*         "OUT OF MEMORY"     - SYSERR
*         "BAD BUFFER SIZE"   - SYSDEATH
*
* THIS ROUTINE FINDS A FREE ENTRY IN THE BUFFER TABLE
* AND THEN CALLS FIND.SEG (MMGR) TO OBTAIN MEMORY FOR IT.
* IF MEMORY IS FOUND THEN THE BUFFER ENTRY IS MARKED "ACTIVE"
* AND THE BUFFER INFO IS INSERTED INTO THE ENTRY
*
                REP        60
*
REQBUF          EQU        *
*
* IF REQUESTED PGCT OUT OF BOUNDS THEN FATAL ERR
*
                TAY
                BEQ        RQB.ERR2             ; FATAL ERR, INVALID BUFFER SIZE
                CPY        #MAXPGCT+1
                BCS        RQB.ERR2             ; FATAL ERR, INVALID BUFFER SIZE
                STY        RQB.PGCT             ; SAVE PAGE COUNT
*
* FIND FREE ENTRY IN BUF.TBL
*
                JSR        GETFREE
                BCS        RQB.ERR              ; ERR, BUFFER TABLE FULL
                STX        RQB.BNUM
*
* FIND PGCT*256 BYTES OF FREE MEMORY
*
                LDA        RQB.PGCT
                JSR        FSEG
                BCS        RQB.ERR1             ; ERR, OUT OF MEMORY
*
* INSERT PGCT, XBYTE, ADRH, SEG#, CHK BYTE IN BUF.TBL(BUF#)
```

```
*
                LDX        RQB.BNUM
                DEC        RQB.PGCT             ; PAGE COUNT FIELD
                LDA        RQB.PGCT
                STA        PGCT.T,X
*
                LDX        F.BASEX              ; XBYTE & ADRH FIELDS
                LDY        F.BASEX+1
                JSR        CNVRT.ADR
                CPX        #$8F
                BNE        RQB010
                LDX        #$7F                 ; IF XBYTE=$8F THEN XBYTE:=$7F
RQB010          TXA
                LDX        RQB.BNUM
                STA        XBYTE.T,X
                TYA
                STA        ADRH.T,X
*
                LDA        F.NUMX               ; SEG# FIELD
                STA        SEG.T,X
*
                LDA        #0                   ; INIT CHECK BYTE TO NULL
                STA        CHK.T,X
*
                TXA                             ; RETURN BUF#
                CLC
                RTS                             ; NORMAL EXIT
*
*
RQB.ERR         LDA        #BUFTBLFULL
                JSR        SYSERR
*
RQB.ERR1        LDA        #OUTOFMEM
                JSR        SYSERR
*
RQB.ERR2        LDA        #BADBUFSIZ
                JSR        SYSDEATH
                PAGE
                REP        60
*
* REQFXBUF
*
* INPUT:  PAGE.CNT (A)
* OUTPUT: BUFNUM   (A)
* ERROR:  "BUFFER TABLE FULL"            - SYSERR
*         "BAD SYSTEM.BUF PARM ADDRESS"  - SYSERR
*         "BAD BUFFER SIZE"              - SYSDEATH
*
* THIS ROUTINE COMPUTES THE ACTUAL BUFFER ADDRESS IN THE OPEN
* CALL (PARM "OPEN.LIST"), AND ALLOCATES A BUFFER ENTRY FOR IT.
* NOTE:  THE SYSBUF PARAMETER MUST BE AN EXTENDED INDIRECT PTR!!
*
                REP        60
*
REQFXBUF        EQU        *
*
* IF REQUESTED PGCT OUT OF BOUNDS THEN FATAL ERR
*
                TAY
                BEQ        RQFB.ERR2            ; FATAL ERR, BAD BUFFER SIZE
                CPY        #MAXPGCT+1
                BCS        RQFB.ERR2            ; FATAL ERR, BAD BUFFER SIZE
*
                STY        RQFB.PGCT            ; SAVE PAGE COUNT
*
* GET A FREE BUFFER ENTRY
*
                JSR        GETFREE
                BCS        RQFB.ERR             ; ERR, BUFFER TABLE FULL
                STX        RQFB.BNUM            ; SAVE BUF#
*
* FETCH SYSTEM.BUF PARAMETER IN OPEN SYSTEM CALL
*
                LDY        #3
                LDA        (OPEN.LIST),Y
                BNE        RQFB.ERR1            ; ERR, SYSBUF ADR
                DEY
                LDA        (OPEN.LIST),Y
                TAY
                LDA        CXPAGE+1,Y
                BPL        RQFB.ERR1            ; ERR, SYSBUF ADR
                CMP        #$8F
                BCS        RQFB.ERR1            ; ERR, SYSBUF ADR
*
* INSERT XBYTE, ADRH, ADRL, PGCT, CHK BYTE INTO BUF.TBL(BUF#)
*
                LDX        RQFB.BNUM
                STA        XBYTE.T,X
*
                LDA        CZPAGE+1,Y
                BEQ        RQFB.ERR1            ; ERR SYSBUF ADR
                CMP        #$81                 ; CHECK FOR ADDRESS COMPENSATION
                BCC        RQFB010
                INC        XBYTE.T,X
                AND        #$7F
RQFB010         STA        ADRH.T,X
*
                LDA        CZPAGE,Y
                STA        ADRL.T,X
*
```

```
                DEC        RQFB.PGCT
                LDA        RQFB.PGCT
                ORA        #ISFIXED
                STA        PGCT.T,X             ; BUFFER ENTRY NOW "ACTIVE"
*
                LDA        #0                   ; INIT CHECK BYTE TO NULL
                STA        CHK.T,X
*
                TXA                             ; RETURN BUF#
                CLC
                RTS                             ; NORMAL EXIT
*
RQFB.ERR        LDA        #BUFTBLFULL
                JSR        SYSERR
*
RQFB.ERR1       LDA        #BADSYSBUF
                JSR        SYSERR
*
RQFB.ERR2       LDA        #BADBUFSIZ
                JSR        SYSDEATH
                PAGE
                REP        60
*
* GETBUFADR
*
* INPUT:  BUFNUM   (A)
*         ZPAGELOC (X)
* OUTPUT: BUF ADR AT: X,X+1 & SXPAGE+1,X
*         PAGE.CNT (A)
*         BUFNUM   (Y)
*
* ERROR:  "BADBUFNUM" SYSDEATH
*
                REP        60
*
GETBUFADR       EQU        *
*
* IF BUF# OUT OF RANGE OR BUF.TBL(BUF#)=FREE
* THEN FATAL ERR
*
                TAY
                BEQ        GTBF.ERR             ; BUF#=0, FATAL ERR
                CPY        #BUF.CNT
                BCS        GTBF.ERR             ; BUF# > MAX BUF TABLE ENTRY, FATAL ERR
                LDA        PGCT.T,Y
                BMI        GTBF.ERR             ; BUF ENTRY MARKED "FREE", FATAL ERR
*
* OTHERWISE, CONSTRUCT BUFFER PTR ON SOS ZPAGE
*
                JSR        GETBUFADR1
```

```
*
* IF BUFFER NOT PREVIOUSLY REFERENCED ON THIS SOS CALL AND CHECK BYTE <> 0
*    THEN COMPARE FIRST BYTE OF BUFFER WITH CHECK BYTE IN BUFFER TABLE.
*         IF NO MATCH THEN KILL SYSTEM.
*
                STX        ZPAGEX
                TYA
                LDX        BUFREF
                BEQ        GTBF020              ; BUFREF EMPTY
*
GTBF010         CMP        BUFREF,X             ; SEARCH FOR PREVIOUS REFERENCE
                BEQ        GTBF030              ; MATCH FOUND
                DEX
                BNE        GTBF010
*
GTBF020         INC        BUFREF               ; LOG BUF # IN BUFREF TABLE
                LDX        BUFREF
                CPX        #BUFREF.CNT
                BCS        GTBF.ERR             ; BUFREF TABLE OVFLOW, KILL SYSTEM
                STA        BUFREF,X
*
                LDA        CHK.T,Y
                BEQ        GTBF030              ; NO CHECK BYTE, SKIP CHECK
                LDX        ZPAGEX
                LDA        ($0,X)               ; COMPARE FIRST BYTE OF BUFFER
                CMP        CHK.T,Y              ; WITH CHECK BYTE IN BUF TABLE
                BNE        GTBF.ERR             ; NO MATCH, PULL THE PLUG
*
* RETURN PAGE.CNT TO CALLER
*
GTBF030         LDA        PGCT.T,Y
                AND        #$3F                 ; STRIP OFF FREE,FIXED FLAGS
                CLC
                ADC        #1
*
                CLC
                RTS
*
*
GTBF.ERR        LDA        #BADBUFNUM
                JSR        SYSDEATH
*
*
                REP        60
*
* GETBUFADR1
*
* INPUT: PGCT.T(BUF#)  (A)
*        ZPAGELOC      (X)
*        BUF#          (Y)
* ERROR: NONE.
*
* EXTRACTS THE BUFFER POINTER FROM THE BUFFER TABLE AND
* PLACES IT ON ZERO PAGE AT X, X+1 & SXPAGE+1,X
*
                REP        60
*
GETBUFADR1      EQU        *
                AND        #$40
                BNE        GTB1010
                LDA        #0                    ; "FIXED" BUFFER
                BEQ        GTB1020               ; ALWAYS TAKEN
GTB1010         LDA        ADRL.T,Y              ; "FLOATING" BUFFER
GTB1020         STA        0,X
                LDA        ADRH.T,Y
                STA        1,X
                LDA        XBYTE.T,Y
                ORA        #$80                  ; ENSURE $7F->$8F
                STA        SXPAGE+1,X
                RTS
                PAGE
                REP        60
*
* CHKBUF
*
* CHECK BUFFER.  FETCHES THE FIRST BYTE OF EACH BUFFER
* REFERENCED DURING THE CURRENT SYSTEM CALL AND PLACES IT
* IN CHK.T(BUF#).
*
* INPUT:  BUFREF TABLE
*         BUFFER TABLE
* OUTPUT: EMPTY BUFREF TABLE
*         BUFFER TABLE'S CHECK BYTES UPDATED
*         Z REG:=$18
* ERROR:  NONE.
*
                REP        60
*
CHKBUF          EQU        *
                LDY        BUFREF                ; PICK UP COUNT
                BEQ        CHKB.EXIT             ; EXIT IF BUFREF EMPTY
*
                LDA        #$18                  ; ENSURE SOS ZPAGE SWITCHED IN
                STA        Z.REG
*
* UPDATE THE CHECK BYTE OF EACH BUF# IN THE BUFREF TABLE
*
CHKB010         LDX        #>SOURCE
                LDA        BUFREF,Y
                TAY
                LDA        PGCT.T,Y
                JSR        GETBUFADR1           ; PUT BUF#S ADR ON ZPAGE
                LDA        ($0,X)
                STA        CHK.T,Y
                DEC        BUFREF
                LDY        BUFREF
                BNE        CHKB010              ; IF COUNT<>0 THEN PROCESS NEXT BUF# IN BUFREF TABLE
*
CHKB.EXIT       RTS                             ; BUFREF TABLE IS EMPTY (COUNT=0)
                PAGE
                REP        60
*
* RELBUF
*
* INPUT:  BUFNUM   (A)
* OUTPUT: NONE.
* ERROR:  "BADBUFNUM" SYSDEATH
*
* THIS ROUTINE RELEASES THE BUFFER ENTRY, CALLS FIND.SEG TO
* RELEASE THE CORRESPONDING MEMORY SEGMENT, AND CALLS
* BUFCOMPACT TO PERFORM BUFFER COMPACTION.
*
                REP        60
*
RELBUF          EQU        *
*
* IF BUF# OUT OF RANGE OR BUF.TBL(BUF#)=FREE
* THEN FATAL ERR
*
                TAY
                BEQ        RLBF.ERR
                CPY        #BUF.CNT
                BCS        RLBF.ERR
                LDA        PGCT.T,Y
                BMI        RLBF.ERR
*
* MARK BUF.TBL(BUF#)=FREE
*
                ORA        #ISFREE
                STA        PGCT.T,Y
*
* IF BUF.TBL(BUF#)=FIXED THEN EXIT
*
                AND        #ISFIXED
                BNE        RLBF.EXIT
*
* OTHERWISE CALL MEMORY MGR TO RELEASE BUFFER'S MEMORY SEG
*
                LDA        #RELSEG
                STA        REQCODE
```

```
*
                LDA        SEG.T,Y
                STA        RLS.NUM
*
                JSR        MMGR
                BCS        RLBF.ERR             ; ANY ERR IS FATAL
*
* AND COMPACT BUFFERS
*
                JSR        BUFCOMPACT
*
RLBF.EXIT       CLC
                RTS
*
RLBF.ERR        LDA        #BADBUFNUM
                JSR        SYSDEATH
                PAGE
                REP        60
*
* BUFCOMPACT
*
* THIS ROUTINE IS RESPONSIBLE FOR PACKING ALL SOS BUFFERS UP
* AGAINST THE HIGHEST AVAILABLE FREE MEMORY.  COULD IMPROVE THE
* EFFICIENCY OF THIS COMPACTION CYCLE BY NOT RELEASING THE "RELEASED" BUFFER
* UNTIL IT IS KNOWN THAT ANOTHER BUFFER WILL NOT BE MOVED INTO ITS LOC.
*
                REP        60
*
BUFCOMPACT      EQU        *
*
* FIND THE FLOATING BUFFER IN BUF.TBL WITH THE LOWEST ADDRESS.
*
BUFC010         LDY        #0
                LDX        #BUF.CNT-1
*
BUFC020         LDA        PGCT.T,X
                AND        #$C0                 ; STRIP OUT PAGE COUNT BITS
                BNE        BUFC030
*
                LDA        ADRH.T,X
                CMP        ADRH.T,Y
                LDA        XBYTE.T,X
                SBC        XBYTE.T,Y
                BCS        BUFC030
*
                TXA                             ; SMALLER BUFFER FOUND, SAVE IN Y
                TAY
*
BUFC030         DEX
                BNE        BUFC020
```

```
*
* IF NO BUFFER FOUND THEN DONE
*
                TYA
                BNE        BUFC040
                JMP        BUFC.EXIT
BUFC040         STY        BUFC.BNUM            ; OTHERWISE SAVE BUF# IN Y REG.
*
* CALL FIND.SEG:  FINDS HIGHEST AVAILABLE FREE MEMORY
*
                LDA        PGCT.T,Y
                AND        #$3F                 ; STRIP OUT "FREE","FIXED" FLAGS
                CLC
                ADC        #1
                JSR        FSEG
                BCS        BUFC.EXIT            ; DONE IF NO FREE SEG FOUND
*
* CONVERT BASE.BKPG TO BUFFER ADR
*
                LDX        F.BASEX              ; BASE BANK
                LDY        F.BASEX+1            ; BASE PAGE
                JSR        CNVRT.ADR
                STX        F.BASEX              ; XBYTE
                STY        F.BASEX+1            ; ADRH
*
* IF NEW SEG'S BASE < CURRENT BUFFER'S BASE ADR THEN DONE
*
                LDY        BUFC.BNUM
                LDA        ADRH.T,Y
                STA        SOURCE+1
                CMP        F.BASEX+1
                LDA        XBYTE.T,Y
                STA        SXPAGE+SOURCE+1
                SBC        F.BASEX
                BCS        BUFC.EXIT1
*
* MOVE DATA FROM CURRENT BUFFER TO NEW BUFFER
*
                LDX        F.BASEX
                STX        SXPAGE+DEST+1
                LDY        F.BASEX+1
                STY        DEST+1
                LDA        #0
                STA        SOURCE
                STA        DEST
*
                TAY
                LDX        F.PGCTX
BUFC200         LDA        (SOURCE),Y           ; MOVE LOOP
                STA        (DEST),Y
                DEY
                BNE        BUFC200
                INC        SOURCE+1
                INC        DEST+1
                DEX
                BNE        BUFC200
*
* UPDATE BUF.TBL(BUF#)
*
                LDY        BUFC.BNUM
                LDA        F.BASEX
                STA        XBYTE.T,Y
                LDA        F.BASEX+1
                STA        ADRH.T,Y
*
                LDX        SEG.T,Y
                LDA        F.NUMX
                STA        SEG.T,Y
*
* AND RELEASE OLD MEMORY SEGMENT
*
                STX        RLS.NUM
                LDA        #RELSEG
                STA        REQCODE
                JSR        MMGR
                BCS        BUFC.ERR
*
                JMP        BUFC010             ; REPEAT COMPACTION CYCLE
*
*
BUFC.EXIT1      LDX        F.NUMX              ; DONE,
                STX        RLS.NUM             ; RELEASE SEG BEFORE EXIT
                LDA        #RELSEG
                STA        REQCODE
                JSR        MMGR
                BCS        BUFC.ERR
*
BUFC.EXIT       LDA        #0
                STA        SERR                ; MASK OUT ANY ERROR FROM MEMORY MGR
                CLC
                RTS                            ; NORMAL EXIT
*
*
BUFC.ERR        LDA        #BADBUFNUM
                JSR        SYSDEATH
                PAGE
                REP        60
*
* FSEG
*
```

```
* INPUT:  PAGE.CNT (A)
* OUTPUT: PAGE.CNT (A) UNCHANGED IF FIND.SEG SUCCESSFUL
* ERROR:  CARRY SET "UNABLE TO FIND MEMORY SEG OF PAGE.CNT*256 BYTES"
*
* THIS ROUTINE BUILDS THE PARAMETERS FOR A FIND.SEG SYSTEM CALL
* AND THEN CALLS THE MEMORY MANAGER.
*
                REP        60
*
FSEG            EQU        *
*
* SETUP INPUT PARAMETERS FOR FIND.SEG CALL
*
                STA        F.PGCTX
                LDA        #FINDSEG
                STA        REQCODE
                LDA        #2
                STA        SRCHMODE
                LDA        #4
                STA        F.ID
*
* SETUP OUTPUT PARAMETER ADRESSES
*
                LDA        #>F.PGCTX
                STA        F.PGCT
                LDA        #<F.PGCTX
                STA        F.PGCT+1
                LDA        #>F.BASEX
                STA        F.BASE
                LDA        #<F.BASEX
                STA        F.BASE+1
                LDA        #>F.LIMX
                STA        F.LIM
                LDA        #<F.LIMX
                STA        F.LIM+1
                LDA        #>F.NUMX
                STA        F.NUM
                LDA        #<F.NUMX
                STA        F.NUM+1
*
                LDA        #0
                STA        F.PGCTX+1
                STA        SXPAGE+F.PGCT+1
                STA        SXPAGE+F.BASE+1
                STA        SXPAGE+F.LIM+1
                STA        SXPAGE+F.NUM+1
*
                JSR        MMGR
                LDA        F.PGCTX
*
```

```
                RTS                                ; EXIT.  CARRY SET->ERR
                PAGE
                REP        60
*
* GETFREE
*
* INPUT:  NONE
* OUTPUT: BUF# (X)
* ERROR:  "BUFTBLFULL" SYSERR
*
* THIS ROUTINE SEARCHES THE BUFFER TABLE, LOOKING FOR A FREE
* ENTRY.  IF FOUND, IT RETURNS THE BUFFER NUMBER, ELSE ERROR.
*
                REP        60
*
GETFREE         EQU        *
                LDX        #BUF.CNT-1
GFR010          LDA        PGCT.T,X
                BMI        GFR.EXIT                ; FREE ENTRY FOUND
                DEX
                BNE        GFR010
*
                LDA        #BUFTBLFULL
                JSR        SYSERR                  ; ERR EXIT
*
GFR.EXIT        CLC
                RTS                                ; NORMAL EXIT
                PAGE
                REP        60
*
* CNVRT.ADR
*
* INPUT:  BANK VALUE (X)
*         PAGE VALUE (Y)
* OUTPUT: XBYTE (X)
*         ADRH  (Y)
* ERROR:  NONE.
*
* THIS ROUTINE CONVERTS A BASE.BKPG PARM (MMGR) INTO A
* VIRTUAL POINTER
*
                REP        60
*
CNVRT.ADR       EQU        *
*
* IF PAGE <> $20 THEN GOTO L2
*
                CPY        #$20
                BNE        CNVA020
*
```

```
* IF BANK <> 0 THEN GOTO L1
*
                TXA
                BNE        CNVA010
*
* XBYTE=$8F
* ADRH:=PAGE
*
                LDX        #$8F
                BMI        CNVA.EXIT
*
* L1: XBYTE:=(BANK-1) ORA #$80
*     ADRH:=#$80
*
CNVA010         ORA        #$80
                TAX
                DEX
                LDY        #$80
                BMI        CNVA.EXIT
*
* L2: XBYTE:=BANK ORA #$80
*     ADRH:=ADRH-#$20
*
CNVA020         TXA
                ORA        #$80
                TAX
                SEC
                TYA
                SBC        #$20
                TAY
*
CNVA.EXIT       RTS
*
                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENBUFMG
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR BUFMGR"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: BUFMGR.SRC
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.CRMGR.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: CFMGR.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1  CHARACTER FILE MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGCFM
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* CHARACTER FILE MANAGER (VERSION = 1.10   )
*                        (DATE    = 8/04/81)
*
* THIS MODULE TRANSFORMS CHARACTER FILE SYSTEM CALLS INTO
* DEVICE CALLS TO THE APPROPRIATE DEVICE HANDLER.  ONLY
* OPEN, NEWLINE, READ, WRITE AND CLOSE CALLS ARE PERMITTED
* ON CHARACTER FILES.
*
                REP        60
*
                ENTRY      CFMGR
*
                ENTRY      CFCB.MAX
                ENTRY      CFCB.DEV
*
                EXTRN      DMGR
                EXTRN      LEVEL
                EXTRN      MAX.DNUM
                EXTRN      SXPAGE
*
                EXTRN      SYSERR
                EXTRN      SERR
                EXTRN      BADSCNUM
                EXTRN      CFCBFULL
                EXTRN      BADREFNUM
                EXTRN      FNFERR
                PAGE
                REP        60
*
```

```
* DATA DECLARATIONS
*
                REP        60
*
* FILE CALL PARM LOCATIONS ON SOS ZPAGE
*
F.TPARMX        EQU        $A0
REQCODE         EQU        F.TPARMX
O.PATH          EQU        F.TPARMX+1           ; OPEN'S PATHNAME LOC
O.REFNUM        EQU        F.TPARMX+3           ; OPEN'S REFNUM LOC
REFNUM          EQU        F.TPARMX+1           ; REFNUM'S LOC IN OTHER CALLS
NL.ISNL         EQU        F.TPARMX+2           ; NEWLINE'S ISNEWLINE LOC
NL.NLCHR        EQU        F.TPARMX+3           ; NEWLINE'S NEWLINECHAR LOC
RW.BUF          EQU        F.TPARMX+2           ; READ/WRITE'S BUF LOC
RW.BYTES        EQU        F.TPARMX+4           ; READ/WRITE'S BYTES LOC
RD.BYTESRD      EQU        F.TPARMX+6           ; READ'S BYTESREAD LOC
*
* FILE REQUEST CODE VALUES
*
OPEN            EQU        8
NEWLINE         EQU        9
READ            EQU        $A
WRITE           EQU        $B
CLOSE           EQU        $C
                PAGE
* DEVICE CALL PARM LOCATIONS ON SOS ZPAGE
*
D.TPARMX        EQU        $C0
D.SCNUM         EQU        D.TPARMX             ; DEVICE SYS CALL # LOC
GDN.DNAME       EQU        D.TPARMX+1           ; GETDEVNUM DNAME LOC
GDN.DNUM        EQU        D.TPARMX+3           ; GETDEVNUM DNUM LOC
D.DNUM          EQU        D.TPARMX+1           ; OPN/CLOSE/RD/WR/CTRL'S DNUM LOC
DRW.BUF         EQU        D.TPARMX+2           ; RD/WR'S BUF LOC
DRW.BYTES       EQU        D.TPARMX+4           ; RD/WR'S BYTES LOC
DRD.BYTESRD     EQU        D.TPARMX+8           ; RD/WR'S BYTESREAD LOC
DC.CCODE        EQU        D.TPARMX+2           ; DCTRL'S CTRLCODE LOC
DC.CLIST        EQU        D.TPARMX+3           ; DCTRL'S CTRLLIST LOC
*
* DEVICE REQUEST CODE VALUES
*
DREAD           EQU        $0
DWRITE          EQU        $1
DCTRL           EQU        $3
GETDEVNUM       EQU        $4
DOPEN           EQU        $6
DCLOSE          EQU        $7
*
CTRL.LIST       DS         2                    ; CONTAINER FOR NEWLINE DCTRL CALL
NEWLINECC       EQU        2                    ; NEWLINE CTRL CODE
*
```

```
* GETDNUM VARS
*
DNUM.TEMP       DS         1
*
* CLOSEALL VARS
*
DCLOSE.ERR      EQU        F.TPARMX+$F
DCLOSE.TBL      EQU        $200
TRUE            EQU        $80
FALSE           EQU        $0
*
*
                REP        60
*
* CHARACTER FILE CONTROL BLOCK TABLE
* (ENTRY 0 IS NOT USED)
*
                REP        60
CFCB.MAX        EQU        17
CFCB.DEV        DS         CFCB.MAX
CFCB.LVL        DS         CFCB.MAX
                PAGE
                REP        60
*
* CHARACTER FILE MANAGER - MAIN ENTRY POINT
*
                REP        60
CFMGR           EQU        *
*
* SWITCH, BASED ON REQUEST CODE
*
                LDA        REQCODE
                CMP        #OPEN
                BEQ        CFOPEN                 ; "OPEN"
                CMP        #NEWLINE
                BEQ        CFNEWLINE              ; "NEWLINE"
                CMP        #READ
                BEQ        CFREAD                 ; "READ"
                CMP        #WRITE
                BNE        CFM010
                JMP        CFWRITE                ; "WRITE"
CFM010          CMP        #CLOSE
                BNE        CFM020
                JMP        CFCLOSE                ; "CLOSE"
CFM020          LDA        #BADSCNUM
                JSR        SYSERR                 ; ERR EXIT
                PAGE
                REP        60
* OPEN(IN.PATHNAME; OUT.REFNUM; IN.OPENLIST,LENGTH) SYSTEM CALL
                REP        60
```

```
CFOPEN          EQU        *                    ; BUILD "D.OPEN" CALL
                JSR        GETDNUM              ; MAP PATH TO DEV#
                BCS        CFOP.ERR1            ; ERR - FILE NOT FOUND
                STA        D.DNUM
*
                JSR        REQ.CFCB             ; BUILD NEW CFCB ENTRY
                BCS        CFOP.ERR1            ; ERR - CFCB FULL
                LDX        #0
                STA        (O.REFNUM,X)         ; RETURN REFNUM TO CALLER
                CPY        #1
                BNE        CFOP.EXIT            ; DEVICE ALREADY OPEN
*
                LDA        #DOPEN
                STA        D.SCNUM
                JSR        DMGR                 ; DOPEN CALL
                BCS        CFOP.ERR
CFOP.EXIT       RTS                             ; NORMAL EXIT
*
CFOP.ERR        LDA        SERR                 ;KLUDGE - 1.0 DRIVERS DON'T SUPPORT CARRY ERR PROTOCOL
                BEQ        CFOP.EXIT            ;NO ERROR
                LDX        #0                   ; RELEASE CFCB ENTRY
                LDA        (O.REFNUM,X)
                JSR        REL.CFCB
CFOP.ERR1       RTS                             ; ERR EXIT
                PAGE
                REP        60
* NEWLINE(IN.REFNUM,IS        .NEWLINE,NEWLINE.CHAR) SYSTEM CALL
                REP        60
CFNEWLINE       EQU        *                    ; BUILD "D.CONTROL" CALL
                LDA        #DCTRL
                STA        D.SCNUM
                LDA        REFNUM
                JSR        GET.CFCB             ; MAP REFNUM TO DEV #
                BCS        CFNL.ERR             ; ERR - BAD REFNUM
*
                STA        D.DNUM
                LDA        #NEWLINECC
                STA        DC.CCODE
*
                LDA        #>CTRL.LIST
                STA        DC.CLIST
                LDA        #<CTRL.LIST
                STA        DC.CLIST+1
                LDA        #0
                STA        SXPAGE+DC.CLIST+1
*
                LDA        NL.ISNL
                STA        CTRL.LIST
                LDA        NL.NLCHR
                STA        CTRL.LIST+1
```

```
*
                JSR        DMGR                    ; DCONTROL CALL
                RTS                                ; NORMAL EXIT
*
CFNL.ERR        RTS                                ; ERR EXIT
                PAGE
                REP        60
* READ(IN.REFNUM,BUF,BYTES,BYTESREAD) SYSTEM CALL
                REP        60
CFREAD          EQU        *                       ; BUILD "D.READ" CALL
                LDA        #DREAD
                STA        D.SCNUM
                LDA        REFNUM
                JSR        GET.CFCB                ; MAP REFNUM TO DEV #
                BCS        CFRD.ERR                ; ERR - BAD REFNUM
*
                STA        D.DNUM
                LDX        #3
CFRD010         LDA        RW.BUF,X
                STA        DRW.BUF,X
                DEX
                BPL        CFRD010
*
                LDA        RD.BYTESRD
                STA        DRD.BYTESRD
                LDA        RD.BYTESRD+1
                STA        DRD.BYTESRD+1
*
                LDA        SXPAGE+RW.BUF+1
                STA        SXPAGE+DRW.BUF+1
                LDA        SXPAGE+RW.BYTES+1
                STA        SXPAGE+DRW.BYTES+1
                LDA        SXPAGE+RD.BYTESRD+1
                STA        SXPAGE+DRD.BYTESRD+1
*
                JSR        DMGR                    ; DREAD CALL
                RTS                                ; NORMAL EXIT
*
CFRD.ERR        RTS                                ; ERR EXIT
                PAGE
                REP        60
* WRITE(IN.REFNUM,BUF,BYTES) SYSTEM CALL
                REP        60
CFWRITE         EQU        *                       ; BUILD "D.WRITE" CALL
                LDA        #DWRITE
                STA        D.SCNUM
                LDA        REFNUM
                JSR        GET.CFCB                ; MAP REFNUM TO DEV #
                BCS        CFWR.ERR                ; ERR - BAD REFNUM
                STA        D.DNUM
                LDX        #3
CFWR010         LDA        RW.BUF,X
                STA        DRW.BUF,X
                DEX
                BPL        CFWR010
                LDA        SXPAGE+RW.BUF+1
                STA        SXPAGE+DRW.BUF+1
                LDA        SXPAGE+RW.BYTES+1
                STA        SXPAGE+DRW.BYTES+1
*
                JSR        DMGR                 ; DWRITE CALL
                RTS                             ; NORMAL EXIT
*
CFWR.ERR        RTS                             ; ERR EXIT
                PAGE
                REP        60
* CLOSE(IN.REFNUM) SYSTEM CALL
                REP        60
CFCLOSE         EQU        *                    ; BUILD "D.CLOSE" CALL
                LDA        #DCLOSE
                STA        D.SCNUM
                LDA        REFNUM
                BEQ        CLOSEALL
*
                JSR        REL.CFCB             ; RELEASE CFCB ENTRY
                BCS        CFCL010
                STA        D.DNUM
                TYA
                BNE        CFCL010
                JSR        DMGR                 ; DCLOSE CALL
CFCL010         RTS                             ; NORMAL EXIT
*
                PAGE
                REP        60
*
* CLOSE ALL CHARACTER FILES W/LEVELS >= TO CURRENT SYSTEM FILE LEVEL.
*
                REP        60
*
CLOSEALL        EQU        *
                LDA        #FALSE               ; SET ENTRIES IN DEV CLOSE TBL TO FALSE
                LDX        MAX.DNUM
CFCL020         STA        DCLOSE.TBL,X
                DEX
                BPL        CFCL020
*
                LDX        #CFCB.MAX-1          ; CLOSE ALL DEVICES >= TO CURRENT LEVEL
CFCL030         LDA        CFCB.DEV,X           ; AND MARK TRUE IN DEV CLOSE TBL
                TAY
                BMI        CFCL050
                LDA        CFCB.LVL,X
                CMP        LEVEL
                BCC        CFCL050
                LDA        #TRUE
                STA        DCLOSE.TBL,Y
                SEC
                ROR        CFCB.DEV,X
CFCL050         DEX
                BNE        CFCL030
*
                LDX        #CFCB.MAX-1          ; DON'T CLOSE DEVICES < CURRENT LEVEL
CFCL060         LDA        CFCB.DEV,X
                TAY
                BMI        CFCL070
                LDA        #FALSE
                STA        DCLOSE.TBL,Y
CFCL070         DEX
                BNE        CFCL060
*
                LDA        #0
                STA        DCLOSE.ERR
                LDX        MAX.DNUM             ; ISSUE D'CLOSE CALLS TO ALL DEVICES MARKED AS TRUE
CFCL080         LDA        DCLOSE.TBL,X         ; IN DEV CLOSE TABLE
                BPL        CFCL090
                TXA
                PHA
                STX        D.DNUM
                JSR        DMGR
                PLA
                TAX
                LDA        SERR
                BEQ        CFCL090              ; IF ERROR,
                STA        DCLOSE.ERR           ; THEN SAVE IT
CFCL090         DEX
                BNE        CFCL080
*
                LDA        DCLOSE.ERR           ; IF $0 THEN NO ERRORS FROM D.CLOSE CALLS
                BNE        CFCL.ERR
                RTS                             ; NORMAL EXIT
CFCL.ERR        JSR        SYSERR               ; RETURN LAST D.CLOSE ERROR REPORTED
                PAGE
                REP        60
*
* GET DEVICE NUMBER
*
* INPUT:  CPATH
* OUTPUT: DEVICE NUMBER (A)
* ERROR:  CARRY SET ("FILE NOT FOUND")
*
* GETDNUM FIRST CALLS THE DMGR (GETDEVNUM) MAP THE PATHNAME
* TO A DEVICE #.  GETDNUM THEN ENSURES THAT THE PATHNAME
* IS NOT A BLOCK DEVICE BY CHECKING THE DBLKLST TABLE.
*
                REP        60
*
GETDNUM         EQU        *
                LDA        #GETDEVNUM
                STA        D.SCNUM
*
                LDA        O.PATH
                STA        GDN.DNAME
                LDA        O.PATH+1
                STA        GDN.DNAME+1
*
                LDA        #>DNUM.TEMP
                STA        GDN.DNUM
                LDA        #<DNUM.TEMP
                STA        GDN.DNUM+1
*
                LDA        SXPAGE+O.PATH+1
                STA        SXPAGE+GDN.DNAME+1
                LDA        #0
                STA        SXPAGE+GDN.DNUM+1
*
                JSR        DMGR
                BCS        GETD.ERR             ; D.NAME NOT FOUND
                BMI        GETD.ERR             ; BLOCK DEVICE FOUND
                LDA        DNUM.TEMP
                RTS
*
GETD.ERR        LDA        #FNFERR
                JSR        SYSERR
                PAGE
                REP        60
* REQUEST FCB ENTRY
*
* INPUT: DNUM (A)
* OUTPUT: REFNUM (A), OPENCT (Y)
* ERROR:  CARRY SET ("CFCB FULL")
*
* REQ.CFCB FIRST SEARCHES THE CFCB TABLE USING THE DEV#
* AS A KEY.  IF FOUND THE OPENCT IS INCREMENTED, OTHERWISE,
* REQ.CFCB FINDS A FREE ENTRY AND STORES THE DEV# AND LEVEL #.
*
                REP        60
*
REQ.CFCB        EQU        *
                LDX        #CFCB.MAX-1
                TAY
REQ010          LDA        CFCB.DEV,X
```

```
                BMI        REQ020
                DEX
                BNE        REQ010
                LDA        #CFCBFULL
                JSR        SYSERR
REQ020          TYA
                STA        CFCB.DEV,X
                LDA        LEVEL
                STA        CFCB.LVL,X
                TXA
                PHA
                TYA
                JSR        OPENCOUNT
                PLA
                ORA        #$80
                CLC
                RTS                     ; NORMAL EXIT
                PAGE
                REP        60
*
* RELEASE FCB ENTRY
*
* INPUT:  REFNUM (A)
* OUTPUT:  DNUM (A), OPENCT (Y)
* ERROR:   CARRY SET ("INVALID REFNUM")
*
* USES REFNUM AS AN CFCB TABLE INDEX TO RELEASE A CFCB ENTRY.
*
                REP        60
REL.CFCB        EQU        *
                AND        #$7F
                CMP        #CFCB.MAX
                BCS        REL.ERR
                TAX
                LDA        CFCB.DEV,X
                BMI        REL.ERR
                SEC                     ; MARK ENTRY FREE
                ROR        CFCB.DEV,X
                JSR        OPENCOUNT
                CLC
                RTS                     ; NORMAL EXIT
*
REL.ERR         LDA        #BADREFNUM
                JSR        SYSERR
                REP        60
*
* OPENCOUNT SUBROUTINE
*
* INPUT:   DEVNUM (A)
* OUTPUT:  DEVNUM (A), OPENCTR (Y)
```

```
*
* OPENCTR:=COUNT OF ALL CFCB ENTRIES W/CFCB.DEV=DEVNUM
*
                REP        60
OPENCOUNT       EQU        *
                LDY        #0
                LDX        #CFCB.MAX-1
OPNCT010        CMP        CFCB.DEV,X
                BNE        OPNCT020
                INY
OPNCT020        DEX
                BNE        OPNCT010
                RTS
                PAGE
                REP        60
*
* GET FCB ENTRY
*
* INPUT:   REFNUM (A)
* OUTPUT:  DNUM (A)
* ERROR:   CARRY SET ("INVALID REFNUM")
*
* USES REFNUM AS AN INDEX TO RETURN THE CORRESPONDING DEVICE #.
* IF THE ENTRY INDICATED BY REFNUM IS A FREE ENTRY, THEN AN
* ERROR, "INVALID REF NUM" IS RETURNED.
*
                REP        60
GET.CFCB        EQU        *
                AND        #$7F
                CMP        #CFCB.MAX
                BCS        GET.ERR
                TAX
                LDA        CFCB.DEV,X
                BMI        GET.ERR
                CLC
                RTS                           ; NORMAL EXIT
*
GET.ERR         LDA        #BADREFNUM
                JSR        SYSERR             ; ERR EXIT
*
                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENCFM
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR CFMGR"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: CFMGR.SRC
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.D3MAIN.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.MAIN.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
* MAIN ENTRY POINT:
*
* DISABLE NMI/RESET AND ENABLE ROM/IO SPACE
*
MAIN            EQU        *
                LDA        E.REG                ;SAVE CALLER'S
                AND        #$FF-$20             ;DROP SCREEN BIT
                STA        ESAVE                ; ENVIRONMENT
                DO         1-TEST               ;NO RESETLOCK FOR TESTING
                LDA        E.REG                ;GET EREG AGAIN
                AND        #$FF-$10             ;DISABLE NMI/RESET
                FIN
                ORA        #$03                 ;ENABLE ROM/IO SPACE
                STA        E.REG
*
                LDA        NOSCROLL             ;DISABLE SMOOTHSCROLL
*
                PHP                             ;IF ALREADY SEI'D, THEN WE
                PLA                             ; STAY THAT WAY...
                ROR        A
                ROR        A
                ROR        A
                ROR        A
                STA        IRQMASK              ;'I' BIT INTO BIT7
*
* MAKE SURE WE HAVE A VALID COMMAND:
*
                LDA        D.COMMAND            ;GET IT
                BMI        BADCMD               ;=>WOW!
                BEQ        IOSETUP              ;=>ZERO IS A READ
                CMP        #10                  ;OFF THE END?
                BCS        BADCMD               ;=>YES
                CMP        #9                   ;REPEAT?
                BNE        CMD1                 ;=>NOPE
*
* REPEAT. SIMPLY GET PRIOR COMMAND:
*
                LDA        PREVUNIT             ;IS THIS REPEAT FOR
                CMP        D.UNITNUM            ; SAME UNIT?
                BNE        BADOP                ;=>NO? ILLEGAL!
                LDA        PREVCMD              ;YES, SET COMMAND
                BEQ        RPTOK                ;=>REPEAT'ED READ IS OK
                CMP        #1                   ;IF NOT, IS IT REPEAT'ED WRITE?
                BNE        BADOP                ;=>CAN'T REPEAT OTHER COMMANDS
RPTOK           EQU        *
                STA        D.COMMAND            ;SAME AS BEFORE
                CMP        #0                   ;READ?
                BEQ        IOSETUP              ;=>YES
* NOW REPEAT GOES LIKE OTHERS:
*
*
CMD1            EQU        *
                CMP        #1                   ;WRITE?
                BNE        CMD2                 ;=>NOPE
                JMP        IOSETUP              ;=>YES
CMD2            EQU        *
                CMP        #2                   ;STATUS?
                BNE        CMD3                 ;=>NOT STATUS
                LDA        D.STATCODE           ;IS IT 'SENSE'?
                BEQ        GOSTAT               ;=>YES
                LDA        #XCTLCODE            ;ILLEGAL CODE
                JMP        EXIT
GOSTAT          EQU        *
                JMP        DRVSETUP             ;=>YES
*
CMD3            EQU        *
                CMP        #8                   ;INIT?
                BNE        BADOP                ;=>NOPE
                JMP        INIT                 ;=>YES, DO INIT
*
BADOP           EQU        *
                LDA        #XBADOP              ;ILLEGAL COMMAND
                JMP        EXIT                 ;BACK TO YOU
*
BADCMD          EQU        *
                LDA        #XREQCODE            ;INVALID COMMAND
                JMP        EXIT                 ;BACK TO YOU
                PAGE
* SETUP WHAT WE HAVE TO BEFORE
*  PERFORMING THE I/O OPERATION:
*
IOSETUP         EQU        *
                LDA        D.BLOCK+1            ;VALIDATE BLOCKNUM
                BEQ        CHKBYTE              ;=> IF <256, IT'S OK
                CMP        #2                   ;IS IT <512?
                BCS        BADBLOCK             ;=>BAD BOY!
                LDA        D.BLOCK              ;YES, CHECK LO HALF
                CMP        #280-256             ; FOR RANGE
                BCC        CHKBYTE              ;=>IT'S OK
```

```
BADBLOCK        EQU        *
                LDA        #XBLKNUM             ;BAD BLOCK NUMBER
                JMP        EXIT                 ;RETURN BAD NEWS
*
CHKBYTE         EQU        *
                LDA        D.BYTES              ;GET LO COUNT
                BNE        BADCOUNT             ;=>ERR, NOT INTEGRAL BLOCK(S)
                LDA        D.BYTES+1            ;GET HI COUNT
                LSR        A                    ;MAKE BLOCK COUNT
                BCS        BADCOUNT             ;=>BAD IF HALF-BLOCK COUNT
                STA        BLKCOUNT             ;SAVE COUNT OF BLOCKS
*
* DOES REQUESTED BYTECOUNT CAUSE US
*  TO RUN OFF END OF DISK?
*
                LDA        BLKCOUNT             ;NO. ADD STARTBLOCK
                CLC                             ; AND BLKCOUNT AND SEE
                ADC        D.BLOCK              ;  IF WE'RE TOO BIG
                LDX        D.BLOCK+1            ;DID IT START OUT > 255?
                BNE        BLKG255              ;=>YES
                BCC        DRVSETUP             ;=>DEFINITELY < 256
                BCS        CHKLO                ;=>IF CARRY,THEN >256
BLKG255         EQU        *
                BCS        BADCOUNT             ;>255+CARRY IS NOW >511
CHKLO           EQU        *
                CMP        #280-256+1           ;281..511 ?
                BCC        DRVSETUP             ;=>NO, WE ARE OK
BADCOUNT        EQU        *
                LDA        #XBYTECNT            ;ILLEGAL BYTECOUNT
                JMP        EXIT                 ;SORRY...
                PAGE
*
* SELECT THE APPROPRIATE DRIVE:
*
DRVSETUP        EQU        *
                LDA        D.COMMAND            ;SAVE THIS COMMAND
                STA        PREVCMD              ; AND DEVICE FOR
                LDA        D.UNITNUM            ;  SUBSEQUENT
                STA        PREVUNIT             ;   'REPEAT' CALL
                LDA        E.REG                ;DOWNSHIFT TO
                ORA        #$80                 ; 1MHZ FOR REMAINDER
                STA        E.REG                ;  OF DRIVER EXECUTION
                JSR        UNITSEL              ;SELECT & START IT
*
* SEE IF THE MOTOR STARTED. IF NOT,
*  THEN IT'S EITHER DISKSWITCH OR NODRIVE.
*
                JSR        CHKDRV               ;MOTOR RUNNING?
                BNE        DOIO                 ;=>YES, GREAT.
*
```

```
* IF WE GET A MOTOR WHEN WE MOVE
*  THE HEAD, THEN IT'S DISKSWITCH.
*
                LDX        D.UNITNUM           ;FORCE HEAD MOTION
                INC        DRVTRACK,X          ; EVEN IF ALREADY ON ZERO
                INC        DRVTRACK,X          ;GIVE HIM A FIRM KNOCKER
                LDA        #0                  ;SEEK TO TRACK ZERO
                JSR        MYSEEK              ; FOR BFM DIR READ
                JSR        CHKDRV              ;RUNNING NOW?
                BNE        DSWITCH             ;=>YES, A SWITCHEROO
                LDA        #0
                LDY        D.UNITNUM           ;FORGET THAT THIS
                STA        DRIVESEL,Y          ; DRIVE WAS 'SELECTED'
                LDA        #XNODRIVE           ;NO, A MISSING DRIVE!
                JMP        EXIT
*
DSWITCH         EQU        *
                LDA        #XDISKSW            ;USER PULLED A FAST ONE
                JMP        EXIT                ; BUT HE CAN'T FOOL US.
                PAGE
* PREPARE TO DO THE OPERATION:
*
DOIO            EQU        *
                LDA        D.BUFL              ;COPY USER BUFFER
                STA        BUFTEMP             ; AND BLOCK NUMBER
                LDA        D.BUFH              ;  TO OUR WORKSPACE
                STA        BUFTEMP+1
                LDA        $1400+D.BUFH
                STA        $1400+BUFTEMP+1
                LDA        D.BLOCK
                STA        BLKTEMP
                LDA        D.BLOCK+1
                STA        BLKTEMP+1
*
* IF CALLER GAVE US A COUNT OF ZERO BYTES,
*  THEN WE'RE ALL DONE!
*
                LDA        D.COMMAND           ;IS IT STATUS?
                CMP        #2                  ;IF SO, THEN BYTECOUNT
                BNE        DOIO2               ; IS MEANINGLESS
                JMP        STATUS
DOIO2           EQU        *
                LDY        BLKCOUNT            ;BLKS=0?
                BEQ        READOK              ;=>YES, YOU GET GOOD RETURN
                CMP        #0                  ;READ COMMAND?
                BEQ        READREQ             ;=>YES
                JMP        WRITEREQ
                PAGE
                REP        40
*  -- READ --
```

```
                REP        40
READREQ         EQU        *
                LDA        #0                   ;CLEAR COUNT OF
                LDY        #0
                STA        (D.BYTRD),Y          ; BYTES READ
                INY
                STA        (D.BYTRD),Y
READREQ2        EQU        *
                JSR        BLK2SECT             ;COMPUTE TRK/SECTOR THIS BLOCK
*
                JSR        SECTORIO             ;READ IT PLEASE
                BCS        READERR              ;=>WE LOSE.
                INC        SECTOR               ;BUMP TO NEXT
                INC        SECTOR               ; LOGICAL SECTOR
                INC        BUF+1                ;BUMP SECTOR BUFFER
                JSR        SECTORIO             ;READ IT TOO
                BCS        READERR              ;=>WE LOSE.
                LDY        #1
                LDA        (D.BYTRD),Y          ;BUMP COUNT OF
                CLC
                ADC        #2
                STA        (D.BYTRD),Y          ; BYTES READ
*
* MORE BLOCKS TO GO?
*
                JSR        MOREBLKS             ;SETUP FOR NEXT BLOCK
                BNE        READREQ2             ;=>MORE TO READ...
READOK          EQU        *
                LDA        #0                   ;GOOD RETURN
                JMP        EXIT                 ;TELL HAPPY USER
*
READERR         EQU        *
                JMP        EXIT                 ;RETURN ERROR CODE
                CHN        DISK3.WRT.SRC

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.MAIN.SRC
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.D3SIO.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SIO.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        40
* NAME    : SECTORIO
* FUNCTION: READ OR WRITE A SECTOR
* INPUT   : IBSTRK, IBSECT, MONTIME,
* RETURNS : CARRY CLEAR IF OK (AC=00)
*         : CARRY SET   IF ERROR (AC=ERRCODE)
*         : SEEKWAIT  ALL SETUP
* DESTROYS: ALL REGISTERS
                REP        40
*
SECTORIO        EQU        *
                LDA        #R.RECAL             ;SETUP THE
* R.RECAL MUST BE NON-ZERO!! (SEE BELOW)
                STA        RECALCNT             ; RECAL TRIES
                NOP                             ; PAD ONE BYTE
                STA        E1908                ; A-REG MUST BE NON-ZERO !!!
* E1908 = NON-ZERO LOCKOUT MOUSE
*
                LDY        D.UNITNUM            ;ARE WE ON-TRACK?
                LDA        TRACK
                CMP        DRVTRACK,Y
                BEQ        SOUGHT               ;=>IF SO, FORGET SEEK & DELAY!
*
* WAIT BEFORE STEPPING:
*
                LDA        SEEKWAIT             ;SEEK DELAY NEEDED?
                BEQ        GOSEEK               ;=>NAW...
                LDA        #0
                STA        SEEKWAIT             ;CLEAR THE FLAG
                LDA        #4                   ;ADD SEEKDELAY TO
                JSR        ADDTIME              ; THE TOTAL UPTIME(S)
                TAY                             ;4*25 MS DELAY
SEEKDEL         EQU        *
                LDA        #0
                JSR        MSWAIT
                DEY
                BNE        SEEKDEL
*
* ISSUE THE SEEK:
```

```
*
GOSEEK          EQU        *
                LDA        TRACK                ;GET DESTINATION TRACK
                JSR        MYSEEK               ;=>..AND YOU SHALL FIND...
*
SOUGHT          EQU        *
                LDA        IRQMASK              ;SET IRQ MASK FOR
                STA        IMASK                ; CORE ROUTINES
                LDA        #R.IRQ               ;SETUP IRQ RETRIES
                STA        INTRTRY
                LDA        #R.IOERR             ; AND ERROR RETRIES
                STA        RETRYCNT
*
* DELAY FOR ANY REMAINING MOTOR-UP TIME:
*
MDELAY          EQU        *
                LDA        MONTIMEH             ;ANY TIME REMAINING?
                BPL        FINDIT               ;=>NO, WE'RE UP TO SPEED.
                LDA        #1                   ;YES, SO BUMP A SLICE OF
                JSR        ADDTIME              ; UPTIME WHILE WE WAIT
                LDA        #0
                JSR        MSWAIT
                JMP        MDELAY               ;=>GO TILL ENOUGH
*
* FIND THE DESIRED SECTOR:
*
* NOTE: FINDSECT RETURNS WITH
*       IRQ INHIBITED!
*
FINDIT          EQU        *
                PHP                             ;INHIBIT IRQ WHILE
                SEI                             ; MESSING WITH VBL FLAGS
                LDA        E.IER                ;DISABLE VBL IRQ
                AND        #$18                 ; DURING SECTOR I/O
                STA        E.IER
                ORA        #$80                 ;FOR 'SET' LATER
                STA        VBLSAVE
                PLP                             ;RESTORE IRQ STATUS
                JSR        FINDSECT             ;FIND ME PLEASE
                BCS        TRYRECAL             ;=>NO? RECAL OR GIVE UP!
                LDX        #$60                 ;SET UP SLOT FOR CORE RTNS
                LDA        D.COMMAND            ;WHAT'S YOUR PLEASURE?
                BNE        SIOWRITE             ;=>WRITE
*
                REP        40
* READ A SECTOR:
*
                JSR        READ                 ;READ THAT SECTOR
                JSR        FIXIRQ               ;ENABLE IRQ IF OK
                LDA        VBLSAVE              ;ALLOW VBL DURING
                STA        E.IER                ; POSTNIB
                BCS        BADIO                ;=>I/O ERR OR IRQ
                LDA        E.REG                ;SET 2MHZ FOR POSTNIB
                AND        #$7F
                STA        E.REG
                JSR        POSTNIB              ;POSTNIB/CHECKSUM
                BCS        IORETRY              ;=>I/O ERR:BAD CHKSUM
                JMP        SIOGOOD              ;=>GOOD READ
*
                REP        40
* WRITE A SECTOR:
*
SIOWRITE        EQU        *
                JSR        WRITE                ;WRITE THE DATA
                JSR        FIXIRQ               ;RE-ENABLE IRQ IF OK
                LDA        VBLSAVE              ;RESTORE
                STA        E.IER                ; VBL IRQ
                BCC        SIOGOOD              ;=>GOOD WRITE
                BVC        SIOWPROT             ;=>WRITE PROTECTED
*
                REP        40
* IT DIDN'T GO WELL FOR US:
*
BADIO           EQU        *
                DO         1-REV0ROM            ;FOR REV1
                BVS        FINDIT               ;=>IRQ. JUST RETRY IT.
                ELSE                            ;FOR REV0
*
* THE REV1 ROM TAKES CARE OF THE
*  IRQ RETRY COUNT, BUT REV0 DOESN'T:
*
                BVC        IORETRY              ;=>I/O ERROR. RETRY IT
                LDA        ROMREV               ;WHICH ROM?
                BNE        FINDIT               ;=>REV1. HE DOES IT.
                LDA        INTRTRY              ;REV0. OUT OF RETRIES?
                BPL        BADIO2               ;=>NO.
                STA        IMASK                ;SET HI BIT FOR IRQ MASK
BADIO2          EQU        *
                DEC        INTRTRY              ;ONE LESS RETRY
                JMP        FINDIT               ;=>RETRY AFTER IRQ
                FIN
*
* RETRY AFTER AN I/O ERROR:
*
IORETRY         EQU        *
                DEC        RETRYCNT             ;ANY RETRIES LEFT?
                BNE        FINDIT               ;=>YEAH, RETRY AFTER ERROR
*
* RETRIES EXHAUSTED. RECALIBRATE:
*
```

```
TRYRECAL        EQU        *
                LDA        VBLSAVE              ;ALLOW VBL IF RECAL
                STA        E.IER                ; OR UNRECOVERABLE ERROR
                DEC        RECALCNT             ;HAVE WE RECALIBRATED YET?
                BMI        SIOERR               ;=>YUP. WE'RE DEAD.
                JSR        RECAL                ;NO, TRY OUR LUCK
                LDY        D.UNITNUM            ;ARE WE ON-TRACK?
                LDA        TRACK
                CMP        DRVTRACK,Y
                BNE        NOTSAME
                JMP        SOUGHT               ;=>IF SO, FORGET RESEEK
NOTSAME         EQU        *
                JMP        GOSEEK               ;TRY AGAIN ON TARGET TRACK
*
                REP        40
SIOERR          EQU        *
                LDA        #XIOERROR            ;RETURN CODE
                SEC                             ;INDICATE HARD ERROR
                BCS        SIORET
SIOWPROT        EQU        *
                LDA        #XNOWRITE            ;RETURN CODE
                SEC                             ;INDICATE HARD ERROR
                BCS        SIORET
SIOGOOD         EQU        *
                LDA        #0
                CLC                             ;INDICATE GOOD COMPLETION
SIORET          LDX        #0                   ; SAY OK TO MOUSE
                STX        E1908                ; WITH THIS GLOBAL $1908
                RTS
                PAGE
                REP        40
* NAME    : FINDSECT
* FUNCTION: LOCATE A DESIRED SECTOR
* INPUT   : IBTRK, IBSECT SETUP
* RETURNS : CARRY CLEAR IF OK,
*         : CARRY SET   IF ERROR.
* DESTROYS: ALL REGISTERS & 'TEMP'
* NOTE    : RETURNS WITH IRQ DISABLED IF NO ERROR!
                REP        40
*
FINDSECT        EQU        *
                LDA        #R.FIND*16           ;SETUP NUMBER OF REVS
                STA        RETRYADR             ; ALLOWED TO FIND SECTOR
                LSR        TEMP                 ;COMPUTE LATENCY FIRST TIME THRU
FINDSEC2        EQU        *
                LDX        #$60                 ;FAKE SLOT FOR CORE ROUTINES
                JSR        RDADR                ;GET NEXT ADDRESS FIELD
                BCS        RDADERR              ;=>UGH! AN ERROR!
*
* MAKE SURE WE'RE ON THE CORRECT TRACK:
```

```
*
                LDA        TRACK                ;IS IT
                CMP        CSSTV+2              ; CORRECT TRACK?
                BNE        FINDERR              ;=>NO?!? IT'S USELESS!
                LDA        SECTOR               ;IS IT
                CMP        CSSTV+1              ; DESIRED SECTOR?
                BEQ        FINDGOOD             ;=>YEAH. GOT IT!
*
* COMPUTE LATENCY. EACH TWO-SECTOR
*  DISTANCE IS 25 MS OF UPTIME.
*
                LDA        TEMP                 ;LATENCY ALREADY COMPUTED?
                BMI        RDADERR              ;=>YES.
                LDA        SECTOR               ;HOW FAR AWAY IS OUR
                SEC                             ; DESIRED SECTOR?
                ROR        TEMP                 ;PREVENT RECOMPUTATION
                SBC        CSSTV+1
                AND        #$0F
                LSR        A                    ;EACH 2-SECTORS IS 25 MS
                JSR        ADDTIME
*
* KEEP LOOKING TILL WE FIND IT:
*
RDADERR         EQU        *
                JSR        FIXIRQ               ;ENABLE IRQ IF APPROPRIATE
                DEC        RETRYADR             ;ANY RETRIES LEFT?
                BEQ        FINDERR              ;=>NO, WE CAN'T FIND IT.
*
* COMPENSATE FOR A BUG IN RDADR: IF WE TRY
*  TO CALL RDADR AGAIN BEFORE THE DATA MARK
*  GOES BY, THEN RDADR WILL ACCIDENTALLY CALL
*  THAT AN ERROR. WE CAN AVOID THIS 'FAKE'
*  ERROR BY DELAYING PAST THE DATA MARK.
                LDY        #200                 ;1 MS IS PLENTY
ADRDELAY        EQU        *
                DEY
                BNE        ADRDELAY
                JMP        FINDSEC2             ;=>NOW TRY LOOKING AGAIN
*
                REP        40
FINDGOOD        EQU        *
                LDA        #0                   ;CLEAR VOLNUM OUT OF
                STA        MONTIMEH             ; MOTORTIME!
                CLC                             ;INDICATE NO ERROR
                RTS
*
FINDERR         EQU        *
                JSR        FIXIRQ               ;ENABLE IRQ IF APPROPRIATE
                LDA        #0                   ;CLEAR VOLNUM OUT OF
                STA        MONTIMEH             ; MOTORTIME!
                SEC                                ;INDICATE THE ERROR
                RTS

                CHN        DISK3.USEL.SRC

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SIO.SRC
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.D3SRC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       'SOS 1.1  DISK /// DRIVER'
TEST            EQU        0                    ;FOR FUNNY-MODE TESTING
                INCLUDE    SOSORG,6,1,254
                DO         TEST
                ORG        $2000
                ELSE
                REL
                ORG        ORGDISK3
                FIN
ZZORG           EQU        *
                CHR        '-'
                MSB        OFF
*
                REP        40
*    COPYRIGHT (C) APPLE COMPUTER INC.
*          ALL RIGHTS RESERVED
                REP        40
*
REV0ROM         EQU        0                    ;1=SUPPORT REV0 ROM
*
                DO         1-TEST
                ENTRY      DIB1                 ;DIB1
                ENTRY      DIB2                 ;DIB2
                ENTRY      DIB3                 ;DIB3
                ENTRY      DIB4                 ;DIB4
                ENTRY      SEEKDSK3             ;SEEK CURRENT DRIVE
*
                EXTRN      SYSERR
*
                EXTRN      XREQCODE
                EXTRN      XBADOP
                EXTRN      XNODRIVE
                EXTRN      XIOERROR
                EXTRN      XNOWRITE
                EXTRN      XBYTECNT
                EXTRN      XBLKNUM
                EXTRN      XDISKSW
                EXTRN      XCTLCODE
*
                EXTRN      E1908                ; GLOBAL FLAG FOR MOUSE DRIVER
```

```
* TO SAY WE CANNOT BE INTERRUPTED
*
                ELSE
XREQCODE        EQU        $20
XBADOP          EQU        $26
XNODRIVE        EQU        $28
XIOERROR        EQU        $27
XNOWRITE        EQU        $2B
XBYTECNT        EQU        $2C
XBLKNUM         EQU        $2D
XDISKSW         EQU        $2E
XCTLCODE        EQU        $21
                FIN
                PAGE
* DISK /// CONTROLLER EQUATES:
*
*       MOTOR SELECT BITS:
*
*    DRIVE    INT   EXT1   EXT2
*    -----    ---   ----   ----
*     .D1      1     X      X
*     .D2      X     0      1
*     .D3      X     1      0
*     .D4      X     1      1
*
MS.INT          EQU        $C0D4                ;MOTOR   SELECT:INTERNAL DRIVE
MD.INT          EQU        $C0D5                ;MOTOR DESELECT:INTERNAL DRIVE
*
MS.EXT1         EQU        $C0D3                ;MOTOR   SELECT:EXTERNAL DRIVE
MS.EXT2         EQU        $C0D1                ;MOTOR   SELECT:EXTERNAL DRIVE
MD.EXT1         EQU        $C0D2                ;MOTOR DESELECT:EXTERNAL DRIVE
MD.EXT2         EQU        $C0D0                ;MOTOR DESELECT:EXTERNAL DRIVE
*
IS.INT          EQU        $C0EA                ;I/O SELECT:INTERNAL DRIVE
IS.EXT          EQU        $C0EB                ;I/O SELECT:EXTERNAL DRIVE
*
NOSCROLL        EQU        $C0D8                ;SMOOTHSCROLL OFF
*
MOTOROFF        EQU        $C0E8                ;MOTOR(S) START POWEROFF T/O
MOTORON         EQU        $C0E9                ;MOTOR(S) POWER ON
Q6L             EQU        $C08C                ;Q7L,Q6L=READ
Q6H             EQU        $C08D                ;Q7L,Q6H=SENSE WPROT
Q7L             EQU        $C08E                ;Q7H,Q6L=WRITE
Q7H             EQU        $C08F                ;Q7H,Q6H=WRITE STORE
*
* OTHER EQUATES:
*
E.REG           EQU        $FFDF                ;ENVIRONMENT REGISTER
E.IER           EQU        $FFEE                ;INTERRUPT ENABLE REGISTER
*
```

```
* RETRY COUNTERS:
*
R.RECAL         EQU        1                    ;MAX RECALIBRATES
* R.RECAL MUST NOT BECOME ZERO! (MOUSE WILL BE LOCKED OUT)
* SEE DISK3.SIO.SRC LINE 14 FOR DETAIL
R.FIND          EQU        3                    ;MAX REVS TO FIND A SECTOR
R.IOERR         EQU        4                    ;MAX RETRIES ON READ ERROR
R.IRQ           EQU        6                    ;MAX IRQ'S TOLERATED BEFORE SEI
                PAGE
* ZPAGE EQUATES FOR CORE ROUTINES:
*
                DSECT
                ORG        $81
IBSLOT          DS         1                    ;SLOT=$60 FOR RTNS
                DS         7                    ;N/A
                DS         1                    ;RDADR:CHECKSUM
                DS         1                    ;N/A
IMASK           DS         1                    ;BIT7 SET IF IRQ ALLOWED
CURTRK          DS         1                    ;SEEK:CURRENT TRACK
                DS         2                    ;N/A
INTRTRY         DS         1                    ;READ: IRQ RETRY COUNT
                DS         5                    ;N/A
                DS         1                    ;RDADR:'MUST FIND' COUNT
                DS         1                    ;READ,WRITE: CHECKSUM
CSSTV           DS         4                    ;RDADR:CKSUM,SEC,TRK,VOL
MONTIMEL        EQU        CSSTV+2              ;MSWAIT:MOTOR-ON TIME
MONTIMEH        EQU        MONTIMEL+1
BUF             DS         2                    ;PRENIB,POSTNIB:USER BUFFER
                DS         1                    ;SEEK:PRIOR PHASE
TRKN            DS         1                    ;SEEK:TARGET TRACK
*
* LOCAL TEMPS:
*
                ORG        $D0                  ;WE'RE ALLOWED TO $FF
BLKTEMP         DS         2                    ;LOCAL TEMP FOR BLKNUMBER
BUFTEMP         DS         2                    ;LOCAL TEMP FOR BUFFER ADDRESS
TRACK           DS         1                    ;LOCAL TEMP FOR TRACK
SECTOR          DS         1                    ;LOCAL TEMP FOR SECTOR
RETRYADR        DS         1                    ;LOCAL TEMP FOR SECTOR-FIND RETRIES
RETRYCNT        DS         1                    ;LOCAL TEMP FOR I/O RETRIES
RECALCNT        DS         1                    ;LOCAL TEMP FOR RECAL COUNT
BLKCOUNT        DS         1                    ;BLKS REQD TO SATISFY BYTECOUNT
SEEKWAIT        DS         1                    ;<>0 IF SEEK DELAY NEEDED
IRQMASK         DS         1                    ;ENTRY 'I' BIT
TEMP            DS         1                    ;JUST A TEMP
                DEND
                PAGE
* DRIVER INTERFACE AREA:
*
                DSECT
                ORG        $C0
D.COMMAND       DS         1                    ;COMMAND CODE
D.UNITNUM       DS         1                    ;UNIT NUMBER
D.BUFL          DS         2                    ;BUFFER ADDRESS
D.BUFH          EQU        D.BUFL+1
D.STATCODE      EQU        D.BUFL               ;DSTATUS CODE
D.STATBUF       EQU        D.BUFH               ;^DSTATUS LIST
D.BYTES         DS         2                    ;BYTECOUNT
D.BLOCK         DS         2                    ;REQUESTED BLOCKNUM
D.BYTRD         DS         2                    ;BYTES READ (READ)
                DS         6                    ;SPARES (OK AS TEMPS)
                DEND
                PAGE
DIB1            EQU        *                    ;DIB FOR .D1
                DW         DIB2                 ;FLINK
                DW         MAIN                 ;ENTRY POINT
                DFB        3                    ;NAME LENGTH
                ASC        '.D1            '
                DFB        $80                  ;DEVNUM: ACTIVE
                DFB        0                    ;SLOT
                DFB        0                    ;UNIT NUMBER
                DFB        $E1,1,0              ;TYPE,SUB,FILLER
                DW         280                  ;BLOCKCOUNT
                DW         1                    ;MANUFACTURER=APPLE
                DW         $1100                ;VERSION=1.1
*
DIB2            EQU        *                    ;DIB FOR .D2
                DW         DIB3                 ;FLINK
                DW         MAIN                 ;ENTRY POINT
                DFB        3                    ;NAME LENGTH
                ASC        '.D2            '
                DFB        $80                  ;DEVNUM: ACTIVE
                DFB        0                    ;SLOT
                DFB        1                    ;UNIT NUMBER
                DFB        $E1,1,0              ;TYPE,SUB,FILLER
                DW         280                  ;BLOCKCOUNT
                DW         1                    ;MANUFACTURER=APPLE
                DW         $1100                ;VERSION=1.1
*
DIB3            EQU        *                    ;DIB FOR .D3
                DW         DIB4                 ;FLINK
                DW         MAIN                 ;ENTRY POINT
                DFB        3                    ;NAME LENGTH
                ASC        '.D3            '
                DFB        $80                  ;DEVNUM: ACTIVE
                DFB        0                    ;SLOT
                DFB        2                    ;UNIT NUMBER
                DFB        $E1,1,0              ;TYPE,SUB,FILLER
                DW         280                  ;BLOCKCOUNT
                DW         1                    ;MANUFACTURER=APPLE
                DW         $1100                ;VERSION=1.1
*
DIB4            EQU        *                    ;DIB FOR .D4
                DW         0                    ;NO FLINK
                DW         MAIN                 ;ENTRY POINT
                DFB        3                    ;NAME LENGTH
                ASC        '.D4            '
                DFB        $80                  ;DEVNUM: ACTIVE
                DFB        0                    ;SLOT
                DFB        3                    ;UNIT NUMBER
                DFB        $E1,1,0              ;TYPE,SUB,FILLER
                DW         280                  ;BLOCKCOUNT
                DW         1                    ;MANUFACTURER=APPLE
                DW         $1100                ;VERSION=1.1
                DW         1                    ;MANUFACTURER=APPLE
                DW         $1100                ;VERSION=1.1

                CHN        DISK3.MAIN.SRC
                INCLUDE    SOSORG,6,1,254

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SRC
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.D3SUBS.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SUBS.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        40
* NAME    : CHKDRV
* FUNCTION: CHECK IF MOTOR(S) RUNNING
* INPUT   : NONE
* RETURNS : 'BNE' IF RUNNING
*         : 'BEQ' IF NOT
* DESTROYS: AC,X
                REP        40
* NOTES: DUE TO A FLOATING PIN, THERE
*  COULD BE A GLITCH WHICH CAUSES THE
*  SHIFTER TO 'FLASH' ONTO THE BUS
*  INSTEAD OF ALWAYS BEING TRISTATED.
*  THIS COULD CAUSE CHKDRV TO THINK
*  THAT THE MOTOR IS SPINNING WHEN IT
*  IS NOT. THUS WE WILL SAMPLE THE SHIFTER
*  FOR 40 US AT 6-US INTERVALS. IF, AFTER
*  THREE (3) CONSECUTIVE PASSES, ANY OF
*  THE PASSES SEES A 'LOCKED' SHIFTER,
*  THEN WE SAY THE DRIVE IS STOPPED.
*
*
CHKDRV          EQU        *
                LDX        #3                    ;CHECK SHIFTER SEVERAL TIMES
CHKD1           EQU        *
                LDA        Q6L+$60               ;GET DATA
                CMP        Q6L+$60               ;HAS IT CHANGED?
                BNE        CHANGED               ;=>YES
                CMP        Q6L+$60               ;HAS IT CHANGED?
                BNE        CHANGED               ;=>YES
                CMP        Q6L+$60               ;HAS IT CHANGED?
                BNE        CHANGED               ;=>YES
                CMP        Q6L+$60               ;HAS IT CHANGED?
                BNE        CHANGED               ;=>YES
                CMP        Q6L+$60               ;HAS IT CHANGED?
                BNE        CHANGED               ;=>YES
                CMP        Q6L+$60               ;HAS IT CHANGED?
                BNE        CHANGED               ;=>YES
                CMP        Q6L+$60               ;HAS IT CHANGED?
                BNE        CHANGED               ;=>YES
                RTS                             ;IF EVER LOCKED, IT'S STOPPED
*
CHANGED         EQU        *
                DEX
                BNE        CHKD1                ;TRY SEVERAL TIMES
                DEX                             ;SET CC=BNE
                RTS                             ;RETURN ZFLAG APPROPRIATELY
                PAGE
                REP        40
* NAME    : ADDTIME
* FUNCTION: ADD TO MOTOR UPTIME(S)
* INPUT   : AC=NO. OF 25 MS INCREMENTS
* DESTROYS: Y
                REP        40
*
ADDTIME         EQU        *
                PHA                             ;PRESERVE AC
                LDY        #4                   ;TABLE INDEX/COUNT
ADD2            EQU        *
                LDA        DRIVESEL-1,Y         ;IS IT SELECTED?
                BEQ        ADD3                 ;=>NOPE
                PLA
                PHA                             ;RECOVER DELTA-T
                CLC
                ADC        UPTIME-1,Y           ;ADD TO MOTOR UPTIME
                CMP        #T1SEC+2             ;IS IT AT MAX TIME?
                BCC        ADD2A                ;=>NO, STORE NEW TIME
                LDA        #T1SEC+1             ;YES, SET TO >1 SEC
ADD2A           EQU        *
                STA        UPTIME-1,Y
ADD3            EQU        *
                DEY
                BNE        ADD2                 ;=>DO ALL 4 DRIVES
*
                PLA                             ;RESTORE AC
                RTS
                PAGE
                REP        40
* NAME    : RECAL
* FUNCTION: RECALIBRATE DRIVE HEAD
* INPUT   : NONE
* DESTROYS: ALL REGISTERS
* NOTE    : A 'QUIET' RECALIBRATE IS DONE
*         : USING TWO ITERATIONS. IF WE ARE
*         : LOST, THEN SEEK 48-TRACKS
*         : TOWARD TRACK ZERO. IF WE KNOW
*         : WHAT TRACK WE'RE CURRENTLY
*         : ON (+- 1/2 TRACK), THEN JUST
*         : ADD A LITTLE EXTRA AND SEEK
*         : TO TRACK ZERO. A 48-TRACK
*         : SEEK WILL ALWAYS GET US BACK
*         : ONTO THE MEDIA, EVEN IF WE
*         : WERE "OFF THE CAM". FROM THAT
*         : POINT, THE 2ND SEEK GETS US
*         : BACK TO TRACK ZERO QUIETLY.
                REP        40
*
RECAL           EQU        *
                LDA        #2                   ;TWO ITERATIONS, PLEASE
RECAL1          EQU        *
                PHA                             ;SAVE LOOPCOUNT
                LDX        #$60                 ;SETUP SLOT FOR CORE RTNS
                JSR        RDADR                ;WHERE ARE WE?
                BCC        RECAL2               ;=>NOW WE KNOW
                JSR        RDADR                ;GIVE SECOND SHOT
                BCC        RECAL2               ;=>THAT GOT IT
                LDA        #48                  ;LOST? TRY 48-TRACK SEEK
                JMP        RECAL3
RECAL2          EQU        *
                LDA        CSSTV+2              ;HERE'S WHERE WE ARE
                CLC                             ;ADD SOME SO WE GET A
                ADC        #3                   ; HARDER SEEK TO ZERO
RECAL3          EQU        *
                LDY        D.UNITNUM            ;THIS IS NOW WHERE
                STA        DRVTRACK,Y           ; WE ARE
                JSR        FIXIRQ               ;ENABLE IRQ IF OK
*
                LDA        #0                   ;DESTINATION TRACK IS 00
                STA        MONTIMEH             ;CLEAR MOTOR-UP TIME SO
                STA        MONTIMEL             ; SEEK KNOWS HOW LONG RECAL TAKES
                JSR        MYSEEK               ;=>SLAM IT BACK!
                PLA                             ;HAVE WE DONE IT TWICE?
                TAY
                DEY
                TYA
                BNE        RECAL1               ;=>DO TWO ITERATIONS
                RTS
                PAGE
                REP        40
* NAME    : SEEKDSK3
* FUNCTION: SEEK CURRENT DRIVE
* INPUT   : AC=DESTINATION TRACK
* OUTPUT  : NONE
* DESTROYS: ALL REGISTERS
* NOTE    : MUST BE CALLED WHILE
*         :  MOTOR IS RUNNING, IN
*         :  1MHZ+ROM+IO MODE
                REP        40
SEEKDSK3        EQU        *
                LDY        PREVUNIT             ;GET DRIVENUM
                STY        D.UNITNUM            ;SET IT UP
                JSR        MYSEEK               ;MOVE IT!
                RTS
                REP        40
* NAME    : MYSEEK
* FUNCTION: SEEK TO DESIRED TRACK
* INPUT   : AC=DESTINATION TRACK
* DESTROYS: ALL REGISTERS
                REP        40
MYSEEK          EQU        *
                STA        TRKN                 ;TEMP HOLD OF AC
                LDY        D.UNITNUM            ;GET DRIVENUM
                LDA        DRVTRACK,Y           ;SETUP CURRENT TRACK
                ASL        A                    ;SET IN HALFTRACKS FOR SEEK
                STA        CURTRK               ; FOR SEEK ROUTINE
                LDX        #$60                 ;SET UP SLOT FOR CORE RTNS
                LDA        MONTIMEH             ;GET STARTING MOTOR TIME
                STA        TEMP
*
* NOTE: IRQ'S WHICH SUSPEND SEEK MAY CAUSE A
*  SEEK FAILURE. WE WILL HAVE TO RECALIBRATE
*  SINCE WE WON'T BE ON-TRACK. WE CAN NOT GET
*  ON A HALFTRACK SINCE SEEK ALLOWS SETTLING
*  TIME OF THE PHASE. BECAUSE VBL IS A SERIOUS
*  OFFENDER, WE INHIBIT HIM.
*
                PHP                             ;INHIBIT IRQ WHILE
                SEI                             ; MESSING WITH VBL FLAGS
                LDA        E.IER
                AND        #$18
                STA        VBLSAVE
                STA        E.IER
                PLP                             ;RESTORE IRQ STATUS
                LDA        TRKN                 ;RESTORE DESTINATION TRACK
                STA        DRVTRACK,Y           ;DEST IS NOW CURRENT
                ASL        A                    ;MAKE IT IN HALFTRACKS
                JSR        SEEK                 ;GO MOVE THE HEAD...
                LDA        VBLSAVE              ;NOW ALLOW THAT
                ORA        #$80                 ; NASTY
                STA        E.IER                ;  VBL INTERRUPT
*
* COMPUTE THE TIME USED BY SEEK:
*
                LDA        MONTIMEH             ;INCLUDE SEEKTIME IN
                SEC
                SBC        TEMP
                JSR        ADDTIME              ; TOTAL MOTOR UPTIME(S)
                RTS
                PAGE
                REP        40
```

```
* NAME    : BLK2SECT
* FUNCTION: COMPUTE TRACK/SECTOR FOR A BLOCK
*           AND ADJUST BUFFER ADDRESS
* INPUT   : D.BLOCK, D.BUF
* OUTPUT  : TRACK, SECTOR, D.BUF
* DESTROYS: AC,Y
                REP        40
*
BLK2SECT        EQU        *
                LDA        BLKTEMP+1            ;GET HI BLK HALF
                ROR        A                    ;MOVE LO BIT TO CARRY
                LDA        BLKTEMP              ;GET LO HALF
                ROR        A                    ;COMBINE WITH HI BIT
                LSR        A
                LSR        A                    ;FINISH OFF DIVIDE-BY-8
                STA        TRACK                ;THAT'S THE TRACK
                LDA        BLKTEMP              ;GET LO HALF AGAIN
                AND        #7
                TAY
                LDA        SECTABLE,Y           ;GET START SECTOR
                STA        SECTOR
*
* ADJUST BUFFER ADDRESS SO THAT I/O
*  WON'T WRAPAROUND IN THE BANK:
* (THIS ALGORITHM RIPPED OFF FROM 1.0)
*
                LDA        BUFTEMP+1            ;GET BUFFER HI ADDRESS
                LDY        $1400+BUFTEMP+1      ; AND XTND BYTE
                CMP        #$82                 ;IF RAM ADDR >=8200 THEN BUMP TO
                BCC        NOADJ                ; NEXT BANK PAIR
                CPY        #$80
                BCC        NOADJ                ;=>NOT USING BANKPAIR
                CPY        #$8F                 ;SPECIAL BANK 0?
                BEQ        NOADJ                ;=>YES
                AND        #$7F                 ;DROP HI ADDRESS AND
                STA        BUFTEMP+1            ; BUMP BANK NUMBER
                INC        $1400+BUFTEMP+1
*
NOADJ           EQU        *
                LDA        BUFTEMP+1            ;COPY BUFFER ADDRESS
                STA        BUF+1                ; FOR PRE & POSTNIB
                LDA        BUFTEMP
                STA        BUF
                LDA        $1400+BUFTEMP+1
                STA        $1400+BUF+1
                RTS
*
SECTABLE        DFB        $00,$04,$08,$0C,$01,$05,$09,$0D
                PAGE
                REP        40
```

```
* NAME    : MOREBLKS
* FUNCTION: SETUP TO DO NEXT BLOCK
* INPUT   : NONE
* RETURNS : 'BNE' IF MORE TO DO
*         : 'BEQ' IF NO MORE TO DO
* DESTROYS:NOTHING
                REP        40
*
MOREBLKS        EQU        *
                INC        BUFTEMP+1            ;BUMP BUFFER ADDRESS
                INC        BUFTEMP+1
                INC        BLKTEMP              ;BUMP BLOCK NUMBER
                BNE        MORE2
                INC        BLKTEMP+1
MORE2           EQU        *
                DEC        BLKCOUNT             ;MORE BLOCKS TO GO?
                RTS                             ;RETURN RESULT OF DEC
                SKP        4
                REP        40
* NAME    : FIXIRQ
* FUNCTION: ENABLE IRQ IF APPROPRIATE
* INPUT   : NONE
* DESTROYS: NOTHING
                REP        40
*
FIXIRQ          EQU        *
                PHA
                LDA        IRQMASK              ;SHOULD IRQ BE ENABLED?
                BMI        FIXRET               ;=>NO, LEAVE IT ALONE
                CLI                             ;ENABLE IRQ
FIXRET          EQU        *
                PLA
                RTS

                CHN        DISK3.DATA.SRC

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SUBS.SRC
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.D3USEL.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.USEL.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        40
* NAME    : UNITSEL
* FUNCTION: SELECT & START A DRIVE,
*           SET UP MOTOR & SEEK DELAYS
* INPUT   : NONE
* OUTPUT  : MONTIME,SEEKTIME
* DESTROYS: ALL REGISTERS
                REP        40
*
UNITSEL         EQU        *
                LDY        D.UNITNUM            ;GET DRIVENUM
                LDA        #0                   ;ASSUME NO SEEKWAIT
                STA        SEEKWAIT             ; WILL BE NEEDED
                STA        MONTIMEL             ;CLEAR MONTIME
                STA        MONTIMEH
*
* SEE IF MOTOR(S) STILL SPINNING:
*
                JSR        CHKDRV               ;MOTOR(S) POWERED UP?
                BNE        SPINNING             ;=>YES. WHO IS IT?
*
* NO MOTOR(S) SPINNING. DESELECT
*  ALL MOTORS AND START AFRESH:
*
                LDX        MD.INT               ;DESELECT ALL
                LDA        #0                   ;SHOW INTERNAL AS
                STA        DRIVESEL+0           ; NOT SELECTED
                STA        UPTIME+0             ;INDICATE DRIVE IS FULLY STOPPED
                JSR        EXTDESEL             ;DESELECT ALL EXTERNALS TOO
                JMP        SETTIME              ;GO SETUP MOTOR DELAY
                REP        40
* MOTOR(S) SPINNING: OURS?
*
SPINNING        EQU        *
                LDA        DRIVESEL,Y           ;HAD WE BEEN SELECTED?
                BNE        GOFORIT              ;=>YES, GO FOR IT RIGHT AWAY.
*
* WE AREN'T SPINNING. SHUTDOWN ANOTHER
*  DRIVE, IF NECESSARY, TO GET GOING:
```

```
*
                CPY        #0                   ;ARE WE THE INTERNAL DRIVE?
                BEQ        SETTIME              ;=>YES, LEAVE EXT MOTOR ALONE
*
* WE'RE AN EXTERNAL DRIVE. STOP ALL EXTERNAL MOTORS
*  UNCONDITIONALLY, BUT LEAVE THE INTERNAL MOTOR ALONE.
* IF WE *DID* HAVE TO STOP ANOTHER EXTERNAL, THEN
*  MAKE SURE WE SET THE CORRECT PRE-SEEK DELAY!
*
                LDA        #0                   ;SEE IF ANOTHER EXTERNAL
                ORA        DRIVESEL+3           ; HAD BEEN
                ORA        DRIVESEL+2           ;  SELECTED
                ORA        DRIVESEL+1           ;   BEFORE...
                BEQ        SETTIME              ;=>NO, SEEK DELAY IS UNNECESSARY
                INC        SEEKWAIT             ;YES, DELAY BEFORE STEPPING
                JSR        EXTDESEL             ;DESELECT ALL EXTERNALS
                JMP        SETTIME              ;=>GO SETUP MOTOR DELAY
                PAGE
                REP        40
* OUR DRIVE IS SPINNING. GO FOR IT!
* DEPENDING OF HOW LONG THE MOTOR'S BEEN ON,
*  THIS COMMAND MAY REQUIRE A MOTOR DELAY.
*
GOFORIT         EQU        *
                LDX        D.COMMAND            ;GET CURRENT COMMAND
                LDA        MTIMES,X             ;GET REQUIRED UPTIME FOR IT
                SEC
                SBC        UPTIME,Y             ;DRIVE RUNNING LONG ENOUGH?
                BCS        SELECT               ;=>NO, AC NOW HAS DELTA-T
                LDA        #0                   ;OTHERWISE, WAIT=0
                JMP        SELECT               ;SET MONTIME & SELECT DRIVE
                REP        40
*
* ALL MOTORS WERE OFF. CHOOSE THE
*  APPROPRIATE MOTOR-ON TIME:
*
SETTIME         EQU        *
                LDA        #0                   ;INDICATE THAT
                STA        UPTIME,Y             ; THE DRIVE WAS OFF
                LDX        D.COMMAND            ;GET CURRENT COMMAND
                LDA        MTIMES,X             ;GET CORRECT DELAY TIME
                REP        40
*
* SELECT THE DRIVE & START IT:
*
SELECT          EQU        *
                STA        MONTIMEH             ;NEGATE IT BECAUSE
                LDA        #0                   ; IT GETS INCREMENTED
                SEC                             ;  INSTEAD OF
                SBC        MONTIMEH             ;   DECREMENTED
                STA        MONTIMEH             ;STUFF MOTOR DELAY
                CPY        #1                   ;ARE WE THE INTERNAL DRIVE?
                BCS        SELEXT               ;=>NO, AN EXTERNAL
                LDA        IS.INT               ;I/O SELECT INTERNAL
                LDA        MS.INT               ;MOTOR SELECT INTERNAL
                JMP        UNITRET              ;=>ALL DONE!
*
SELEXT          EQU        *
                LDA        IS.EXT               ;I/O SELECT EXTERNAL
                CPY        #2                   ;ARE WE 2, 3, OR 4 ?
                BCS        NOTD2                ;=>DEFINITELY 3 OR 4
                LDA        MD.EXT1              ;MOTOR SELECT
                LDA        MS.EXT2              ; ONLY .D2
                JMP        UNITRET              ;=>ALL DONE!
*
NOTD2           EQU        *
                BNE        ISD4                 ;=>DEFINITELY NOT 3
                LDA        MS.EXT1              ;MOTOR SELECT
                LDA        MD.EXT2              ; ONLY .D3
                JMP        UNITRET              ;=>ALL DONE!
*
ISD4            EQU        *
                LDA        MS.EXT1              ;MOTOR SELECT
                LDA        MS.EXT2              ; ONLY .D4
*
*
UNITRET         EQU        *
                LDA        MOTORON              ;PROVIDE MOTOR POWER
                LDA        #1                   ;SAY WE'VE SELECTED
                STA        DRIVESEL,Y           ; THIS DRIVE
*
* IF WE HAVE MOTORTIME TO BURN,
*  THEN DELAY 50 MS. THIS ENSURES
*  A GOOD SOLID CHKDRV AFTER
*  TURNING ON THE MOTOR.
*
                LDA        MONTIMEH             ;ANY MOTORTIME?
                BPL        UNITRTS              ;=>NO, WE GO FOR IT.
                LDY        #5                   ;5*10 MS
UNITDEL         EQU        *
                LDA        #100                 ;100*100US IS 10MS
                JSR        MSWAIT
                DEY
                BNE        UNITDEL
                LDA        #2                   ;INCLUDE THE 50MS
                JSR        ADDTIME              ; IN MOTOR UPTIME(S)
UNITRTS         EQU        *
                RTS
                SKP        5
                REP        40
```

```
* NAME    : EXTDESEL
* FUNCTION: DESELECT ALL EXTERNAL DRIVE MOTORS
* INPUT   : NONE
* DESTROYS: AC,X
                REP        40
*
EXTDESEL        EQU        *
                LDA        MD.EXT1             ;DESELECT ALL EXTERNAL
                LDA        MD.EXT2             ; DRIVE MOTORS
                LDX        #3                  ;SHOW THAT THEY ARE
                LDA        #0                  ; ARE ALL DEAD DUCKS
EDS1            STA        DRIVESEL,X
                STA        UPTIME,X            ;DRIVE MOTORS ARE OFF
                DEX
                BNE        EDS1
                RTS

                CHN        DISK3.SUBS.SRC

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.USEL.SRC
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.D3WRT.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.WRT.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        40
* --- WRITE ---
                REP        40
*
WRITEREQ        EQU        *
                JSR        BLK2SECT            ;COMPUTE TRK/SECTOR THIS BLOCK
                LDA        E.REG               ;SET 2 MHZ
                AND        #$7F
                STA        E.REG
                JSR        PRENIB              ;PRENIBBLIZE FOR WRITE
                JSR        SECTORIO            ;WRITE IT OUT...
                BCS        WRITERR             ;=>SOMETHING'S WRONG
*
                INC        SECTOR              ;BUMP TO NEXT
                INC        SECTOR              ; LOGICAL SECTOR
                INC        BUF+1               ;BUMP SECTOR BUFFER ADDRESS
                LDA        E.REG               ;SET 2 MHZ
                AND        #$7F
                STA        E.REG
                JSR        PRENIB              ;PRENIBBLIZE FOR WRITE
                JSR        SECTORIO            ;WRITE IT OUT
                BCS        WRITERR             ;=>SOMETHING'S WRONG
*
* MORE BYTES TO DO?
*
                JSR        MOREBLKS            ;SETUP FOR NEXT
                BNE        WRITEREQ            ;=>MORE TO DO
                LDA        #0                  ;GOOD RETURN
                JMP        EXIT
*
WRITERR         EQU        *
                JMP        EXIT                ;RETURN ERROR CODE
                PAGE
                REP        40
*  --- STATUS ---
                REP        40
*
STATUS          EQU        *
                LDX        #$60                ;DUMMY SLOT
                LDA        Q6H,X                ;SENSE WRITE PROTECT
                LDA        Q7L,X
                ASL        A                    ;PRESERVE IT IN CARRY
                LDA        Q6L,X                ;BACK TO READ MODE
                LDA        #0                   ;NOW MOVE BIT TO
                ROL        A                    ; PROPER POSITION
                ROL        A                    ; ($02)
                LDY        #0
                STA        (D.STATBUF),Y        ;RETURN IT
                LDA        #0                   ;GOOD RETURN
                JMP        EXIT                 ;DONE
                PAGE
                REP        40
* --- INIT ---
                REP        40
*
INIT            EQU        *
                LDA        INITFLAG             ;INIT'ED YET?
                BMI        GOODINIT             ;=>YES, DONE
*
                LDA        #$60                 ;SETUP SLOT FOR
                STA        IBSLOT               ; CORE ROUTINES
                LDA        #$FF                 ;PREVENT SECOND
                STA        INITFLAG             ; INIT
                LDA        #0                   ;CLEAR STUFF OUT
                STA        PREVUNIT             ;SOSBOOT JUST USED .D1
                LDY        #4
CLRDRVS         EQU        *
                LDA        #0
                STA        DRIVESEL-1,Y         ;NOBODY SELECTED
                STA        UPTIME-1,Y           ;ALL OFF
                STA        DRVTRACK-1,Y
                DEY
                BNE        CLRDRVS
                DO         1-TEST               ;ONLY IF NOT TESTING
*
* SET UP .D1 SINCE LOADER'S USING IT:
*
                LDA        E.REG                ;SET 1MHZ FOR THE
                ORA        #$80                 ; STATEMACHINE I/O
                STA        E.REG
                JSR        CHKDRV               ;IS .D1 MOTOR SPINNING?
                BEQ        INIT2                ;=>NO, MOTOR'S OFF
                LDA        #T200MS              ;UPTIME GOOD FOR READS
                STA        UPTIME+0
INIT2           EQU        *
                LDA        #1
                STA        DRIVESEL+0           ;.D1 IS THE CURRENT DRIVE
                LDA        $0300+CURTRK         ;RETRIEVE CURRENT TRACK
                STA        DRVTRACK+0           ;REMEMBER IT
                FIN
*
* SET UP JMP TABLE FOR CORRECT ROM:
*
                DO         REV0ROM             ;ONLY IF SUPPORTING IT!
                LDA        $F1B9               ;LOOK FOR START OF RDADR
                CMP        #$A0                ;IS IT RDADR (REV1)?
                BEQ        INITREV1            ;=>YES
                CMP        #$60                ;IS IT END OF READ (REV0)?
                BNE        INITERR             ;=>NEITHER!
                LDY        #0                  ;REV=0
                BEQ        INITVECT            ;(ALWAYS TAKEN)
INITREV1        EQU        *
                LDY        #VSIZE
INITVECT        EQU        *
                STY        ROMREV              ;SET ROM REVISION INDICATOR
                LDX        #VSIZE
MOVEVECT        EQU        *
                LDA        REV0,Y              ;GET A BYTE
                STA        JMPTAB,Y            ;MOVE IT
                INY
                DEX
                BNE        MOVEVECT
                FIN
GOODINIT        EQU        *
                LDA        #0                  ;RETCODE=GOOD, IF YOU CARE
                CLC                            ;SAY 'GOOD INIT'
                BCC        EXIT                ;(ALWAYS TAKEN)
                DO         REV0ROM
INITERR         EQU        *
                SEC                            ;SAY 'BAD INIT'
* FALL THRU TO EXIT
                FIN
                PAGE
                REP        40
* -- EXIT PATH --
                REP        40
*
EXIT            EQU        *
                PHA                            ;SAVE RETURN CODE
*
* UPDATE UPTIME BY 50 MS (3 SECTOR-TIMES)
*  TO ACCOUNT FOR READ/WRITE TIME:
*
                LDA        D.COMMAND           ;GET COMMAND
                CMP        #2                  ;SENSE OR INIT?
                BCS        EXIT2               ;=>YES, NO TIME USED UP
                LDA        #2                  ;TIME=50 MS (2 UNITS)
                JSR        ADDTIME             ;BUMP UPTIME(S)
*
```

```
* RESTORE CALLER ENVIRONMENT:
*
EXIT2           EQU        *
                LDA        E.REG                ;GET CURRENT STATE
                AND        #$20                 ; OF THE SCREEN
                ORA        ESAVE                ;MERGE WITH CALLER STATE
                STA        E.REG
                JSR        FIXIRQ               ;RE-ENABLE IRQ IF OK
                LDA        MOTOROFF             ;START MOTOR-OFF TIMEOUT
                PLA                             ;RESTORE RETURN CODE
                DO         TEST                 ;IF TEST, NO SYSERR
                RTS
                ELSE
                BNE        GOERR                ;=>ERROR RETURN VIA SYSERR
                CLC
                RTS                             ;GOOD RETURN W/CARRY CLEAR
GOERR           EQU        *
                JSR        SYSERR               ;RETURN VIA SYSERR
                FIN

                CHN        DISK3.SIO.SRC

**************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.WRT.SRC
**************************************************************************

```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.DEVMGR.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: DEVMGR.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1 DEVICE MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGDMGR
ZZORG           EQU        *
                MSB        OFF
                REP        100
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        100
*
* DEVICE MANAGER (VERSION = 1.10   )
*                (DATE    = 8/04/81)
*
* THIS MODULE IS RESPONSIBLE FOR CALLING THE CORRECT DEVICE
* DRIVER WHEN A D.READ...D.INIT SYSTEM CALL IS MADE.
* (NOTE:  D.OPEN,D.CLOSE AND D.INIT ARE ONLY CALLABLE FROM
* INSIDE THE OPERATING SYSTEM).  D.INFO AND GET.DNUM CALLS
* ARE HANDLED INSIDE THIS MODULE. REPEAT.IO BYPASSES THIS MODULE.
                REP        100
*
                ENTRY      DMGR
*
                ENTRY      MAX.DNUM
                ENTRY      SDT.SIZE
                ENTRY      SDT.DIBL
                ENTRY      SDT.DIBH
                ENTRY      SDT.ADRL
                ENTRY      SDT.ADRH
                ENTRY      SDT.BANK
                ENTRY      SDT.UNIT
                ENTRY      BLKD.SIZE
                ENTRY      BLKDLST
*
                EXTRN      SYSERR
                EXTRN      SERR
                EXTRN      NODNAME
                EXTRN      BADDNUM
                EXTRN      SYSDEATH
                EXTRN      BADSYSCALL
```

```
*
                EXTRN      SXPAGE
*
E.REG           EQU        $FFDF                ; ENVIRONMENT REGISTER
B.REG           EQU        $FFEF                ; BANK REGISTER
                PAGE
                REP        100
*
* SYSTEM DEVICE TABLE (SDT)
*
* CONTAINS THE ADDRESS OF EACH DRIVER'S DIB (SDT.DIB), THE
* ADDRESS OF EACH DRIVER'S ENTRY POINT (SDT.ADR), AND THE
* UNIT # OF EACH DRIVER (SDT.UNIT).  THE TABLE IS INDEXED
* BY DEVICE NUMBER.  ENTRY 0 IS RESERVED FOR FUTURE USE.
*
                REP        100
*
SDT.SIZE        EQU        25
*
MAX.DNUM        DS         1                    ;MAX DEV NUMBER IN SYSTEM+1
SDT.DIBL        DS         SDT.SIZE             ;ADR OF DEVICE INFORMATION BLOCK
SDT.DIBH        DS         SDT.SIZE
*
SDT.ADRL        DS         SDT.SIZE             ;ADR OF ENTRY POINT
SDT.ADRH        DS         SDT.SIZE
*
SDT.BANK        DS         SDT.SIZE             ;BANK # OF DEVICE
*
SDT.UNIT        DS         SDT.SIZE             ;UNIT  # OF DRIVER
*
                REP        100
* BLOCK DEVICE LIST TABLE
*
BLKD.SIZE       EQU        13
BLKDLST         DFB        $00
                DS         BLKD.SIZE-1
                PAGE
                REP        100
*
* DATA DECLARATIONS
*
                REP        100
*
D.TPARMX        EQU        $C0
REQCODE         EQU        D.TPARMX
*
* D.READ/WRITE/CTRL/STATUS/OPEN/CLOSE/INIT/REPEAT PARMS
*
DNUM            EQU        D.TPARMX+1
*
```

```
* D.INFO PARMS
*
I.DNUM          EQU        D.TPARMX+1
I.DNAME         EQU        D.TPARMX+2
I.DLIST         EQU        D.TPARMX+4
I.LENGTH        EQU        D.TPARMX+6
*
* GET.DEV.NUM PARMS
*
G.DNAME         EQU        D.TPARMX+1
G.DNUM          EQU        D.TPARMX+3
*
* SDT ENTRY (=DIB) FIELDS
*
DIB.SLOT        EQU        $11                  ;DIB'S DEVICE SLOT FIELD
DIB.DTYPE       EQU        $13                  ;DIB'S DEVICE TYPE FIELD
*
SDTP            EQU        D.TPARMX+$10         ; PTR TO CURRENT SDT ENTRY
                PAGE
                REP        100
*
* DEVICE MANAGER (MAIN ENTRY POINT)
*
                REP        100
DMGR            EQU        *
*
                LDA        REQCODE
                CMP        #4
                BCC        DRIVER               ; D.READ/WRITE/CTRL/STATUS CALL
                BNE        DM000
                JMP        GET.DNUM             ; GET.DEV.NUM CALL
DM000           CMP        #5
                BEQ        D.INFO               ; D.INFO CALL
                CMP        #$A
                BCC        DRIVER               ; D.OPEN/CLOSE/INIT
                LDA        #BADSYSCALL          ; ELSE FATAL ERROR
                JSR        SYSDEATH             ; EXIT
                PAGE
                REP        100
* D.READ/WRITE/CTRL/STATUS/OPEN/CLOSE/INIT CALLS
* "JSR" TO DEVICE DRIVER
                REP        100
DRIVER          EQU        *
*
                LDX        DNUM                 ; GET DNUM SYSCALL PARM
                BEQ        DM005                ; WITHIN BOUNDS?
                CPX        MAX.DNUM             ;       "
                BCC        DM010
*
* DNUM TOO LARGE
```

```
*
DM005           LDA        #>BADDNUM            ; INVALID DEVICE NUMBER
                JSR        SYSERR               ;   ERROR EXIT
*
* MAP DEV# TO UNIT#
*
DM010           LDA        SDT.UNIT,X
                STA        DNUM
*
* "JSR" TO DEVICE DRIVER VIA JMP TABLE
*
                LDA        B.REG                ; STACK B.REG
                PHA
                LDA        #<DM.RTN-1           ; STACK RETURN ADDRESS
                PHA
                LDA        #>DM.RTN-1
                PHA
*
                LDA        SDT.BANK,X           ; SELECT RAM BANK
                STA        B.REG
                LDA        SDT.ADRH,X           ; STACK DRIVER ENTRY POINT ADDRESS
                PHA
                LDA        SDT.ADRL,X
                PHA
*
                LDA        E.REG                ; SWITCH IN I/O BANK
                ORA        #$40
                STA        E.REG
                RTS                             ; AND, "JSR" TO DEVICE DRIVER
*
DM.RTN          LDA        E.REG                ; SWITCH OUT I/O BANK
                AND        #$BF
                STA        E.REG
                PLA                             ; RESTORE B.REG
                STA        B.REG
                SEC
                LDA        SERR                 ; RETRIEVE ERROR CODE
                BNE        DM017                ; ENSURE CARRY CLEARED IF NO ERROR
                CLC
DM017           RTS                             ; AND, EXIT TO CALLER
                PAGE
                REP        100
* D.INFO(IN.DNUM, OUT.DNAME, OUT.DEVLIST, IN.LENGTH) SYSTEM CALL
                REP        100
D.INFO          EQU        *
*
                LDX        I.DNUM               ; GET DNUM PARM
                BEQ        DM020                ; WITHIN BOUNDS?
                CPX        MAX.DNUM             ;      "
                BCC        DM030
```

```
DM020           LDA        #>BADDNUM           ; NO, DNUM TOO LARGE
                JSR        SYSERR              ; ERROR EXIT
*
* MOVE PARMS FM SDT ENTRY (DEV INFO BLOCK) TO CALLER'S
* PARM LIST
*
DM030           JSR        SETUP.SDT           ; SET UP ZPAGE PTR TO SDT ENTRY
*
* OUPUT DNAME PARM
*
                LDA        (SDTP),Y            ; LOAD PARM'S BYTE COUNT
                TAY
DM040           LDA        (SDTP),Y
                STA        (I.DNAME),Y
                DEY
                BPL        DM040
*
* OUTPUT DEVINFO PARM (SLOT,UNIT,DEVID,PRODCODE)
*
                LDA        #DIB.SLOT
                CLC                            ; ADVANCE SDTP TO 2ND PARM IN SDT
                ADC        SDTP
                STA        SDTP
                BCC        DM045
                INC        SDTP+1
DM045           LDY        I.LENGTH            ; LOAD BYTE COUNT
                BEQ        DM.EXIT             ; IF 0 THEN DONE
                DEY
                CPY        #$B
                BCC        DM050
                LDY        #$A
DM050           LDA        (SDTP),Y
                STA        (I.DLIST),Y
                DEY
                BPL        DM050
*
DM.EXIT         CLC
                RTS                            ; NORMAL EXIT
                PAGE
                REP        100
* GET.DEV.NUM(IN.DNAME; OUT.DNUM) SYSTEM CALL
                REP        100
*
GET.DNUM        EQU        *
*
                LDX        #1                  ; SETUP PTR TO 1ST SDT ENTRY
*
DM070           JSR        SETUP.SDT           ; SET UP ZPAGE PTR TO SDT ENTRY
*
                LDA        (SDTP),Y            ; COMPARE DNAME LENGTHS
                CMP        (G.DNAME),Y
                BNE        NXTSDT
*
                TAY                             ; LENGTHS MATCH, NOW COMPARE CHARS
DM080           LDA        (G.DNAME),Y
                CMP        #$60
                BCC        DM090
                AND        #$DF                 ; UPSHIFT
DM090           CMP        (SDTP),Y
                BNE        NXTSDT
                DEY
                BNE        DM080
*
                TXA                             ; CHARS MATCH
                LDY        #0
                STA        (G.DNUM),Y           ; OUTPUT DEV NUM PARM
                LDY        #DIB.DTYPE           ; SET "N" FLAG IN STATUS REG.
                LDA        (SDTP),Y             ; N=1(BLOCK DEVICE) N=0(CHAR DEVICE)
                CLC
                RTS                             ; NORMAL EXIT
*
NXTSDT          INX                             ; LAST SDT ENTRY?
                CPX        MAX.DNUM
                BCC        DM070
*
                LDA        #>NODNAME            ; ERROR, DNAME NOT FOUND IN SDT
                JSR        SYSERR               ;  RETURN TO CALLER
                PAGE
                REP        100
* SETUP.SDT(IN.X=DNUM, OUT.SDTP, B.REG, Y=0)   X="UNCHANGED"
                REP        100
SETUP.SDT       EQU        *
                LDA        SDT.DIBL,X           ; SET UP ZPAGE PTR TO SDT ENTRY
                STA        SDTP                 ; (POINTS TO DNAME FIELD)
                LDA        SDT.DIBH,X
                STA        SDTP+1
                LDA        SDT.BANK,X
                STA        B.REG
                LDY        #0
                STY        SXPAGE+SDTP+1
                RTS
*
                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENDMGR
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR DEVMGR"
                FIN

*************************************************************************
```

```
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DEVMGR.SRC
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.DISK3.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.DATA.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
* GENERAL DATA:
*
PREVUNIT        DS         1                    ;PRIOR UNIT ACCESSED (FOR REPEAT)
PREVCMD         DS         1                    ;PRIOR CMD (FOR REPEAT)
*
ESAVE           DS         1                    ;SAVED E.REG
VBLSAVE         DS         1                    ;SAVED E.IER
INITFLAG        DFB        0                    ;<0 IS INITTED
                DO         REV0ROM
ROMREV          DS         1                    ;0=REV0, <>0=REV1
                FIN
*
* MOTOR-UP TIMES PER COMMAND
T50MS           EQU        $02                  ; 50MS FOR MONTIMEH
T200MS          EQU        $08                  ;200 MS FOR MONTIMEH
T1SEC           EQU        $27                  ;1-SEC FOR MONTIMEH
*
MTIMES          DFB        T200MS,T1SEC,T50MS   ;READ,WRITE,SENSE
*
                REP        40
* DRIVE TABLES:
*
DRIVESEL        DS         4                    ;NONZERO IF SELECTED
*
UPTIME          DS         4                    ;MOTOR RUNTIME SINCE STARTED
DRVTRACK        DS         4                    ;CURRENT HEAD POSITION
                PAGE
                DO         REV0ROM              ;ONLY IF SUPPORTING IT!
* JUMP TABLE TO MONITOR ROUTINES.
*  THIS TABLE FILLED IN BY 'INIT'.
*
JMPTAB          EQU        *
RDADR           JMP        *
READ            JMP        *
WRITE           JMP        *
SEEK            JMP        *
MSWAIT          JMP        *
PRENIB          JMP        *
POSTNIB         JMP        *
```

```
*
REV0            EQU        *                    ;REV0 ADDRESSES
                JMP        $F1BD                ;RDADR
                JMP        $F148                ;READ
                JMP        $F219                ;WRITE
                JMP        $F400                ;SEEK
                JMP        $F456                ;MSWAIT
                JMP        $F2C6                ;PRENIB
                JMP        $F311                ;POSTNIB
VSIZE           EQU        *-REV0               ;TABLE SIZE
*
REV1            EQU        *                    ;REV1 ADDRESSES
                JMP        $F1B9                ;RDADR
                JMP        $F148                ;READ
                JMP        $F216                ;WRITE
                JMP        $F400                ;SEEK
                JMP        $F456                ;MSWAIT
                JMP        $F2C4                ;PRENIB
                JMP        $F30F                ;POSTNIB
                ELSE                            ;FOR REV1 WE USE EQUATES
RDADR           EQU        $F1B9                ;RDADR
READ            EQU        $F148                ;READ
WRITE           EQU        $F216                ;WRITE
SEEK            EQU        $F400                ;SEEK
MSWAIT          EQU        $F456                ;MSWAIT
PRENIB          EQU        $F2C4                ;PRENIB
POSTNIB         EQU        $F30F                ;POSTNIB
                FIN

ZZEND           EQU        *
ZZLEN           EQU        *-ZZORG
                IFNE       ZZLEN-LENDISK3
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR DISK3"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.DATA.SRC
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.FMGR.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: FMGR.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1  FILE MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGFMGR
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* FILE MANAGER (VERSION = 1.10   )
*              (DATE    = 8/04/81)
*
* THIS MODULE IS ENTERED FROM THE SYSTEM CALL MANAGER, AND
* IS RESPONSIBLE FOR SWITCHING TO EITHER THE BLOCK FILE
* MANAGER, OR THE CHARACTER FILE MANAGER.
*
                REP        60
*
                ENTRY      FMGR
                ENTRY      LEVEL
*
                EXTRN      BFMGR
                EXTRN      CFMGR
                EXTRN      SYSERR
                EXTRN      SERR
                EXTRN      BADPATH
                EXTRN      FNFERR
                EXTRN      LVLERR
*
F.TPARMX        EQU        $A0                  ; LOC OF FILE SYSTEM CALL PARMS
OPEN            EQU        $8
CLOSE           EQU        $C
SETLEVEL        EQU        $12
GETLEVEL        EQU        $13
F.REQCODE       EQU        F.TPARMX
F.LEVEL         EQU        F.TPARMX+$1
PATHNAME        EQU        F.TPARMX+$1
REFNUM          EQU        F.TPARMX+$1
```

```
PERIOD          EQU        $2E
LEVEL           DFB        $1
                PAGE
                REP        60
*
* FILE MANAGER
*
                REP        60
FMGR            EQU        *
*
                LDA        F.REQCODE
                CMP        #OPEN
                BCC        FMGR010
                BEQ        FMGR020
                CMP        #CLOSE
                BCC        FMGR030
                BEQ        FMGR040
                CMP        #SETLEVEL
                BEQ        SLEVEL
                CMP        #GETLEVEL
                BEQ        GLEVEL
*
FMGR010         JMP        BFMGR                ; EXIT
*
FMGR020         LDY        #1
                LDA        (PATHNAME),Y
                CMP        #PERIOD
                BNE        FMGR010
                JSR        CFMGR
                BCC        FMGR024
                LDA        SERR
                CMP        #FNFERR
                BEQ        FMGR026
FMGR024         RTS                             ; EXIT
*
FMGR026         LDA        #0
                STA        SERR
                JMP        BFMGR                ; EXIT
*
FMGR030         LDA        REFNUM
FMGR031         BPL        FMGR010
                JMP        CFMGR                ; EXIT
*
FMGR040         LDA        REFNUM
                BNE        FMGR031
                JSR        BFMGR                ; CLOSE (0)
                JMP        CFMGR                ; EXIT
*
SLEVEL          LDA        F.LEVEL
                BEQ        LVL.ERR
                CMP        #4
                BCS        LVL.ERR
                STA        LEVEL
                RTS
LVL.ERR         LDA        #LVLERR
                JSR        SYSERR
*
GLEVEL          LDY        #0
                LDA        LEVEL
                STA        (F.LEVEL),Y
                RTS
*
                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENFMGR
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR FMGR"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: FMGR.SRC
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.MMGR.A.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.A.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1  MEMORY MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGMEMMG
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* MEMORY MANAGER (VERSION = 1.1O   )
*                (DATE    = 8/04/81)
*
* THIS MODULE CONTAINS ALL OF THE MEMORY MANAGEMENT SYSTEM
* CALLS SUPPORTED BY THE SARA OPERATING SYSTEM.  IT IS
* ALSO CALLED BY THE BUFFER MANAGER.
*
                REP        60
*
                ENTRY      MMGR
*
                ENTRY      ST.CNT
                ENTRY      ST.ENTRY
                ENTRY      ST.FREE
                ENTRY      ST.FLINK
                ENTRY      VRT.LIM
*
                EXTRN      SYSERR
                EXTRN      BADSCNUM
                EXTRN      BADBKPG
                EXTRN      SEGRQDN
                EXTRN      SEGTBLFULL
                EXTRN      BADSEGNUM
                EXTRN      SEGNOTFND
                EXTRN      BADSRCHMODE
                EXTRN      BADCHGMODE
                EXTRN      BADPGCNT
                PAGE
                REP        60
```

```
*
* SEGMENT TABLE
* (NOTE: ENTRY 0 IS NOT USED)
*
                REP        60
*
ST.FREE         DS         1                    ; PTR TO FIRST FREE SEG TABLE ENTRY
ST.ENTRY        DS         1                    ; PTR TO HIGHEST ALLOC SEG TABLE ENTRY
ST.SIZ          EQU        7
ST.CNT          EQU        32
ST.TBL          DS         ST.SIZ*ST.CNT
ST.BLINK        EQU        ST.TBL               ; BACK LINK TO PREV ALLOC SEG ENTRY
ST.FLINK        EQU        ST.BLINK+ST.CNT      ; FORWARD LINK     "
ST.BASEL        EQU        ST.FLINK+ST.CNT      ; BASE BANK/PAGE
ST.BASEH        EQU        ST.BASEL+ST.CNT
ST.LIML         EQU        ST.BASEH+ST.CNT      ; LIMIT BANK/PAGE
ST.LIMH         EQU        ST.LIML+ST.CNT
ST.ID           EQU        ST.LIMH+ST.CNT       ; SEG ID
                PAGE
                REP        60
*
* DATA DECLARATIONS
*
                REP        60
*
ZPAGE           EQU        $40                  ; BEGINNING OF ZPAGE TEMP SPACE FOR MEMORY MANAGER
VRT.BASE        EQU        $0                   ; INTERNAL BK/PG PTR TO LOWEST VIRT PAGE
VRT.LIM         EQU        ZPAGE+$0             ; &$1, INTERNAL BK/PG PTR TO HIGHEST VIRT PAGE
PHY1BASE        EQU        $0780                ; BANK "F",PAGE "0"
PHY1LIM         EQU        $079F                ; BANK "F",PAGE "1F"
PHY2BASE        EQU        $0820                ; BANK "10",PAGE "A0"
PHY2LIM         EQU        $087F                ; BANK "10",PAGE "FF"
*
* REQUEST.SEG DATA DECLARATIONS
*
M.TPARMX        EQU        $60                  ; BEGINNING ADDRESS OF MMGR SOS CALL PARMS
M.RQCODE        EQU        M.TPARMX
RQ.BASE         EQU        M.TPARMX+1           ; BASE.BANK/PAGE
RQ.LIM          EQU        M.TPARMX+3           ; LIMIT.BANK/PAGE
RQ.ID           EQU        M.TPARMX+5
RQ.NUM          EQU        M.TPARMX+6
*
RQ.REGION       EQU        ZPAGE+$2             ;VRT(0),PHY0(1),PHY1(2)
*
* FIND.SEG DATA DECLARATIONS
*
SRCHMODE        EQU        M.TPARMX+1           ; SEARCH MODE (0,1,2)
F.ID            EQU        M.TPARMX+2           ; SEG ID
F.PGCT          EQU        M.TPARMX+3           ; PAGE COUNT (LO
FX.PGCT         EQU        ZPAGE+$3             ; &$4, INTERNAL PAGE COUNT
```

```
F.BASE          EQU        M.TPARMX+5           ; BASE.BANK/PAGE
F.LIM           EQU        M.TPARMX+7           ; LIMIT.BANK/PAGE
F.NUM           EQU        M.TPARMX+9           ; SEG NUM
F.ERR           EQU        ZPAGE+$5             ; ERROR FLAG
TRUE            EQU        $80
FALSE           EQU        $0
CFS.PGCT        EQU        ZPAGE+$6             ; &7, CURRENT FREE SEGMENT'S PAGE COUNT
CFS.BASE        EQU        ZPAGE+$8             ; &9,        "             BASE.BANK/PAGE
CFS.LIM         EQU        ZPAGE+$A             ; &$B,       "             LIMIT.BANK/PAGE
CFS.BLINK       EQU        ZPAGE+$C             ;          "             BACK LINK
CFS.BASE0       EQU        ZPAGE+$D             ; &$E,     "             BASE (SMODE=0)
CFS.BASE1       EQU        ZPAGE+$F             ; &$10,     "             BASE (SMODE=1)
CFS.NEXT        EQU        ZPAGE+$11            ;          "             NEXT ENTRY
CFS.PREV        EQU        ZPAGE+$12            ;          "             PREV ENTRY
CFS.PTR         EQU        ZPAGE+$13            ; &$14     "             POINTER TO NXT FREE PG
BFS.PGCT        EQU        ZPAGE+$15            ; &$16, BIGGEST FREE SEGMENT'S PAGE COUNT
BFS.BASE        EQU        ZPAGE+$17            ; &$18     "             BASE.BANK/PAGE
BFS.LIM         EQU        ZPAGE+$19            ; &$1A     "             LIMIT.BANK/PAGE
BFS.BLINK       EQU        ZPAGE+$1B            ;          "             BACK LINK
*
* CHANGE.SEG DATA DECLARATIONS
*
CHG.NUM         EQU        M.TPARMX+1           ; SEGNUM PARM
CHG.MODE        EQU        M.TPARMX+2           ; CHANGE MODE PARM
CHG.PGCT        EQU        M.TPARMX+3           ; PAGE COUNT PARM
CHG.PGCTX       EQU        ZPAGE+$1C            ; &$1D, INTERNAL STORE FOR PGCT
CHG.NEW         EQU        ZPAGE+$1E            ; &$1F, BANK/PAGE OF SEG'S NEW LIMIT OR BASE
*
* GET.SEG.INFO DATA DECLARATIONS
*
GSI.NUM         EQU        M.TPARMX+1
GSI.BASE        EQU        M.TPARMX+2
GSI.LIM         EQU        M.TPARMX+4
GSI.PGCT        EQU        M.TPARMX+6
GSI.ID          EQU        M.TPARMX+8
*
* GET.SEG.NUM DATA DECLARATIONS
*
GSN.BKPG        EQU        M.TPARMX+1
GSN.NUM         EQU        M.TPARMX+3
*
* RELEASE.SEG DATA DECLARATIONS
*
RLS.NUM         EQU        M.TPARMX+1           ; SEG NUM
*
* REGION - DATA DECLARATIONS
*
RGN.BKPG        DS         2                    ; TEMP CONTAINER FOR BANK/PAGE
                PAGE
                REP        60
```

```
*
* MMGR
*
* THIS ROUTINE IS THE MAIN ENTRANCE TO THE MEMORY MANAGER
* MODULE.  IT FUNCTIONS AS A SWITCH, BASED UPON THE RECEIVED
* REQUEST CODE, TO TRANSFER CONTROL TO THE ROUTINE THAT
* HANDLES THE SPECIFIC SYSTEM CALL.
*
                REP        60
*
MMGR            EQU        *
                LDA        M.RQCODE
                BEQ        MMGR010             ; "REQ.SEG"
                CMP        #1
                BEQ        MMGR020             ; "FIND.SEG"
                CMP        #2
                BEQ        MMGR030             ; "CHANGE.SEG"
                CMP        #3
                BEQ        MMGR040             ; "GET.SEG.INFO"
                CMP        #4
                BEQ        MMGR050             ; "GET.SEG.NUM"
                CMP        #5
                BEQ        MMGR060             ; "RELEASE.SEG"
*
                LDA        #BADSCNUM
                JSR        SYSERR
*
MMGR010         JMP        REQ.SEG
MMGR020         JMP        FIND.SEG
MMGR030         JMP        CHG.SEG
MMGR040         JMP        GET.SEG.INFO
MMGR050         JMP        GET.SEG.NUM
MMGR060         JMP        RELEASE.SEG
                PAGE
                REP        60
*
* REQUEST.SEG(IN.BASE.BANKPAGE,LIMIT.BANKPAGE,SEGID; OUT.SEGNUM)
*
                REP        60
*
REQ.SEG         EQU        *
*
* CONVERT CALLER'S BASE.BANK/PAGE TO INTERNAL FMT
*
                LDX        RQ.BASE
                LDY        RQ.BASE+1
                JSR        CNVRT.IBP
                BCC        RQ005
*
RQ.ERR          RTS                            ; ERR EXIT - INVALID BANK/PAGE
```

```
*
RQ005           STX        RQ.BASE
                STY        RQ.BASE+1
                STA        RQ.REGION
*
* CONVERT CALLER'S LIMIT.BANK/PAGE TO INTERNAL FMT
*
                LDX        RQ.LIM
                LDY        RQ.LIM+1
                JSR        CNVRT.IBP
                BCS        RQ.ERR               ; ERR - INVALID BANK/PAGE
                STX        RQ.LIM
                STY        RQ.LIM+1
*
* IF BASE AND LIMIT ARE IN DIFFERENT REGIONS THEN ERR
*
                CMP        RQ.REGION
                BNE        RQ.ERR1              ; ERR - INVALID BANK/PAGE PAIR
* IF CALLER'S BASE > LIMIT THEN ERR
*
                LDA        RQ.LIM
                CMP        RQ.BASE
                LDA        RQ.LIM+1
                SBC        RQ.BASE+1
                BCC        RQ.ERR1              ; ERR - INVALID BANK/PAGE PAIR
*
* PREV SEGNUM:=NULL; NEXT SEGNUM:=FIRST ENTRY
*
                LDX        #0
                LDY        ST.ENTRY             ; NOTE: PREV/NEXT CARRIED IN X & Y REGISTERS
*
* IF NO SEGS IN SEG TABLE THEN ALLOCATE REQUESTED SEG
*
                BEQ        RQ030
*
* IF FIRST SEG IN SEG TABLE BELOW REQUESTED SEG
* THEN ALLOCATE SEG
*
                LDA        ST.LIML,Y
                CMP        RQ.BASE
                LDA        ST.LIMH,Y
                SBC        RQ.BASE+1
                BCC        RQ030
*
* ADVANCE TO NEXT SEG ENTRY
*
RQ010           TYA
                TAX
                LDA        ST.FLINK,Y
                TAY
```

```
*
* IF THERE IS NO NEXT SEG ENTRY
*   IF REQUESTED SEG IS BELOW PREV SEG
*      THEN ALLOCATE REQ SEG
*      ELSE ERR
*
                BNE        RQ020
                LDA        RQ.LIM
                CMP        ST.BASEL,X
                LDA        RQ.LIM+1
                SBC        ST.BASEH,X
                BCC        RQ030
*
                BCS        RQ.ERR2              ; ERR - SEGMENT REQUEST DENIED
*
* IF REQUESTED LIMIT >= PREV SEG'S BASE THEN ERR
*
RQ020           LDA        RQ.LIM
                CMP        ST.BASEL,X
                LDA        RQ.LIM+1
                SBC        ST.BASEH,X
                BCS        RQ.ERR2              ; ERR - SEGMENT REQUEST DENIED
*
* IF REQUESTED BASE > NEXT SEG'S LIMIT
*    THEN ALLOCATE REQUESTED SEGMENT
*
                LDA        ST.LIML,Y
                CMP        RQ.BASE
                LDA        ST.LIMH,Y
                SBC        RQ.BASE+1
                BCS        RQ010                ; NO, ADVANCE TO NEXT SEGMENT
*
RQ030           TXA                             ; ALLOCATE REQUESTED SEGMENT
                JSR        GET.FREE
                BCS        RQ.ERR3              ; ERR - SEG TABLE FULL
*
* ENTER BASE,LIMIT AND ID IN NEW SEG ENTRY
*
                TAX
                LDA        RQ.BASE
                STA        ST.BASEL,X
                LDA        RQ.BASE+1
                STA        ST.BASEH,X
*
                LDA        RQ.LIM
                STA        ST.LIML,X
                LDA        RQ.LIM+1
                STA        ST.LIMH,X
*
                LDA        RQ.ID
                STA        ST.ID,X
*
* RETURN NEW SEG NUM TO CALLER AND RETURN
*
                LDY        #0
                TXA
                STA        (RQ.NUM),Y
*
                CLC
                RTS                             ; NORMAL EXIT
*
RQ.ERR1         LDA        #BADBKPG
                JSR        SYSERR               ; ERR EXIT
RQ.ERR2         LDA        #SEGRQDN
                JSR        SYSERR               ; ERR EXIT
*
RQ.ERR3         LDA        #SEGTBLFULL
                JSR        SYSERR               ; ERR EXIT
                PAGE
                REP        60
*
* FIND.SEG(IN.SRCHMODE,SEGID; INOUT.PAGECT;
*          OUT.BASE.BKPG,LIMIT.BKPG,SEGNUM)
*
                REP        60
*
FIND.SEG        EQU        *
*
* RETRIEVE PAGE COUNT PARAMETER AND CLEAR ERR FLAG
*
                LDY        #0
                LDA        (F.PGCT),Y
                STA        FX.PGCT
                INY
                LDA        (F.PGCT),Y
                STA        FX.PGCT+1
*
                BNE        FIND001
                LDA        FX.PGCT
                BNE        FIND001
                LDA        #BADPGCNT            ; ERR, PAGECT=0, EXIT
                JSR        SYSERR
*
FIND001         LDA        #FALSE
                STA        F.ERR
*
* IF SEARCH MODE>2 THEN ERR
*
                LDA        SRCHMODE
                CMP        #3
                BCC        FIND005
                LDA        #BADSRCHMODE
                JSR        SYSERR              ; ERR EXIT
*
* INITIALIZE NEXT FREE SEGMENT SUBROUTINE,
* AND BIGGEST FREE SEGMENT PAGE COUNT
*
FIND005         JSR        NXTFRSEG.I
                LDA        #0
                STA        BFS.PGCT
                STA        BFS.PGCT+1
*
* GET NEXT FREE SEGMENT
*
FIND010         JSR        NXTFRSEG
                BCC        FIND015             ; PROCESS FREE SEGMENT
*
* NO MORE FREE SEGMENTS LEFT
* RETURN BIGGEST FREE SEGMENT FOUND
* ALONG WITH ERR
*
                LDA        #TRUE
                STA        F.ERR
                LDX        #0                  ; SEG#:=0
                JMP        FIND070
*
* FREE SEGMENT FOUND.
*   IF FREE SEGMENT > BIGGEST FREE SEGMENT THEN BFS:=CFS
*
FIND015         LDA        BFS.PGCT
                CMP        CFS.PGCT
                LDA        BFS.PGCT+1
                SBC        CFS.PGCT+1
                BCS        FIND030
*
                LDX        #6
FIND020         LDA        CFS.PGCT,X
                STA        BFS.PGCT,X
                DEX
                BPL        FIND020
*
* IF BFS.PGCT<F.PGCT THEN GET NEXT FREE SEGMENT
*
FIND030         LDA        BFS.PGCT
                CMP        FX.PGCT
                LDA        BFS.PGCT+1
                SBC        FX.PGCT+1
                BCC        FIND010
*
* BFS.BASE:=BFS.LIM-FX.PGCT+1
* BFS.PGCT:=FX.PGCT
*
                LDA        BFS.LIM
                SBC        FX.PGCT
                STA        BFS.BASE
                LDA        BFS.LIM+1
                SBC        FX.PGCT+1
                STA        BFS.BASE+1
                INC        BFS.BASE
                BNE        FIND050
                INC        BFS.BASE+1
*
FIND050         LDA        FX.PGCT
                STA        BFS.PGCT
                LDA        FX.PGCT+1
                STA        BFS.PGCT+1
*
* DELINK ENTRY FROM FREE LIST, AND LINK
* IT INTO SEGMENT LIST
*
                LDA        BFS.BLINK
                JSR        GET.FREE
                BCC        FIND060
                RTS                             ; ERR - SEG TABLE FULL
*
* ST.ID(NEW):=F.ID
* ST.BASE(NEW):=BFS.BASE
* ST.LIM(NEW):=BFS.LIM
*
FIND060         TAX
                LDA        F.ID
                STA        ST.ID,X
*
                LDA        BFS.BASE
                STA        ST.BASEL,X
                LDA        BFS.BASE+1
                STA        ST.BASEH,X
*
                LDA        BFS.LIM
                STA        ST.LIML,X
                LDA        BFS.LIM+1
                STA        ST.LIMH,X
*
* RETURN SEGNUM, PAGE COUNT, BASE BANK/PAGE, AND LIMIT BANK/PAGE
* TO CALLER
FIND070         LDY        #0
                TXA
                STA        (F.NUM),Y
*
                LDA        BFS.PGCT
                STA        (F.PGCT),Y
                INY
                LDA        BFS.PGCT+1
                STA        (F.PGCT),Y
*
                LDX        BFS.BASE
                LDY        BFS.BASE+1
                JSR        CNVRT.XBP
                TYA
                LDY        #1
                STA        (F.BASE),Y
                DEY
                TXA
                STA        (F.BASE),Y
*
                LDX        BFS.LIM
                LDY        BFS.LIM+1
                JSR        CNVRT.XBP
                TYA
                LDY        #1
                STA        (F.LIM),Y
                DEY
                TXA
                STA        (F.LIM),Y
*
                LDA        F.ERR                 ; IF ERR FLAG TRUE THEN REPORT IT.
                BNE        FIND.ERR
*
                CLC
                RTS                              ; NORMAL EXIT
*
FIND.ERR        LDA        #SEGRQDN
                JSR        SYSERR                ; ERR EXIT

                CHN        MEMMGR.B.SRC

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.A.SRC
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.MMGR.B.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.B.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        60
*
* NEXT FREE SEGMENT - INITIALIZATION
*
* INPUT:  SEGMENT TABLE
* OUTPUT: CFS.PTR "1ST FREE BANK/PAGE IN VIRTUAL MEMORY
*         CFS.PREV "PREVIOUS SEGMENT EXAMINED"
*         CFS.NEXT "SEGMENT FOLLOWING CFS.PREV"
* ERROR:  NONE (IF NO FREE BK/PG FOUND, THEN CFS.PTR="FFFF")
*
                REP        60
*
NXTFRSEG.I      EQU        *
*
* CFS.PTR := VRT.LIM
* CFS.PREV := 0
* CFS.NEXT := ST.ENTRY
*
                LDA        >VRT.LIM
                STA        CFS.PTR
                LDA        >VRT.LIM+1
                STA        CFS.PTR+1
*
                LDA        #0
                STA        CFS.PREV
*
                LDX        ST.ENTRY
                STX        CFS.NEXT
*
* L0:  IF CFS.NEXT=0 THEN DONE
*
FRSGI010        BEQ        FRSGI.EXIT
*
* IF ST.LIM(CFS.NEXT)<=VRT.LIM THEN GOTO L1
*
                LDA        >VRT.LIM
                CMP        ST.LIML,X
                LDA        >VRT.LIM+1
                SBC        ST.LIMH,X
                BCS        FRSGI020
*
* CFS.PREV:=CFS.NEXT
* CFS.NEXT:=ST.FLINK(CFS.NEXT)
* GOTO L0
*
                STX        CFS.PREV
                LDA        ST.FLINK,X
                TAX
                STX        CFS.NEXT
                JMP        FRSGI010
*
* L1:  IF ST.LIM(CFS.NEXT)<VRT.LIM THEN DONE
*
FRSGI020        LDA        ST.LIML,X
                CMP        >VRT.LIM
                LDA        ST.LIMH,X
                SBC        >VRT.LIM+1
                BCC        FRSGI.EXIT
*
*
                JSR        NXTFRPG
*
FRSGI.EXIT      RTS                                 ; NORMAL EXIT
                PAGE
                REP        60
*
* NEXT FREE SEGMENT
*
* INPUT:  SEG TABLE
* OUTPUT: CFS.BLINK
*         CFS.BASE
*         CFS.LIMIT
*         CFS.PGCT
* OWN:    CFS.PREV
*         CFS.NEXT
*         CFS.PTR
*
* BUILDS A CANDIDATE FREE SEGMENT, WHOSE LIMIT BANK/PAGE =
* THE CURRENT FREE PAGE (CFS.PTR).
*
                REP        60
*
NXTFRSEG        EQU        *
*
* IF CFS.PTR="FFFF" THEN EXIT
*
                LDA        CFS.PTR+1
                BPL        FRSG010
*
```

```
                SEC
                RTS                                 ; EXIT - NO MORE FREE SEGMENTS LEFT
*
* CFS.BLINK:=CFS.PREV
* CFS.LIM:=CFS.PTR
*
FRSG010         LDA        CFS.PREV
                STA        CFS.BLINK
*
                LDA        CFS.PTR
                STA        CFS.LIM
                LDA        CFS.PTR+1
                STA        CFS.LIM+1
*
* IF CFS.NEXT=0 THEN CFS.BASE:=0
*    ELSE CFS.BASE:=ST.LIM(CFS.NEXT)+1
*
                LDA        CFS.NEXT
                BNE        FRSG020
                LDA        #0
                STA        CFS.BASE
                STA        CFS.BASE+1
                BEQ        FRSG030
*
FRSG020         LDX        CFS.NEXT
                CLC
                LDA        ST.LIML,X
                ADC        #1
                STA        CFS.BASE
                LDA        ST.LIMH,X
                ADC        #0
                STA        CFS.BASE+1
*
* CFS.BASE0:=CFS.LIM AND $FF80
*
FRSG030         LDY        CFS.LIM+1
                STY        CFS.BASE0+1
                LDA        CFS.LIM
                AND        #$80
                STA        CFS.BASE0
*
* CFS.BASE1:=CFS.BASE0-32K
*
                SEC
                SBC        #$80
                STA        CFS.BASE1
                TYA
                SBC        #0
                STA        CFS.BASE1+1
                BCS        FRSG035
                LDA        #0
                STA        CFS.BASE1
                STA        CFS.BASE1+1
*
* IF CFS.BASE>=CFS.BASE0 THEN GOTO L1
*
FRSG035         LDA        CFS.BASE
                CMP        CFS.BASE0
                LDA        CFS.BASE+1
                SBC        CFS.BASE0+1
                BCS        FRSG050
*
* IF SEARCH MODE=0 THEN CFS.BASE:=CFS.BASE0
* GOTO L1
*
                LDA        SRCHMODE
                BNE        FRSG040
                LDA        CFS.BASE0
                STA        CFS.BASE
                LDA        CFS.BASE0+1
                STA        CFS.BASE+1
                JMP        FRSG050
*
* IF CFS.BASE<CFS.BASE1 AND SEARCH MODE=1
*    THEN CFS.BASE:=CFS.BASE1
*
FRSG040         LDA        CFS.BASE
                CMP        CFS.BASE1
                LDA        CFS.BASE+1
                SBC        CFS.BASE1+1
                BCS        FRSG050
*
                LDA        SRCHMODE
                CMP        #1
                BNE        FRSG050
*
                LDA        CFS.BASE1
                STA        CFS.BASE
                LDA        CFS.BASE1+1
                STA        CFS.BASE+1
*
* L1:  CFS.PGCT:=CFS.LIM-CFS.BASE+1
*
FRSG050         SEC
                LDA        CFS.LIM
                SBC        CFS.BASE
                STA        CFS.PGCT
                LDA        CFS.LIM+1
                SBC        CFS.BASE+1
                STA        CFS.PGCT+1
                INC        CFS.PGCT
                BNE        FRSG052
                INC        CFS.PGCT+1
*
* ADVANCE FREE PAGE POINTER TO NEXT FREE PAGE
*
* IF SEARCH MODE<>1 THEN L2:
*
FRSG052         LDA        SRCHMODE
                CMP        #1
                BNE        FRSG060
*
* IF CFS.BASE < CFS.BASE0 THEN CFS.PTR:=CFS.BASE0-1
*
                LDA        CFS.BASE
                CMP        CFS.BASE0
                LDA        CFS.BASE+1
                SBC        CFS.BASE0+1
                BCS        FRSG060
*
                LDY        CFS.BASE0+1
                LDX        CFS.BASE0
                BNE        FRSG055
                DEY
FRSG055         DEX
                STX        CFS.PTR
                STY        CFS.PTR+1
*
                JMP        FRSG070             ; AND EXIT
* L2: CFS.PTR:=CFS.BASE-1
*
FRSG060         SEC
                LDA        CFS.BASE
                SBC        #1
                STA        CFS.PTR
                LDA        CFS.BASE+1
                SBC        #0
                STA        CFS.PTR+1
*
* IF CFS.PTR="FFFF" OR CFS.NEXT=0 THEN EXIT
*
                BCC        FRSG070
                LDA        CFS.NEXT
                BEQ        FRSG070
*
* IF CFS.PTR > ST.LIM(CFS.NEXT) THEN EXIT
*
                LDX        CFS.NEXT
                LDA        ST.LIML,X
                CMP        CFS.PTR
                LDA        ST.LIMH,X
                SBC        CFS.PTR+1
                BCC        FRSG070
*
* OTHERWISE, ADVANCE CFS PTR TO NEXT FREE PAGE BELOW NEXT
* SEGMENT IN SEGMENT LIST
*
                JSR        NXTFRPG
*
FRSG070         CLC
                RTS                             ; EXIT - FREE SEGMENT FOUND
                PAGE
                REP        60
*
* NEXT FREE PAGE
*
* "WALKS" THE FREE PAGE PTR (CFS.PTR) TO THE NEXT FREE PAGE
* IMMEDIATELY BELOW THE CURRENT FREE SEGMENT.
*
                REP        60
*
NXTFRPG         EQU        *
*
* L0: CFS.PTR:=ST.BASE(CFS.NEXT)-1
*     IF CFS.PTR="FFFF" THEN DONE
*
                LDX        CFS.NEXT
                SEC
                LDA        ST.BASEL,X
                SBC        #1
                STA        CFS.PTR
                LDA        ST.BASEH,X
                SBC        #0
                STA        CFS.PTR+1
                BCC        NFRPG.EXIT
*
* CFS.PREV:=CFS.NEXT
* CFS.NEXT:=ST.FLINK(CFS.NEXT)
*
                STX        CFS.PREV
                LDA        ST.FLINK,X
                TAX
                STX        CFS.NEXT
*
* IF CFS.NEXT=0 OR ST.LIM(CFS.NEXT)<CFS.PTR
*    THEN DONE
*    ELSE GOTO L0
*
                BEQ        NFRPG.EXIT
                LDA        ST.LIML,X
                CMP        CFS.PTR
                LDA        ST.LIMH,X
                SBC        CFS.PTR+1
                BCS        NXTFRPG
*
NFRPG.EXIT      RTS                             ; NORMAL EXIT
                PAGE
                REP        60
*
* CHANGE.SEG(IN.SEGNUM,CHG.MODE; INOUT.PAGECT) SYSTEM CALL
*
                REP        60
*
CHG.SEG         EQU        *
*
* MOVE CALLER'S PAGE COUNT TO INTERNAL BUFFER
*
                LDY        #0
                LDA        (CHG.PGCT),Y
                STA        CHG.PGCTX
                INY
                LDA        (CHG.PGCT),Y
                STA        CHG.PGCTX+1
*
* IF SEG# OUT OF RANGE OR ST.FLINK(SEG#)=FREE THEN ERR
*
                LDX        CHG.NUM
                BEQ        CHGS.ERR
                CPX        #ST.CNT
                BCS        CHGS.ERR
                LDA        ST.FLINK,X
                BPL        CHGS005
*
CHGS.ERR        LDA        #BADSEGNUM
                JSR        SYSERR               ; ERR EXIT
                REP        35
* CASE OF CHANGE MODE
                REP        35
CHGS005         LDY        CHG.MODE
                CPY        #1
                BCC        CHGS010
                BEQ        CHGS020
                CPY        #3
                BCC        CHGS030
                BEQ        CHGS040
*
                LDA        #BADCHGMODE
                JSR        SYSERR               ; ERR EXIT
                PAGE
                REP        35
```

```
* CHANGE MODE = 0(BASE UP)
                REP        35
* CHG.NEW:=ST.BASE(SEG#)+PGCT
*
CHGS010         CLC
                LDA        ST.BASEL,X
                ADC        CHG.PGCTX
                STA        CHG.NEW
                LDA        ST.BASEH,X
                ADC        CHG.PGCTX+1
                STA        CHG.NEW+1
*
                BCS        CHGS014             ; OVERFLOW, PEG IT
*
* IF CHG.NEW <= ST.LIM(SEG#) THEN EXIT
*
                LDA        ST.LIML,X
                CMP        CHG.NEW
                LDA        ST.LIMH,X
                SBC        CHG.NEW+1
                BCS        CHGS016
*
* OTHERWISE, CHG.NEW:=ST.LIM(SEG#)
*
CHGS014         LDA        ST.LIML,X
                STA        CHG.NEW
                LDA        ST.LIMH,X
                STA        CHG.NEW+1
*
CHGS016         JMP        CHGS.EXIT
                REP        35
* CHANGE MODE = 1(BASE DOWN)
                REP        35
* CHG.NEW:=ST.BASE(SEG#)-PGCT
*
CHGS020         SEC
                LDA        ST.BASEL,X
                SBC        CHG.PGCTX
                STA        CHG.NEW
                LDA        ST.BASEH,X
                SBC        CHG.PGCTX+1
                STA        CHG.NEW+1
                BCS        CHGS050
                BCC        CHGS052             ; OVERFLOW, PEG IT
                REP        35
* CHANGE MODE = 2(LIMIT UP)
                REP        35
* CHG.NEW:=ST.LIM(SEG#)+PGCT
*
CHGS030         CLC
```

```
                LDA        ST.LIML,X
                ADC        CHG.PGCTX
                STA        CHG.NEW
                LDA        ST.LIMH,X
                ADC        CHG.PGCTX+1
                STA        CHG.NEW+1
                BCC        CHGS050
                BCS        CHGS052              ; OVERFLOW, PEG IT
                REP        35
* CHANGE MODE = 3(LIMIT DOWN)
                REP        35
* CHG.NEW:=ST.LIM(SEG#)-PGCT
*
CHGS040         SEC
                LDA        ST.LIML,X
                SBC        CHG.PGCTX
                STA        CHG.NEW
                LDA        ST.LIMH,X
                SBC        CHG.PGCTX+1
                STA        CHG.NEW+1
                BCC        CHGS044              ; OVERFLOW, PEG IT
*
* IF CHG.NEW >= ST.BASE(SEG#) THEN EXIT
*
                LDA        CHG.NEW
                CMP        ST.BASEL,X
                LDA        CHG.NEW+1
                SBC        ST.BASEH,X
                BCS        CHGS046
*
* OTHERWISE CHG.NEW:=ST.BASE(SEG#)
*
CHGS044         LDA        ST.BASEL,X
                STA        CHG.NEW
                LDA        ST.BASEH,X
                STA        CHG.NEW+1
*
CHGS046         JMP        CHGS.EXIT
*
* DETERMINE NEW BANK/PAGE'S REGION,
* IF NEW BANK/PAGE IS INVALID THEN
* SET TO BASE OR LIMIT (CASE CHANGE MODE)
*
CHGS050         LDX        CHG.NEW
                LDY        CHG.NEW+1
                JSR        REGION
                BCS        CHGS052
                BNE        CHGS052
                BEQ        CHGS100
CHGS052         LDA        CHG.MODE
```

```
                CMP        #1
                BNE        CHGS054
                LDX        #>VRT.BASE
                LDY        #<VRT.BASE
                JMP        CHGS056
CHGS054         LDX        >VRT.LIM
                LDY        >VRT.LIM+1
CHGS056         STX        CHG.NEW
                STY        CHG.NEW+1
                PAGE
*
* COMPUTE BANK/PAGE OF ADJACENT SEGMENT, IF ANY
*   CASE CHANGE MODE
*
CHGS100         LDX        CHG.NUM
                LDA        CHG.MODE
                CMP        #1
                BNE        CHGS200
*   "1" IF ST.FLINK(SEG#)=0 THEN EXIT
                LDA        ST.FLINK,X
                BEQ        CHGS.EXIT
*       X,Y:=ST.LIM(ST.FLINK(SEG#))+1
                TAY
                LDA        ST.LIML,Y
                TAX
                LDA        ST.LIMH,Y
                TAY
                INX
                BNE        CHGS110
                INY
*       IF CHG.NEW < X,Y THEN CHG.NEW:=X,Y
CHGS110         CPY        CHG.NEW+1
                BCC        CHGS.EXIT
                BEQ        CHGS120
                BCS        CHGS300
CHGS120         CPX        CHG.NEW
                BCC        CHGS.EXIT
                BCS        CHGS300
*   "2" IF ST.BLINK(SEG#)=0 THEN EXIT
CHGS200         LDA        ST.BLINK,X
                BEQ        CHGS.EXIT
*       X,Y:= ST.BASE(ST.BLINK(SEG#))-1
                TAY
                LDA        ST.BASEL,Y
                TAX
                LDA        ST.BASEH,Y
                TAY
                TXA
                BNE        CHGS210
                DEY
```

```
CHGS210         DEX
*       IF CHG.NEW > X,Y THEN CHG.NEW:=X,Y
                CPY        CHG.NEW+1
                BCC        CHGS300
                BEQ        CHGS220
                BCS        CHGS.EXIT
CHGS220         CPX        CHG.NEW
                BCS        CHGS.EXIT
*
CHGS300         STX        CHG.NEW
                STY        CHG.NEW+1
                PAGE
                REP        35
*
* COMPUTE DELTA PAGE COUNT AND RETURN IT TO CALLER
* (CASE OF CHG.MODE)
*
                REP        35
CHGS.EXIT       LDX        CHG.NUM
                LDY        #0
                LDA        CHG.MODE
                CMP        #1
                BCC        CHGS500
                BEQ        CHGS510
                CMP        #3
                BCC        CHGS520
                BEQ        CHGS530
*
* "0" -- PAGECOUNT:=NEW-BASE
*
CHGS500         SEC
                LDA        CHG.NEW
                SBC        ST.BASEL,X
                STA        (CHG.PGCT),Y
                LDA        CHG.NEW+1
                SBC        ST.BASEH,X
                JMP        CHGS600
*
* "1" -- PAGECOUNT:=BASE-NEW
*
CHGS510         SEC
                LDA        ST.BASEL,X
                SBC        CHG.NEW
                STA        (CHG.PGCT),Y
                LDA        ST.BASEH,X
                SBC        CHG.NEW+1
                JMP        CHGS600
*
* "2" -- PAGECOUNT:=NEW-LIM
*
```

```
CHGS520         SEC
                LDA        CHG.NEW
                SBC        ST.LIML,X
                STA        (CHG.PGCT),Y
                LDA        CHG.NEW+1
                SBC        ST.LIMH,X
                JMP        CHGS600
*
* "3" -- PAGECOUNT:=LIM-NEW
*
CHGS530         SEC
                LDA        ST.LIML,X
                SBC        CHG.NEW
                STA        (CHG.PGCT),Y
                LDA        ST.LIMH,X
                SBC        CHG.NEW+1
*
CHGS600         INY
                STA        (CHG.PGCT),Y
*
* IF NEW PAGE COUNT < REQUESTED PAGECOUNT THEN ERR
*
                TAX
                DEY
                LDA        (CHG.PGCT),Y
                CMP        CHG.PGCTX
                TXA
                SBC        CHG.PGCTX+1
                BCS        CHGS610
                LDA        #SEGRQDN
                JSR        SYSERR               ; ERR EXIT
*
* OTHERWISE, ENTER CHG.NEW IN SEGMENT TABLE AND EXIT
*
CHGS610         LDX        CHG.NUM
                LDA        CHG.MODE
                CMP        #2
                LDA        CHG.NEW
                LDY        CHG.NEW+1
                BCS        CHGS620
*
                STA        ST.BASEL,X
                TYA
                STA        ST.BASEH,X
                CLC
                RTS                             ; NORMAL EXIT
*
*
CHGS620         STA        ST.LIML,X
                TYA
                STA        ST.LIMH,X
                CLC
                RTS                                 ; NORMAL EXIT

                CHN        MEMMGR.C.SRC

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.B.SRC
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.MMGR.C.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.C.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        60
*
* GET.SEG.INFO(IN.SEGNUM; OUT.BASE.BKPG,LIMIT.BKPG,PGCT,SEGID)
*
                REP        60
*
GET.SEG.INFO    EQU        *
*
* IF SEG# OUT OF BOUNDS OR ST.FLINK(SEG#)=ST.FREE THEN ERR
*
                LDX        GSI.NUM
                BEQ        GSI.ERR              ; ERR - INVALID SEGNUM
                CPX        #ST.CNT
                BCS        GSI.ERR              ; ERR - INVALID SEGNUM
                LDA        ST.FLINK,X
                BMI        GSI.ERR              ; ERR - INVALID SEGNUM
*
* RETURN BASE.BKPG TO CALLER
*
                LDY        ST.BASEH,X
                LDA        ST.BASEL,X
                TAX
                JSR        CNVRT.XBP
                TYA
                LDY        #1
                STA        (GSI.BASE),Y
                DEY
                TXA
                STA        (GSI.BASE),Y
*
* RETURN LIMIT.BKPG TO CALLER
*
                LDX        GSI.NUM
                LDY        ST.LIMH,X
                LDA        ST.LIML,X
                TAX
                JSR        CNVRT.XBP
                TYA
                LDY        #1
                STA        (GSI.LIM),Y
                DEY
                TXA
                STA        (GSI.LIM),Y
*
* RETURN SEGID TO CALLER
*
                LDX        GSI.NUM
                LDA        ST.ID,X
                STA        (GSI.ID),Y
*
* COMPUTE PAGE COUNT
*
                SEC
                LDA        ST.LIML,X
                SBC        ST.BASEL,X
                TAY
                LDA        ST.LIMH,X
                SBC        ST.BASEH,X
                TAX
                INY
                BNE        GSI010
                INX
*
* RETURN PAGE COUNT TO CALLER
*
GSI010          TYA
                LDY        #0
                STA        (GSI.PGCT),Y
                INY
                TXA
                STA        (GSI.PGCT),Y
*
                CLC
                RTS                             ; NORMAL EXIT
*
GSI.ERR         LDA        #BADSEGNUM
                JSR        SYSERR               ; ERR EXIT
                PAGE
                REP        60
*
* GET.SEG.NUM(IN.BANKPAGE; OUT.SEGNUM) SYSTEM CALL
*
*
                REP        60
*
GET.SEG.NUM     EQU        *
*
* CONVERT BANKPAGE TO INTERNAL FORMAT
*
```

```
                LDX        GSN.BKPG
                LDY        GSN.BKPG+1
                JSR        CNVRT.IBP
                BCS        GSN.ERR                     ; ERR - INVALID BANK PAGE
                STX        GSN.BKPG
                STY        GSN.BKPG+1
*
* QUIT IF NO ENTRIES IN SEG TABLE
*
                LDA        ST.ENTRY
                BEQ        GSN.ERR1                    ; ERR - SEG NOT FOUND
*
* L1: IF BANKPAGE>ST.LIM(SEG#) THEN ERR
*
GSN010          TAX
                LDA        ST.LIML,X
                CMP        GSN.BKPG
                LDA        ST.LIMH,X
                SBC        GSN.BKPG+1
                BCC        GSN.ERR1                    ; ERR - SEG NOT FOUND
*
* IF BANKPAGE>=ST.BASE(SEG#) THEN FOUND!
*
                LDA        GSN.BKPG
                CMP        ST.BASEL,X
                LDA        GSN.BKPG+1
                SBC        ST.BASEH,X
                BCS        GSN020
*
* SEG#:=ST.FLINK(SEG#); GOTO L1
*
                LDA        ST.FLINK,X
                BEQ        GSN.ERR1                    ; ERR - SEG NOT FOUND
                JMP        GSN010
*
* RETURN SEG# TO CALLER
*
GSN020          LDY        #0
                TXA
                STA        (GSN.NUM),Y
                CLC
                RTS                                    ; NORMAL EXIT
*
GSN.ERR         RTS                                    ; ERROR EXIT
*
GSN.ERR1        LDA        #SEGNOTFND
                JSR        SYSERR                      ; ERROR EXIT
                PAGE
                REP        60
*
```

```
* RELEASE.SEG(IN.SEGNUM) SYSTEM CALL
*
                REP        60
*
RELEASE.SEG     EQU        *
*
* IF ST.FLINK(SEG#)=ST.FREE THEN ERR
*
                LDX        RLS.NUM
                BEQ        RLS.ALL              ; RELEASE.SEG(SEG#=0)
                CPX        #ST.CNT
                BCS        RLS.ERR              ; ERR - SEG# TOO LARGE
                LDA        ST.FLINK,X
                BMI        RLS.ERR              ; ERR - INVALID SEGNUM
                BPL        REL.SEG              ; RELEASE.SEG(SEG#>0)
                REP        35
*
* RELEASE ALL
*
                REP        35
RLS.ALL         LDX        ST.ENTRY
                BEQ        RLS0.EXIT
                STX        RLS.NUM
*
RLS0.LOOP       LDA        ST.ID,X
                CMP        #$10                 ; CARRY SET/CLEARED HERE
*
                LDA        ST.FLINK,X
                PHA
                BCC        RLS006               ; IF ID=SYS SEG THEN SKIP
                JSR        REL.SEG              ; RELEASE ONE SEGMENT
RLS006          PLA
                BEQ        RLS0.EXIT
                STA        RLS.NUM
                TAX
                BNE        RLS0.LOOP            ; ALWAYS TAKEN
*
RLS0.EXIT       CLC
                RTS                             ; NORMAL EXIT ; ALL NON SYSTEM SEGMENTS RELEASED.
                REP        35
*
* REL SEG
*
                REP        35
* Y:=ST.FLINK(SEG#)
* X:=ST.BLINK(SEG#)
*
REL.SEG         TAY
                LDA        ST.BLINK,X
                TAX
```

```
*
* IF X<>0 THEN ST.FLINK(X):=Y
*         ELSE ST.ENTRY:=Y
*
                BEQ        RLS010
                TYA
                STA        ST.FLINK,X
                JMP        RLS020
RLS010          STY        ST.ENTRY
*
* IF Y<>0 THEN ST.BLINK(Y):=X
*
                TYA
RLS020          BEQ        RLS030
                TXA
                STA        ST.BLINK,Y
*
* ST.FLINK(SEG#):=ST.FREE
* ST.FREE:=SEG# AND #$80
*
RLS030          LDA        ST.FREE
                LDX        RLS.NUM
                STA        ST.FLINK,X
                TXA
                ORA        #$80
                STA        ST.FREE
*
                CLC
                RTS                             ; NORMAL EXIT
*
RLS.ERR         LDA        #BADSEGNUM
                JSR        SYSERR               ; ERR EXIT
                PAGE
                REP        60
*
* CONVERT INTERNAL BANK PAGE
*
* INPUT:  EXTERNAL BANK (X)
*             "     PAGE (Y)
* OUTPUT: INTERNAL BKPG LOW  (X)
*             "     BKPG HIGH (Y)
*         REGION (A) 0=>VIRT BANK
*                    1=>PHY BANK (0-$2000)
*                    2=>   "     ($A000-$FFFF)
* ERROR:  CARRY SET ("INVALID BANK PAGE")
*
                REP        60
*
CNVRT.IBP       EQU        *
*
```

```
* CONVERT FROM EXTERNAL TO INTERNAL FORMAT
*
* CASE OF BANK:  ADD PAGE BIAS
*
                TYA
                CPX        #$F
                BEQ        CNVI010
                BCS        CNVI020
*
                CMP        #$20                 ; BANK < "F"
                BCC        CNVI.ERR1
                CMP        #$A0
                BCS        CNVI.ERR1
                SEC
                SBC        #$20
                JMP        CNVI030
*
CNVI010         CMP        #$20                 ; BANK = "F"
                BCS        CNVI.ERR1
                CLC
                ADC        #$80
                JMP        CNVI030
*
CNVI020         CPX        #$10                 ; BANK = "10"
                BNE        CNVI.ERR1
                CMP        #$A0
                BCC        CNVI.ERR1
                SEC
                SBC        #$80
*
CNVI030         TAY                             ; SHIFT BANK RIGHT ONE BIT
                TXA                             ; INTO HIGH BIT OF PAGE BYTE.
                LSR        A
                TAX
                TYA
                BCC        CNVI040
                ORA        #$80
*
* EXCHANGE X & Y
*
CNVI040         PHA
                TXA
                TAY
                PLA
                TAX
*
* COMPUTE REGION (VIRT=0,PHY1=1,PHY2=2)
*
                JSR        REGION               ; REGION RETURNED IN A REG.
                BCS        CNVI.ERR1            ; ERR - INVALID BANK PAGE
```

```
*
                RTS                                ; NORMAL EXIT
*
CNVI.ERR1       LDA        #BADBKPG
                JSR        SYSERR
                PAGE
                REP        60
*
* CONVERT EXTERNAL BANK PAGE
*
* INPUT:  INTERNAL BKPG LOW  (X)
*             "         HIGH (Y)
* OUTPUT: EXTERNAL BANK (X)
*             "    PAGE (Y)
* ERROR:  NO ERROR CHECKING DONE.  ASSUMES THAT INTERNAL #S
* ARE VALID.
*
                REP        60
*
CNVRT.XBP       EQU        *
*
* CONVERT FROM INTERNAL TO EXTERNAL FORMAT
*
                TXA
                ASL        A
                TXA
                AND        #$7F
                TAX
                TYA
                ROL        A
                TAY
*
* CASE OF BANK: ADD PAGE BIAS
*
                TXA
                CPY        #$F
                BEQ        CNVX020              ; BANK = "F"
                BCS        CNVX010
*
                CLC                             ; BANK < "F"
                ADC        #$20
                JMP        CNVX020
*
CNVX010         CLC                             ; BANK = "10"
                ADC        #$80
*
* EXCHANGE X & Y
*
CNVX020         PHA
                TYA
                TAX
                PLA
                TAY
                RTS                             ; NORMAL EXIT
                PAGE
                REP        60
*
* REGION
*
* INPUT:  INTERNAL BKPG LOW  (X)
*             "         HIGH (Y)
* OUTPUT: REGION (A)
*         INTERNAL BKPG LOW  (X) UNCHANGED
*             "         HIGH (Y)    "
* ERROR:  CARRY SET ("INVALID BANK/PAGE")
*
                REP        60
*
REGION          EQU        *
                STX        RGN.BKPG
                STY        RGN.BKPG+1
*
* IF BANKPAGE>PHY2LIM THEN ERR
*
                LDA        #>PHY2LIM
                CMP        RGN.BKPG
                LDA        #<PHY2LIM
                SBC        RGN.BKPG+1
                BCC        RGN.ERR              ; ERR - INVALID BANK PAGE
*
* IF BANKPAGE>=PHY2BASE THEN REGION:=2
*
                LDA        RGN.BKPG
                CMP        #>PHY2BASE
                LDA        RGN.BKPG+1
                SBC        #<PHY2BASE
                BCC        RGN010
                LDA        #2
                BNE        RGN040
*
* IF BANKPAGE>PHY1LIMIT THEN ERR
*
RGN010          LDA        #>PHY1LIM
                CMP        RGN.BKPG
                LDA        #<PHY1LIM
                SBC        RGN.BKPG+1
                BCC        RGN.ERR              ; ERR - INVALID BANK PAGE
*
* IF BANKPAGE>=PHY1BASE THEN REGION:=1
*
```

```
                LDA        RGN.BKPG
                CMP        #>PHY1BASE
                LDA        RGN.BKPG+1
                SBC        #<PHY1BASE
                BCC        RGN020
                LDA        #1
                BNE        RGN040
*
* IF BANKPAGE>VIRTUAL LIMIT THEN ERR
*
RGN020          LDA        >VRT.LIM
                CMP        RGN.BKPG
                LDA        >VRT.LIM+1
                SBC        RGN.BKPG+1
                BCC        RGN.ERR
                LDA        #0
*
RGN040          CLC                          ; "N" FLAG ALWAYS REFLECTS REGION VAL IN A REG!
                RTS                          ; NORMAL EXIT
*
RGN.ERR         SEC                          ; INVALID BANK PAGE
                RTS
                PAGE
                REP        60
*
* GET FREE
*
* INPUT:  PREVIOUS SEG # (A)
* OUTPUT: NEW SEG #      (A)
* ERROR:  CARRY SET ("SEG TBL FULL")
*
                REP        60
*
GET.FREE        EQU        *
*
* SAVE PREV SEG # IN X
* NOTE: PREV SEG # CARRIED IN X
*       NEW SEG # CARRIED IN Y
*
                TAX
*
* IF NO FREE ENTRIES THEN ERR
*
                LDA        ST.FREE
                CMP        #$80
                BEQ        GTFR.ERR
*
* TURN OFF FREE FLAG (BIT7) AND DELINK FROM FREE LIST
*
                AND        #$7F
                TAY
                LDA        ST.FLINK,Y
                STA        ST.FREE
*
* IF PREV SEG # IS NULL THEN LINK NEW ENTRY TO START
* OF SEGMENT LIST
*
                CPX        #0
                BNE        GTFR010
                LDA        ST.ENTRY
                STA        ST.FLINK,Y
                LDA        #0
                STA        ST.BLINK,Y
                STY        ST.ENTRY
                JMP        GTFR020
*
* OTHERWISE LINK NEW ENTRY TO PREV SEG #
*
GTFR010         LDA        ST.FLINK,X
                STA        ST.FLINK,Y
                TXA
                STA        ST.BLINK,Y
                TYA
                STA        ST.FLINK,X
*
* IF ST.FLINK(NEW)<>NULL THEN
*    ST.BLINK(ST.FLINK(NEW)):=NEWSEG #
GTFR020         LDA        ST.FLINK,Y
                BEQ        GTFR030
                LDA        ST.FLINK,Y
                TAX
                TYA
                STA        ST.BLINK,X
*
* RETURN WITH NEW SEG #
*
GTFR030         TYA
                CLC
                RTS                                ; NORMAL EXIT
*
GTFR.ERR        LDA        #SEGTBLFULL
                JSR        SYSERR
*
                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENMEMMG
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR MEMMGR"
                FIN

```

```
*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.C.SRC
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.SCMGR.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SCMGR.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1 SYSTEM CALL MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGSCMGR
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* SYSTEM CALL MANAGER (VERSION = 1.10  )
*                     (DATE    = 8/04/81)
*
* THE SYSTEM CALL MANAGER:
* (1) RETRIEVE THE SYSCALL #,
* (2) DETERMINE THE LOCATION OF THE SYSTEM CALL PARMS AND
*     MOVE THEM TO THE SOS ZPAGE,
* (3) TRANSFER CONTROL TO THE APPROPRIATE INTERFACE MANAGER,
*     (FILE,DEVICE,UTILITY,MEMORY)
*
                REP        60
*
                ENTRY      SCMGR
*
                EXTRN      FMGR
                EXTRN      DMGR
                EXTRN      UMGR
                EXTRN      MMGR
                EXTRN      DBUGBRK
*
                EXTRN      SYSERR
                EXTRN      SERR
                EXTRN      BADSCNUM
                EXTRN      BADCZPAGE
                EXTRN      BADXBYTE
                EXTRN      BADSCPCNT
                EXTRN      BADSCBNDS
*
                EXTRN      SZPAGE
                EXTRN      SXPAGE
                EXTRN      CZPAGE
                EXTRN      CXPAGE
                EXTRN      CSPAGE
                PAGE
                REP        60
*
* SYSTEM CALL PARAMETER DEFINITION TABLES
*
* EACH ENTRY IS FOUR BYTES LONG.  THE FIRST BYTE CONTAINS THE
* NUMBER OF PARMS IN THE CALL.  THE REMAINING SIX NIBBLES, EACH
* DEFINE A PARAMETER IN THE CALL.  THE FIRST BIT OF THE
* NIBBLE DEFINES WHETHER THE PARM IS INPUT (0) OR OUTPUT (1).
* THE NEXT BIT DEFINES WHETHER THE PARM IS BY VALUE (0)
* OR BY REFERENCE (1).  THE FINAL TWO BITS SPECIFY THE
* PARM LENGTH IN BYTES (E.G. 0=LENGTH OF 1, 3=LENGTH OF 4 BYTES)
*
                REP        60
*
*   FILE SYSTEM CALL DEFINITIONS
*
FSC.CNT         EQU        $13
FSC.TBL         EQU        *
                DFB        $3,$5D,$00,$00       ; SCNUM=$C0 - CREATE
                DFB        $1,$50,$00,$00       ;   "  =$C1 - DESTROY
                DFB        $2,$55,$00,$00       ;   "  =$C2 - RENAME
                DFB        $3,$5D,$00,$00       ;   "  =$C3 - SET.FILE.INFO
                DFB        $3,$5D,$00,$00       ;   "  =$C4 - GET.FILE.INFO
                DFB        $4,$55,$99,$00       ;   "  =$C5 - VOLUME
                DFB        $1,$50,$00,$00       ;   "  =$C6 - SET.PREFIX
                DFB        $2,$50,$00,$00       ;   "  =$C7 - GET.PREFIX
                DFB        $4,$58,$D0,$00       ;   "  =$C8 - OPEN
                DFB        $3,$00,$00,$00       ;   "  =$C9 - NEW.LINE
                DFB        $4,$05,$19,$00       ;   "  =$CA - READ
                DFB        $3,$05,$10,$00       ;   "  =$CB - WRITE
                DFB        $1,$00,$00,$00       ;   "  =$CC - CLOSE
                DFB        $1,$00,$00,$00       ;   "  =$CD - FLUSH
                DFB        $3,$00,$30,$00       ;   "  =$CE - SET.MARK
                DFB        $2,$0B,$00,$00       ;   "  =$CF - GET.MARK
                DFB        $3,$00,$30,$00       ;   "  =$D0 - SET.EOF
                DFB        $2,$0B,$00,$00       ;   "  =$D1 - GET.EOF
                DFB        $1,$00,$00,$00       ;   "  =$D2 - SET.LEVEL
                DFB        $1,$80,$00,$00       ;   "  =$D3 - GET.LEVEL
                PAGE
*
*   DEVICE SYSTEM CALL DEFINITIONS
*
DSC.CNT         EQU        5
DSC.TBL         EQU        *
                DFB        $5,$05,$11,$90       ; SCNUM=$80 - D.READ
                DFB        $4,$05,$11,$00       ;   "  =$81 - D.WRITE
                DFB        $3,$00,$50,$00       ;   "  =$82 - D.STATUS
                DFB        $3,$00,$50,$00       ;   "  =$83 - D.CONTROL
                DFB        $2,$58,$00,$00       ;   "  =$84 - GET.DEV.NUM
                DFB        $4,$05,$D0,$00       ;   "  =$85 - D.INFO
*
*   UTILITY SYSTEM CALL DEFINITIONS
*
USC.CNT         EQU        5
USC.TBL         EQU        *
                DFB        $1,$00,$00,$00       ; SCNUM=$60 - SET.FENCE
                DFB        $1,$80,$00,$00       ;   "  =$61 - GET.FENCE
                DFB        $1,$50,$00,$00       ;   "  =$62 - SET.TIME
                DFB        $1,$50,$00,$00       ;   "  =$63 - GET.TIME
                DFB        $2,$0B,$00,$00       ;   "  =$64 - JOYSTICK
                DFB        $0,$00,$00,$00       ;   "  =$65 - COLD.START
*
*   MEMORY SYSTEM CALL DEFINITIONS
*
MSC.CNT         EQU        5
MSC.TBL         EQU        *
                DFB        $4,$11,$08,$00       ; SCNUM=$40 - REQUEST.SEG
                DFB        $6,$00,$99,$98       ;   "  =$41 - FIND.SEG
                DFB        $3,$00,$90,$00       ;   "  =$42 - CHANGE.SEG
                DFB        $5,$09,$99,$80       ;   "  =$43 - GET.SEG.INFO
                DFB        $2,$18,$00,$00       ;   "  =$44 - GET.SEG.NUM
                DFB        $1,$00,$00,$00       ;   "  =$45 - RELEASE.SEG
*
*   DEBUG SYSTEM CALL DEFINITION
*
DBUG            EQU        $FE
                PAGE
                REP        60
*
* DATA DECLARATIONS
*
                REP        60
Z.REG           EQU        $FFD0
SP.SAVE         EQU        $01FF
Z.SAVE          EQU        $01FD
B.SAVE          EQU        $01FC
*
ADR.LOW         EQU        $2000                ; LOW    ADDRESS   (BOUNDS CHECKING)
ADR.HIGH        EQU        $B800                ; HIGH   ADDRESS
ADR.MID         EQU        $A000                ; MIDDLE ADDRESS
*
* SCMGR'S VARIABLES
*
SCM.VARS        EQU        $E0
SCNUM           EQU        SCM.VARS+0           ; SYSTEM CALL NUMBER
```

```
SCRNUM          EQU        SCM.VARS+0           ; SYSTEM CALL REQUEST NUMBER
SCPTR           EQU        SCM.VARS+1           ;&2  SYSTEM CALL POINTER
MOVE.VARS       EQU        SCPTR+2              ; !! (LOOKOUT) !!
*
*
F.TPARMX        EQU        $A0                  ; FILE SYS CALL PARM START LOC
D.TPARMX        EQU        $C0                  ; DEVICE SYS CALL PARM START LOC
U.TPARMX        EQU        $C0                  ; UTILITY SYS CALL PARM START LOC
M.TPARMX        EQU        $60                  ; MEMORY SYS CALL PARM START LOC
*
* MOVE.PARM'S VARIABLES
*
TPARMX          EQU        MOVE.VARS+0          ; TARGET ADR OF SYS CALL PARMS
DFN.PTR         EQU        MOVE.VARS+1          ;&2
DFN.PTRX        EQU        MOVE.VARS+3
SCPTRX          EQU        MOVE.VARS+4
RGHT.NIB        EQU        MOVE.VARS+5
SCT.DFN         EQU        MOVE.VARS+6
SCT.DCNT        EQU        MOVE.VARS+7
PARM.CNT        EQU        MOVE.VARS+8
                PAGE
                REP        60
*
* SYSTEM CALL MANAGER
*
                REP        60
*
SCMGR           EQU        *
                LDA        #<SZPAGE             ; SET Z REG TO SOS ZPAGE
                STA        Z.REG
*
* SET SYSTEM X BYTES TO ABSOLUTE ADDRESS MODE.
*
                LDA        #0
                STA        SXPAGE+SCPTR+1
                STA        SERR                 ; AND INIT SYSTEM ERR CODE
*
* CALLER'S Z REG MUST BE $1A !!
* (B REG NOT CHECKED)
*
                LDA        Z.SAVE
                CMP        #<CZPAGE
                BEQ        SCM005
                LDA        #>BADCZPAGE
                JSR        SYSERR               ; EXIT TO DISPATCHER
*
* RETRIEVE CALLER'S PC ON HIS STACK
*
SCM005          LDX        SP.SAVE
                LDA        CSPAGE+6,X
                STA        SCPTR+1
                LDA        CSPAGE+5,X
                STA        SCPTR
                BNE        SCM010                 ; AND POINT IT TO SYS CALL NUM
                DEC        SCPTR+1
SCM010          DEC        SCPTR
*
* ADVANCE CALLER'S PC ON HIS STACK.
*
                CLC
                LDA        CSPAGE+5,X
                ADC        #2
                STA        CSPAGE+5,X
                BCC        SCM020
                INC        CSPAGE+6,X
*
* RETRIEVE SYSTEM CALL NUMBER
*
SCM020          LDY        #0
                LDA        (SCPTR),Y
                CMP        #DBUG
                BNE        SCM025
                JSR        DBUGBRK                ; DEBUG SYSTEM CALL
SCM025          STA        SCNUM
*
* RETRIEVE SYSTEM CALL PARAMETER ADDRESS
*
                INY
                LDX        #>SCPTR
                JSR        POINTER
                BCC        SCM030
                RTS                               ; ERROR EXIT
*
* CASE INTERFACE CODE OF SYSTEM CALL NUMBER
*  (INTERFACE CODE STRIPPED, LEAVING REQUEST CODE)
*
SCM030          LDA        #$20
                BIT        SCNUM
                BPL        SCM050
                LDA        SCNUM
                AND        #$3F
                STA        SCRNUM
                BVC        SCM040
*
                LDA        #F.TPARMX              ; "11XXXXXX" - JMP TO FILE MANAGER.
                STA        TPARMX
                LDX        #>FSC.TBL
                LDY        #<FSC.TBL
                LDA        #FSC.CNT
                JSR        MOVE.PARMS
                BCS        SCM.ERR1             ; ERR EXIT
                JMP        FMGR
*
SCM040          LDA        #D.TPARMX            ; "10XXXXXX" - JMP TO DEVICE MANAGER.
                STA        TPARMX
                LDX        #>DSC.TBL
                LDY        #<DSC.TBL
                LDA        #DSC.CNT
                JSR        MOVE.PARMS
                BCS        SCM.ERR1             ; ERR EXIT
                JMP        DMGR
*
SCM050          BVC        SCM.ERR
                PHP
                LDA        SCNUM
                AND        #$1F
                STA        SCRNUM
                PLP
                BEQ        SCM060
*
                LDA        #U.TPARMX            ; "011XXXXX" - JMP TO UTILITY MANAGER.
                STA        TPARMX
                LDX        #>USC.TBL
                LDY        #<USC.TBL
                LDA        #USC.CNT
                JSR        MOVE.PARMS
                BCS        SCM.ERR1             ; ERR EXIT
                JMP        UMGR
*
SCM060          LDA        #M.TPARMX            ; "010XXXXX" - JMP TO MEMORY MANAGER.
                STA        TPARMX
                LDX        #>MSC.TBL
                LDY        #<MSC.TBL
                LDA        #MSC.CNT
                JSR        MOVE.PARMS
                BCS        SCM.ERR1             ; ERR EXIT
                JMP        MMGR
*
SCM.ERR         LDA        #>BADSCNUM           ; ERROR, INVALID SYSTEM CALL NUMBER.
SCM.ERR1        JSR        SYSERR               ;   EXIT TO DISPATCHER ON ERROR
                PAGE
                REP        60
*
* MOVE.PARMS
*
* MOVES THE CALLER'S PARAMETERS TO THE OPERATING SYSTEM'S
* ZERO PAGE, ACCORDING TO THE SPECIFICATIONS CONTAINED
* IN THE SPECIFIED SYS CALL DFN TABLE.
*
* INPUT: (A) = MAX # ENTRIES IN PARM DFN TABLE
*        (X) = PARM DFN TBL ADR (LO)
*        (Y) =         "        (HI)
*      SCPTR = ADR OF CALLER'S SYS CALL PARMS
* ERROR: CARRY SET (SYSERR)
*
                REP        60
*
MOVE.PARMS      EQU        *
                STX        DFN.PTR                ; SAVE ADR OF DEFINITION TABLE
                STY        DFN.PTR+1
*
* IF REQ NUM > MAX REQ NUM (A REG)
*
                CMP        SCRNUM
                BCS        MOVE010
*
*   THEN ERR(BAD SYS CALL NUM)
*
                LDA        #>BADSCNUM
                BCC        SYSERR1                ;BRANCH ALWAYS TAKEN
*
* CALCULATE DEFINITION TABLE INDEX
*  AND INIT SYS CALL PARM INDEX
*
MOVE010         LDA        SCRNUM
                ASL        A
                ASL        A
                STA        DFN.PTRX
                LDA        #0
                STA        SXPAGE+DFN.PTR+1       ; AND X BYTE
                STA        SCPTRX
*
* IF SCPTR(SCPTRX)<>DFN.PTR(DFN.PTRX) THEN ERR
*
                TAY
                LDA        (SCPTR),Y
                LDY        DFN.PTRX
                CMP        (DFN.PTR),Y
                BEQ        INITLOOPCT
*
                LDA        #>BADSCPCNT            ; ERR, CALLER'S PARM COUNT INVALID
SYSERR1         JSR        SYSERR                 ; EXIT
*
* INIT LOOP CTR(PARM.CNT) TO # OF PARMS IN SYS CALL
*
INITLOOPCT      STA        PARM.CNT
*
* ADVANCE PTRS
*
*
```

```
                INC        SCPTRX
                INC        DFN.PTRX
*
* MOVE REQ CODE TO SYS ZPAGE PARM LIST
*  AND ADVANCE SYS ZPAGE PTR (X=TPARMX)
*
                LDA        SCRNUM
                LDX        TPARMX
                STA        0,X
                INX
*
* INIT NIBBLE FLAG TO "RIGHT" NIBBLE
*  ZERO STATE="LEFT" NIBBLE
*
                LDA        #$FF
                STA        RGHT.NIB
                REP        60
*
*  BEGIN PARAMETER PROCESSING LOOP
*
PARMLOOP        LDA        RGHT.NIB
                EOR        #$FF                 ; COMPLEMENT NIBBLE FLAG
                STA        RGHT.NIB
*
* IF "LEFT" NIBBLE
*
                BNE        ELSE.RNIB
*
* THEN FETCH SYS CALL PARM DFN
*  AND # OF BYTES IN PARM WITHIN IT
*
                LDY        DFN.PTRX
                LDA        (DFN.PTR),Y
                STA        SCT.DFN
                AND        #$30
                LSR        A
                LSR        A
                LSR        A
                LSR        A
                STA        SCT.DCNT
                BPL        VALUE                ;BRANCH ALWAYS
*
* ELSE FETCH SYS CALL PARM DFN
*  AND # OF BYTES IN PARM WITHIN IT
*  FROM "RIGHT" NIBBLE OF DFN BYTE
*
ELSE.RNIB       LDA        SCT.DFN
                TAY
                AND        #$03
                STA        SCT.DCNT
                TYA
                ASL        A
                ASL        A
                ASL        A
                ASL        A
                STA        SCT.DFN
                INC        DFN.PTRX             ; ADVANCE SYS CALL DFN PTR
                REP        60
*
*  PARAMETER PASSED BY VALUE
*
                REP        60
VALUE           BIT        SCT.DFN
                BVS        REFERENCE
                BMI        VAL.OUT
*
*  INPUT BY VALUE
*
                LDY        SCPTRX               ; MOVE BYTES TO ZPAGE
VAL.IN          LDA        (SCPTR),Y
                STA        0,X
                INY
                INX
                DEC        SCT.DCNT
                BPL        VAL.IN
                STY        SCPTRX
                JMP        ENDLOOP1
*
*  OUTPUT BY VALUE
*
VAL.OUT         CLC                             ; BUILD PTR TO PARM ON ZPAGE
                LDA        SCPTR
                ADC        SCPTRX
                STA        0,X
                INX
                LDA        SCPTR+1
                ADC        #0
                STA        0,X
*
                CLC                             ; ADVANCE INDEX TO NEXT PARM
                LDA        SCPTRX
                ADC        SCT.DCNT
                STA        SCPTRX
*
                LDA        SXPAGE+SCPTR+1       ; INCLUDE X BYTE
                STA        SXPAGE,X
                JMP        ENDLOOP2
                REP        60
*
*  PARAMETER PASSED BY REFERENCE
```

```
*
                REP        60
REFERENCE       BPL        REF1
*
* "LIST" PTR FOUND, CHK IF "LENGTH" PARM = 0
*
                LDY        SCPTRX
                INY
                INY
                LDA        (SCPTR),Y
                BEQ        ENDLOOP0             ; "LENGTH" PARM=0, SKIP "LIST" PARM
*
REF1            LDY        SCPTRX               ; MOVE PTR TO ZPAGE
                JSR        POINTER
                BCS        PARM.ERR             ; ERROR EXIT
*
* ADVANCE SYSTEM ZPAGE POINTER (X), CALLER'S PARM PTR.
* DECREMENT PARM CTR AND CHECK IF LAST PARM PROCESSED.
*
ENDLOOP0        INX
                INC        SCPTRX
ENDLOOP2        INX
                INC        SCPTRX
ENDLOOP1        DEC        PARM.CNT
                BEQ        PARM.EXIT
                BMI        PARM.EXIT            ;SPECIAL FOR 'COLD START'
                JMP        PARMLOOP
*
*  END OF PARAMETER PROCESSING LOOP
*
                REP        60
*
PARM.EXIT       CLC                             ; NO ERRORS
PARM.ERR        RTS                             ; RETURN TO SYS CALL MANAGER
                PAGE
                REP        60
*
* POINTER
*
* INPUT:   SRC ADR   (SCPTR),Y & (SCPTR),Y+1
*          DEST ADR  (X)
*
* OUTPUT:  SCPTR     UNCHANGED
*          X REG         "
*          A,Y REGS  FLATTENED
*
* ERROR:   CARRY SET (SYSERR)
*
* POINTER.  RETRIEVES THE CALLER'S POINTER PARAMETER IN
* (SCPTR),Y, PERFORMS ADDRESS COMPENSATION, IF NECESSARY
* AND PLACES THE RESULTING POINTER AT X, X+1 AND SXPAGE+1,X.
*
                REP        60
*
POINTER         EQU        *
                LDA        (SCPTR),Y
                PHA
                INY
                LDA        (SCPTR),Y
                BEQ        INDIRECT
*
                STA        1,X                     ; DIRECT POINTER
                PLA
                STA        0,X
                LDY        #0
                BEQ        PTR010
*
INDIRECT        PLA                                ; INDIRECT POINTER
                TAY
                LDA        CZPAGE,Y
                STA        0,X
                LDA        CZPAGE+1,Y
                STA        1,X
                LDA        CXPAGE+1,Y
                TAY
*
PTR010          LDA        1,X
*
* CHECK BOUNDS OF CALLER'S POINTER PARAMETER
*
                CPY        #$8F
                BCC        PTR.X808E
                BEQ        PTR.X8F
                BCS        PTR.ERR1                ; ERROR, INVALID X BYTE
PTR.X8F         CMP        #<ADR.LOW
                BCC        PTR.ERR
                CMP        #<ADR.HIGH
                BCS        PTR.ERR
                BCC        PTR.EXIT
*
* X BYTE = 80..8E
*
PTR.X808E       CPY        #$80
                BCC        PTR.X0
                CMP        #0
                BEQ        PTR.ERR
                CMP        #$FF
                BNE        PATCH
                INY                                ; $8N:FFXX --> $8N+1:7FXX
                LDA        #$7F
                BNE        PTR.EXIT
*
* X BYTE = 0
*
PTR.X0          CPY        #0
                BNE        PTR.ERR1
                CMP        #<ADR.LOW
                BCC        PTR.ERR
                CMP        #<ADR.HIGH
                BCS        PTR.ERR
                CMP        #<ADR.MID
                BCS        PTR.EXIT
*
                PHA
                LDA        B.SAVE
                AND        #$0F
                BNE        PTR030
                PLA                             ; $B=0:2000..9FFF --> $8F:2000.9FFF
                LDY        #$8F
                BNE        PTR.EXIT
*
PTR030          ORA        #$80                 ; $B<>0:2000..9FFF --> $8B:0000..7FFF
                TAY
                PLA
                SEC
                SBC        #$20
                BNE        PATCH
                DEY                             ; $8B:00XX --> $8B-1:80XX
                LDA        #$80
*
PATCH           CPY        #$80                 ; KLUDGE FOR BFM:  $8N:01XX --> $8N-1:81XX
                BCC        PTR.EXIT
                CMP        #1
                BNE        PTR.EXIT
                CPY        #$80
                BEQ        PTR.ERR              ; ERROR, $80:01XX NOT ALLOWED
                DEY
                LDA        #$81
*
PTR.EXIT        STA        1,X
                TYA
                STA        SXPAGE+1,X
                CLC
                RTS
*
*
PTR.ERR         LDA        #>BADSCBNDS
                JSR        SYSERR
PTR.ERR1        LDA        #>BADXBYTE
                JSR        SYSERR
```

```
*
                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENSCMGR
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR SCMGR"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SCMGR.SRC
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.2of5.TWO:SOS.SYSERR.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SYSERR.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1 SYSTEM ERROR ROUTINES"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGSERR
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* SYSTEM ERROR ROUTINES  (VERSION = 1.10   )
*                        (DATE    = 12/02/81)
*
* THIS MODULE CONTAINS THE SYSTEM ERROR AND SYSTEM FAILURE ROUTINES.
*
                REP        60
*
                ENTRY      SYSERR
                ENTRY      SYSDEATH
*
                EXTRN      SERR
                EXTRN      SDEATH.REGS
                EXTRN      SCRNMODE
                PAGE
                REP        60
*
* DATA DECLARATIONS
*
                REP        60
*
E.REG           EQU        $FFDF
Z.REG           EQU        $FFD0
B.REG           EQU        $FFEF
*
S.SAVE          EQU        $09                  ; REGISTER SAVE AREA
PCH.SAVE        EQU        $08
PCL.SAVE        EQU        $07
P.SAVE          EQU        $06
A.SAVE          EQU        $05
```

```
X.SAVE          EQU        $04
Y.SAVE          EQU        $03
E.SAVE          EQU        $02
Z.SAVE          EQU        $01
B.SAVE          EQU        $00
*
NMI.VECTOR      EQU        $FFFA
*
TXT.CLR         EQU        $C050
MIX.CLR         EQU        $C052
HIRES.CLR       EQU        $C056
*
PG2.CLR         EQU        $C054
*
MSGBASE         EQU        $7E4
MSGBASE2        EQU        $BE4
MSG             ASC        ' SYSTEM FAILURE = $'
MSGLEN          EQU        *-MSG
                PAGE
                REP        60
*
* SYSTEM ERROR ROUTINE
*
* THIS ROUTINE IS CALLED WHEN AN ERROR CONDITION HAS BEEN
* ENCOUNTERED.  THE ERROR NUMBER IS PASSED IN THE A REG
* AND THE CALL TO THIS ROUTINE MUST ALWAYS BE A JSR.
*
                REP        60
SYSERR          EQU        *
*
                STA        SERR
                PLA
                STA        SDEATH.REGS+PCL.SAVE
                PLA
                STA        SDEATH.REGS+PCH.SAVE
                SEC
                LDA        SERR
                BNE        SERR.EXIT
                CLC
SERR.EXIT       RTS                          ; RETURNS ONE LEVEL BEYOND CALLER
                PAGE
                REP        60
*
* SYSTEM DEATH ROUTINE
*
* CALLED TO IMMEDIATELY TERMINATE EXECUTION OF THE MACHINE
* BECAUSE A FATAL ERROR HAS BEEN DETECTED BY THE OPERATING
* SYSTEM.  THE ERROR CODE IS PASSED IN THE A REG.  THE
* CALL TO THIS ROUTINE MUST ALWAYS BE A JSR.
*
```

```
                REP        60
SYSDEATH        EQU        *
*
                STA        SDEATH.REGS+A.SAVE   ; SAVE REGISTERS
                STX        SDEATH.REGS+X.SAVE
                STY        SDEATH.REGS+Y.SAVE
                PHP
                PLA
                STA        SDEATH.REGS+P.SAVE
                TSX
                STX        SDEATH.REGS+S.SAVE
                LDA        E.REG
                STA        SDEATH.REGS+E.SAVE
                LDA        Z.REG
                STA        SDEATH.REGS+Z.SAVE
                LDA        B.REG
                STA        SDEATH.REGS+B.SAVE
                PLA
                STA        SDEATH.REGS+PCL.SAVE
                PLA
                STA        SDEATH.REGS+PCH.SAVE
*
                SEI                             ; TURN OFF INTERRUPTS
                CLD
*
                LDX        #0                   ; SAVE SYSTEM STACK PAGE IN PAGE $17
SD005           LDA        $100,X
                STA        $1700,X
                DEX
                BNE        SD005
*
                LDA        $C059                ; ENSURE SILENTYPE PORT SHUT DOWN
                LDA        $C0DD
                LDA        $C0DF
                LDA        $C05F
                LDA        $C05A
*
                LDA        $C040                ; SOUND BELL
*
                LDA        #$74                 ; ENSURE RESET LOCK OFF & RAM SWITCHED IN.
                STA        E.REG
*
                LDA        TXT.CLR              ; SWITCH TO 40 COL B&W DISPLAY MODE
                LDA        MIX.CLR
                LDA        HIRES.CLR
                LDA        PG2.CLR              ; & SELECT PAGE 1
*
                LDA        #$02
                BIT        SCRNMODE
                BVS        SD015                ; IF GRAPHICS MODE THEN KEEP 40 COL MODE
                BEQ     SD015                ; IF 40 COL MODE THEN KEEP
                LDA     MIX.CLR+1            ; ELSE SWITCH TO 80 COL DISPLAY MODE
*
                LDX     #MSGLEN+1            ; ENSURE BKGRND SET TO INVERSE SPACES
                LDA     #$20                 ; SPACE CHAR W/INVERSE
SD010           STA     MSGBASE2-1,X
                DEX
                BPL     SD010
*
SD015           LDX     #0                   ; MOVE MSG TO TEXT SCREEN
SD020           LDA     MSG,X
                STA     MSGBASE-1,X
                INX
                CPX     #MSGLEN
                BNE     SD020
*
                LDA     SDEATH.REGS+A.SAVE   ; DISPLAY ERROR CODE (2 HEX DIGITS)
                CLC
                LSR     A
                LSR     A
                LSR     A
                LSR     A
                JSR     PRINT                ; FIRST DIGIT
                INX
                LDA     SDEATH.REGS+A.SAVE
                AND     #$0F
                JSR     PRINT                ; SECOND DIGIT
*
                LDA     #>SD100
                STA     NMI.VECTOR
                LDA     #<SD100
                STA     NMI.VECTOR+1
*
*
                JMP     *                    ; HANG UNTIL REBOOT (CTRL/RESET)
                REP     60
SD100           RTI                          ; NMI VECTOR POINT HERE TO MASK THEM OUT
*
*
* PRINT SUBROUTINE
*
PRINT           EQU     *
                CMP     #$A
                BCS     PRNT100
                ADC     #$30                 ; "0"-"9"
                BCC     PRNT110              ; ALWAYS TAKEN
PRNT100         ADC     #$36                 ; "A"-"F"
PRNT110         STA     MSGBASE-1,X
                RTS
*
```

```
                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENSERR
                FAIL       2,"SOSORG           FILE IS INCORRECT FOR SYSERR"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SYSERR.SRC
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.3of5.THREE:SOS.ALLOC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: ALLOC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

*
DEALLOC         STX        BMCNT                ; SAVE HIGH ORDER ADDRESS OF BLOCK TO BE FREED.
                PHA                             ; SAVE IT
                LDX        VCBPTR               ; WHILE THE BITMAP
                LDA        VCB+VCBTBLK+1,X      ; DISK ADDRESS IS CHECKED
                CMP        BMCNT                ; TO SEE IF IT MAKES SENSE
                PLA                             ; RESTORE
                BCC        DEALERR1             ; BRANCH IF IMPOSSIBLE
                TAX
                AND        #$7                  ; GET THE BIT TO BE OR-ED IN.
                TAY
                LDA        WHICHBIT,Y           ; (SHIFTING TAKES 7 BYTES, BUT IS SLOWER)
                STA        NOFREE               ; SAVE BIT PATTERN
                TXA                             ; GET LOW BLOCK ADDRESS AGAIN.
                LSR        BMCNT
                ROR        A                    ; GET POINTER TO BYTE IN BITMAP THAT REPRESENTS
                LSR        BMCNT                ; THE BLOCK ADDRESS.
                ROR        A
                LSR        BMCNT
                ROR        A
                STA        BMPTR                ; SAVE POINTER.
                LSR        BMCNT                ; NOW TRANSFER BIT WHICH SPECIFIES WHICH PAGE OF BITMAP.
                ROL        HALF
                LDX        BMTAB                ; (THIS POINTS TO THE TABLE FOR THE BITMAP BUFFER USED).
                LDA        BMACMAP,X            ; WHAT IS THE CURRENT MAP
                CMP        BMCNT                ; IS IN CORE BIT MAP THE ONE WE WANT?
                BEQ        DEALL1               ; BRANCH IF IN-CORE IS CORRECT.
                JSR        BMAPUP               ; PUT CURRENT MAP AWAY.
                BCS        DEALERR              ; PASS BACK ANY ERROR.
                LDA        BMCNT                ; GET DESIRED MAP NUMBER.
                LDY        #VCBCMAP
                STA        (VCBPTR),Y           ; AND MAKE IT CURRENT.
                LDX        BMTAB
                LDA        BMADEV,X
                JSR        GTBMAP               ; READ IT INTO THE BUFFER,
                BCS        DEALERR
DEALL1          LDY        BMPTR                ; INDEX TO BYTE.
                LSR        HALF
                BCC        DEALL2               ; BRANCH IF ON PAGE ONE OF BITMAP.
                INC        BMADR+1
```

```
DEALL2          LDA        NOFREE               ; THE INDIVIDUAL BIT.
                ORA        (BMADR),Y
                STA        (BMADR),Y
                BCC        DEALL3               ; BRANCH IF ADDRESS IS PROPER
                DEC        BMADR+1
DEALL3          LDX        BMTAB                ; MARK BITMAP AS MODIFIED.
                LDA        #$80
                ORA        BMASTAT,X
                STA        BMASTAT,X
                CLC
DEALERR         RTS
DEALERR1        LDA        #BITMAPADR           ; BIT MAP BLOCK NUMBER IMPOSSIBLE
                SEC                             ; SAY BIT MAP DISK ADDRESS WRONG
                RTS                             ;   (PROBABLY DATA MASQUERADING AS INDEX BLOCK)
*
WHICHBIT        DFB        $80,$40,$20,$10
                DFB        8,4,2,1
*
*
                PAGE
*
ALCIDXS         LDA        #0                   ; ALLOCATION OF THE INDEXES ALWAYS FILLS IN
                STA        SAPTR                ; STARTING AT THE BEGINNING OF THE BLOCK.
                JSR        ALC1BLK              ; THIS GETS FIRST INDEX AND SETS UP A
                BCS        ERRALC1              ; POINTER TO THE FREE BLOCKS (TO AVOID
ALIDX1          LDY        SAPTR                ; SCANNING THE WHOLE BLOCK EVERY TIME).
                STA        (TINDX),Y            ; SAVE INDEX BLOCK ADDRESS (LOW)
                INC        TINDX+1
                LDA        SCRTCH+1             ; GET HIGH BYTE OF ADDRESS
                STA        (TINDX),Y            ; (AND SAVE IT)
                DEC        TINDX+1
                DEC        REQL                 ; HAS REQUEST BEEN SATIFIED?
                BEQ        ALDXEND              ; (CARRY IS CLEAR)
                INC        SAPTR                ; BUMP INDEX POINTER
                LDY        BMPTR                ; GET INDEX POINTER TO LAST ACCESSED BIT GROUP
                LDA        HALF                 ; WHICH HALF OF MAP? (BOTH BMPTR & HALF SET UP BY 'ALC1BLK')
                BNE        SECNDHAF
                JSR        GETBITS1             ; GET NEXT FREE BLOCK ADDRESS.
                BCC        ALIDX1               ; BRANCH IF NO ERROR
ERRALC1         RTS
*
SECNDHAF        JSR        GETBITS2             ; GET NEXT FREE BLOCK ADDRESS FROM SECOND HALF OF BIT MAP
                BCC        ALIDX1               ; BRANCH IF NO ERROR.
ALDXEND         RTS                             ; RETURN STATUS (CARRY SET INDICATES ERROR)
*
*
ALC1BLK         JSR        FNDBMAP              ; GET ADDRESS OF BIT MAP IN 'BMADR'
                BCS        ERRALC1              ; BRANCH IF ERROR ENCOUNTERED
SRCHFRE         LDY        #0                   ; START SEARCH AT BEGINNING OF BIT MAP BLOCK
                STY        HALF                 ; INDICATE WHICH HALF (PAGE) WE'RE SEARCHING.
```

```
GETBITS1        LDA        (BMADR),Y
                BNE        BITFOUND             ; FREE BLOCKS ARE INDICATED BY 'ON' BITS
                INY
                BNE        GETBITS1             ; CHECK ALL OF 'EM IN FIRST PAGE.
                INC        BMADR+1              ; BUMP HIGH ADDRESS OF CURRENT BITMAP
                INC        HALF                 ; INDICATE SEARCH HAS PROGRESSED TO PAGE 2
                INC        BASVAL               ; BASE VALUE= BASE ADDRESS/2048
GETBITS2        LDA        (BMADR),Y            ; SEARCH SECOND HALF FOR FREE BLOCK
                BNE        BITFOUND
                INY
                BNE        GETBITS2
                DEC        BMADR+1              ; RESET BIT MAP ADDRESS TO BEGINNING.
                INC        BASVAL               ; ADD 2048 OFFSET FOR NEXT PAGE
                JSR        NXTBMAP              ; GET NEXT BITMAP (IF IT EXISTS) AND UPDATE VCB.
                BCC        SRCHFRE              ; BRANCH IF NO ERROR ENCOUNTERED.
                RTS                             ; RETURN ERROR.
                PAGE
*
BITFOUND        STY        BMPTR                ; SAVE INDX POINTER TO VALID BIT GROUP
                LDA        BASVAL               ; SET UP FOR BLOCK ADDRESS CALCULATION
                STA        SCRTCH+1
                TYA                             ; GET ADDRESS OF BIT PATTERN
                ASL        A                    ; MULTIPLY THIS AND BASVAL BY 8
                ROL        SCRTCH+1
                ASL        A
                ROL        SCRTCH+1
                ASL        A
                ROL        SCRTCH+1
                TAX                             ; NOW X= LOW ADDRESS WITHIN 7 OF ACTUAL ADDRESS.
                LDA        (BMADR),Y            ; GET BIT PATTERN AGAIN
                SEC                             ; MARK RIGHT END OF BYTE.
ADCALC          ROL        A                    ; FIND LEFT MOST 'ON' BIT
                BCS        BOUNCE               ; BRANCH IF FOUND.
                INX                             ; ADJUST LOW ADDRESS
                BNE        ADCALC               ; BRANCH ALWAYS
BOUNCE          LSR        A                    ; RESTORE ALL BUT LEFT MOST BIT TO ORIGINAL POSITION
                BCC        BOUNCE               ; LOOP UNTIL MARK (SET ABOVE) MOVES INTO CARRY
                STA        (BMADR),Y            ; UPDATE BITMAP TO SHOW ALLOCATED BLOCK IN USE.
                STX        SCRTCH               ; SAVE LOW ADDRESS.
                LDX        BMTAB                ; UPDATE BIT MAP BUFFER STATUS
                LDA        #$80                 ; INDICATE MAP HAS BEEN MODIFIED
                ORA        BMASTAT,X            ; (X IS EITHER 0 OR 6 FOR
                STA        BMASTAT,X            ; BUFFER 'A' OR 'B' RESPECTIVELY.)
                LDY        #VCBTFRE             ; SUBTRACT 1 FROM TOTAL FREE
                LDA        (VCBPTR),Y           ; BLOCKS IN VCB TO ACCOUNT FOR NEWLY
                SBC        #1                   ; ALLOCATED BLOCK (CARRY IS SET FROM 'BOUNCE')
                STA        (VCBPTR),Y
                BCS        RET1BLK              ; BRANCH IF HI FREE COUNT DOESN'T NEED ADJUSTMENT.
                INY
                LDA        (VCBPTR),Y           ; ADJUST HIGH COUNT.
                SBC        #0                   ; (CARRY IS CLEAR, SO ACC=ACC-1)
                STA        (VCBPTR),Y
RET1BLK         CLC                             ; INDICATE NO ERROR ENCOUNTERED
                LDA        SCRTCH               ; GET ADDRESS LOW IN ACC.
                LDY        SCRTCH+1             ; AND HIGH ADDRESS IN Y
                RTS                             ; RETURN ADDRESS OF NEWLY ALLOCATED BLOCK.
*
                PAGE
*
GTTINDX         LDY        #VCBDEV              ; GET DEVICE NUMBER SO WE DON'T
                LDX        #0                   ; ANTICPATE USING BUFFER 'A'.
                LDA        (VCBPTR),Y           ; USE THE BUFFER USED BY IT!
                CMP        BMADEV               ; IS IT IN BUFFER 'A'?
                BEQ        FREEBE               ; IF SO, FREE 'B'!
                CMP        BMBDEV               ; IF NOT, IS IT IN 'B'?
                BEQ        FREEA                ; IF SO, FREE UP BUFFER 'A'
                JSR        FNDBMAP              ; OTHERWISE, FORCE ALLOCATION FOR ONE OF THE BUFFERS
                BCC        GTTINDX              ; NOW TRY AGAIN.
                RTS                             ; RETURN ERROR.
*
FREEBE          LDX        #BMTABSZ             ; DE-ALLOCATE BUFFER IF NECESSARY
FREEA           STX        NOFREE               ; SAVE WHICH BUFFER WE'RE LOOKIN AT.
                LDY        BMASTAT,X            ; DO WE NEED TO WRITE BUFFER TO FREE IT?
                BPL        USEBUF               ; NO, THEN USE IT.
                STX        ZPGTEMP              ; SAVE BM BUFFER ID FOR A BIT
                JSR        WRTBMAP              ; WRITE BM TO OWNING UNIT
                BCS        SOMERR1              ; RETURN ANY ERROR (W/O RELEASING BM)
                LDX        ZPGTEMP              ; FETCH THE BM BUFFER ID
                LDA        #0
                STA        BMASTAT,X            ; AND MARK BM BUFFER AS FREE
USEBUF          LDX        NOFREE               ; GET INDEX TO BUFFER INFO
                LDA        #0                   ; MARK STATUS OF BUFFER AS FREE.
                STA        BMADEV,X             ; (DEVICE 0 IS NOT ANY DEVICE)
                STA        TINDX
                STA        BMADR
                LDA        BMAMADR,X            ; GET MEMORY ADDRESS OF FREE BUFFER.
                STA        TINDX+1
                TXA                             ; SET UP PROPER HI ADDRESS OF BIT MAP TOO...
                EOR        #BMTABSZ             ; SELECT ALTERNATE BIT MAP TABLE.
                STA        BMTAB                ; (TO INDICATE WHICH IS BITMAP)
                TAX
                LDA        BMAMADR,X            ; GET HIGH ADDRESS OF BIT MAP.
                STA        BMADR+1
                LDA        BMBUFBNK             ; AND BANK PAIR NUMBER.
                STA        SSTIDXH
                STA        SISBMADR
                CLC                             ; INDICATE NO ERRORS
SOMERR1         RTS
*
                PAGE
```

```
NXTBMAP         LDY        #VCBTBLK+1           ; BEFORE BUMPING TO NEXT MAP,
                LDA        (VCBPTR),Y           ; CHECK TO BE SURE THERE IS
                LSR        A                    ; INDEED A NEXT MAP!
                LSR        A
                LSR        A
                LSR        A
                LDY        #VCBCMAP
                CMP        (VCBPTR),Y           ; ARE THERE MORE MAPS?
                BEQ        NOMORBIT             ; BRANCH IF NO MORE TO LOOK AT.
                LDA        (VCBPTR),Y           ; ADD 1 TO CURRENT MAP
                CLC
                ADC        #1
                STA        (VCBPTR),Y
                LDY        #VCBDEV
                LDA        (VCBPTR),Y
                TAX                             ; GO WRITE OUT LAST MAP IF NECESSARY
                JSR        UPBMAP
                JMP        FNDBMAP              ; READ NEXT BIT MAP INTO BUFFER
*
GETA.BUF        LDX        #0
                BEQ        FRESHMAP
*
GETB.BUF        LDX        #BMTABSZ
                BNE        FRESHMAP             ; BRANCH ALWAYS
*
*
FNDBMAP         LDY        #VCBDEV              ; GET DEVICE NUMBER
                LDA        (VCBPTR),Y
                LDX        #0                   ; START WITH MAP 'A'
FNDMAP1         CMP        BMADEV,X
                BNE        TRYMAP2
FRESHMAP        STX        BMTAB                ; SAVE POINTER TO BIT MAP INFO TABLE
                LDY        BMASTAT,X            ; IS THIS ONE ALREADY MODIFIED?
                BMI        BMFOUND              ; YES, RETURN POINTER IN 'BMADR'
                JSR        GTBMAP               ; OTHERWISE READ IN FRESH BIT MAP
                BCC        BMFOUND              ; BRANCH IF SUCCESSFUL.
                RTS                             ; OTHERWISE, RETURN ERROR.
*
TRYMAP2         DEX                             ; WAS LAST FAILURE MAP 'A'
                BPL        FRBMBUF              ; NO, MUST FREE UP ONE OF THE BUFFERS
                LDX        #BMTABSZ             ; TRY BIT MAP BUFFER 'B'.
                JMP        FNDMAP1
                PAGE
*
BMFOUND         LDX        BMTAB                ; WHICH TABLE?
                LDY        #VCBCMAP
                LDA        (VCBPTR),Y
                ASL        A
                STA        BASVAL
                LDA        BMAMADR,X            ; GET HIGH ADDRESS
                STA        BMADR+1
                LDA        BMBUFBNK             ; GET BANK NUMBER OF BUFFER BIT MAP BUFFERS
                STA        SISBMADR
                LDA        #0                   ; BUFFERS ALWAYS FALL ON A PAGE BOUNDARY
                STA        BMADR
                CLC                             ; INDICATE ALL IS VALID AND GOOD!
                RTS
*
NOMORBIT        LDA        #OVRERR              ; INDICATE REQUEST CAN'T BE FILLED.
                SEC                             ; INDICATE ERROR
                RTS
*
FRBMBUF         SEC
                LDX        BMTAB                ; FIND OUT WHICH WAS LAST USED.
                BEQ        CHKBMB               ; IF 'A' WAS USED CHECK 'B' FIRST
                CLC                             ; INDICATE 'A' IS CHECKED FIRST
                BIT        BMASTAT              ; IS BUFFER 'A' FREE (UNMODIFIED)?
                BPL        GETA.BUF             ; YES, USE IT.
CHKBMB          BIT        BMBSTAT              ; IS BUFFER 'B' FREE?
                BCC        FREBUF1              ; BRANCH IF BOTH ARE USED
                BPL        GETB.BUF             ; YES...
                BIT        BMASTAT              ; (CHECK 'A')
                BPL        GETA.BUF
FREBUF1         LDX        #0
                BCC        FREBUFA              ; BRANCH IF BUFFER 'A' HAS LEAST PRIORITY.
                LDX        #BMTABSZ
FREBUFA         STX        ZPGTEMP              ; SAVE BM BUFF ID FOR A BIT
                JSR        WRTBMAP              ; XREG PASSES BM BUFF ID
                BCS        NOGO                 ; ERROR ENCOUNTERED ON WRITING
                LDX        ZPGTEMP              ; FETCH BM BUFF ID
                LDA        #0
                STA        BMASTAT,X            ; AND MARK BM BUFFER AS FREE
                BCC        FNDBMAP              ; LOOK AGAIN FOR FRRE BIT MAP BUFFER SPACE
NOGO            RTS                             ; RETURN ERROR ON WRITING BM
*
UPBMAP          CPX        BMADEV               ; UPDATE BIT MAP OF DEVICE X
                BNE        UPBM1
                CLC                             ; FREE BUFFER 'A' IF NEEDED.
                BIT        BMASTAT
                BMI        FREBUF1              ; (CARRY CLEAR FOR BUFFER 'A')
                RTS
                PAGE
*
UPBM1           CPX        BMBDEV
                BNE        NOUPDAT              ; DON'T UPDATE IF NOT NECESSARY.
                BIT        BMBSTAT
                BMI        FREBUF1              ; (CARRY IS SET)
NOUPDAT         CLC
                RTS                             ; RETURN 'NO ERROR'
*
```

```
CLEARBMS        EQU        *                    ; MAKE SURE ALL BIT MAPS ASSOCIATED
* WITH A DEVICE ARE MARKED INVALID
* IF A NEW VOLUME IS LOGGED IN ON IT.
* INPUT ARG: A REG = DEVNUM
* X REG PRESERVED
                LDY        #0
                CMP        BMADEV
                BNE        CLRBM1               ; BRANCH IF BIT MAP A NOT OWNED
                BIT        BMASTAT
                BMI        CLRBM2               ; BRANCH IF BITMAP A BUSY
                STY        BMADEV               ; ELSE, CLEAR IT
CLRBM2          RTS                             ; NEED ONLY CLEAR ONE
CLRBM1          CMP        BMBDEV               ; BIT MAP B?
                BNE        CLRBM2               ; BRANCH IF BIT MAP B NOT OWNED BY DEVNUM
                BIT        BMBSTAT
                BMI        CLRBM2               ; BRANCH IF BITMAP B BUSY
                STY        BMBDEV               ; ELSE CLEAR IT
                RTS                             ; AND RETURN TO CALLER (NO ERRORS)
*
GTBMAP          STA        BMADEV,X             ; SAVE ACC AS CURRENT DEVICE FOR BUFFER
                LDA        BMAMADR,X            ; GET HIGH ORDER ADDRESS OF BUFFER
                STA        BMADR+1              ; SELECTED BY X
                LDA        BMBUFBNK             ; AND GET BANK PAIR NUMBER
                STA        SISBMADR             ; OF BOTH BIT MAP BUFFERS 'A' AND 'B'
                LDY        #VCBCMAP             ; GET LOWEST MAP NUMBER WITH FREE BLOCKS IN IT.
                LDA        (VCBPTR),Y
                STA        BMACMAP,X            ; ASSOCIATE THE OFFSET WITH THE BITMAP CONTROL BLOCK
                CLC
                LDY        #VCBDMAP             ; ADD THIS NUMBER TO THE BASE
                ADC        (VCBPTR),Y           ; ADDRESS OF FIRST BIT MAP
                STA        BMADADR,X            ; SAVE LOW ADDRESS OF BIT MAP TO BE USED.
                INY                             ; NOW GET HIGH DISK ADDRESS OF MAP
                LDA        (VCBPTR),Y           ; ADD TO THIS THE STATE OF THE CARRY
                ADC        #0
                STA        BMADADR+1,X          ; SAVE HIGH DISK ADDRESS TOO.
; DROP INTO 'RDBMAP'
*
                PAGE
*
                LDA        #RDCMD               ; (X CONTAINS AN INDEX TO DETERMINE WHICH BUFFER)
DOBMAP          STA        DHPCMD               ; SAVE DEVICE COMMAND
                LDA        DEVNUM               ; FIX THE 'BIT MAP TRASH BUG'
                PHA                             ; BY NOT MUNGING DEVNUM
                LDA        BMADEV,X             ; GET DEVICE NUMBER.
                STA        DEVNUM
                LDA        BMADADR,X            ; AND MAP'S DISK ADDRESS
                STA        BLOKNML
                LDA        BMADADR+1,X
                STA        BLOKNMH
                LDA        BMAMADR,X            ; LASTLY GET THE ADDRESS OF THE BUFFER
                LDX        BMBUFBNK             ; AND BANK NUMBER.
                JSR        DOBITMAP             ; (NOTE: LOW ADDRESS IS FIXED TO ZERO AS THIS IS A BUFFER)
                PLA                             ; RESTORE
                STA        DEVNUM               ; THE DEVNUM WE CAME IN WITH!
                RTS
*
WRTBMAP         LDA        #WRTCMD              ; WRITE BIT MAP POINTED TO BY X
                JMP        DOBMAP
*
WRTGBUF         LDA        #WRTCMD              ; SET CALL FOR WRITE.
                BNE        SVGCMD               ; BRANCH ALWAYS.
RDGBUF          LDA        #RDCMD               ; SET CALL FOR READ.
SVGCMD          STA        DHPCMD               ; PASSED TO DEVICE HANDLER.
                LDA        BLOKNML              ; SAVE CURRENT
                STA        TTLINK               ;   GBUF BLOCK
                LDA        BLOKNMH              ; ADDRESS
                STA        TTLINK+1             ; FOR DIRECTORY EXTEND
                LDA        #GBUF/256            ; GET HIGH ADDRESS OF GENERAL BUFFER
                LDX        #0                   ; TO FORCE ACCESS TO NON BANK MEMORY.
DOBITMAP        EQU        *
DOIDX           STA        DBUFPH
                STX        SISBPH               ; SELECT BANK
                LDA        #0                   ; GENERAL PURPOSE BUFFERS ALWAYS
                STA        DBUFPL               ; START ON A PAGE BOUNDARY.
                JMP        FILEIO2              ; END VIA DEVICE DISPATCHER.
*
TTLINK          DS         2                    ; GBUF CURRENT ADDRESS
*
WRTINDX         LDA        #WRTCMD
                LDX        IDXADRL              ; GET BLOCK ADDRESS OF INDEX BLOCK
                LDY        IDXADRH
DOFRST          STA        DHPCMD               ; (ENTRY USED BY RD/WRTDFRST)
                STX        BLOKNML
                STY        BLOKNMH
                LDA        TINDX+1              ; HIGH RAM ADDRESS OF INDEX BLOCK
                LDX        SSTIDXH              ; AND BANK NUMBER.
                JMP        DOIDX                ; AND GO DO REQUESTED OPERATION.
*
WRTDFRST        LDA        #WRTCMD              ; WRITE FILE'S FIRST BLOCK (USED
                BNE        FADDR                ; BY CREATE, SO ADDRESS IN 'D.' STUFF).
RDFRST          LDA        #RDCMD
FADDR           LDX        DFIL+D.FRST          ; (BUFFER ADDRESS IS IN 'TINDX')
                LDY        DFIL+D.FRST+1
                JMP        DOFRST
*
*
                CHN        POSN/OPEN,4,2

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: ALLOC
```

```
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.3of5.THREE:SOS.CREATE.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: CREATE
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
CREATE          EQU        *
                INC        CFLAG                 ; SAY WE ARE IN CREATE (DIR EXTEND)
                JSR        LOOKFILE              ; CHECK FOR DUPLICATE / GET FREE ENTRY
                BCS        TSTFNF                ; ERROR CODE IN ACC MAY BE 'FILE NOT FOUND'
                LDA        #DUPERR               ; TELL EM A FILE OF THAT NAME ALREADY EXISTS
CRERR1          SEC                              ; INDICATE ERROR ENCOUNTERED
                RTS                              ; RETURN ERROR IN ACC.
*
TSTFNF          CMP        #FNFERR               ; 'FILE NOT FOUND' IS WHAT WE WANT
                BNE        CRERR1                ; PASS BACK OTHER ERROR.
                LDA        NOFREE                ; TEST FOR DIRECTORY SPACE
                BNE        CREAT1                ; BRANCH IF VALID FREE ENTRY WAS FOUND.
                LDA        #DIRFULL              ; RETURN DIRECTORY FULL ERROR
                SEC
                RTS
*
CREAT1          LDY        #$9                   ; SET UP DEFAULT PARAMETERS FOR CREATE
                LDA        #0                    ; IN THE SPACE DIRECTLY FOLLOWING THE
ZERCALL         STA        C.FILID,Y             ; CALL SPECIFCATION AND THEN
                DEY                              ; CHECK FOR ADDITIONAL PARAMETERS FROM
                BPL        ZERCALL               ; USER'S CALL SPEC VIA 'C.CLIST'
                LDA        #SEEDTYP              ; DEFAULT TYPE IS 'SEED' TREE INDEX
                STA        C.STOR
                LDY        C.XLEN                ; GET THE LENGTH OF THE CALL XTENSION LIST
                BEQ        CRENAM                ; IF ZERO THEN USE DEFAULTS
                DEY                              ; (SINCE THE POINTER IS AT BYTE 0)
                CPY        #$9                   ; MAKE SURE WE DON'T HAVE TOO MANY PARAMETERS
                BCC        MOVPARM               ; MOVE 'EM IF REASONABLE COUNT.
                LDA        #BADLSTCNT            ; INVALID LIST COUNT
                RTS                              ; RETURN ERROR.
*
MOVPARM         LDA        (C.XLIST),Y           ; MOVE IN THE USER SPECIFIED
                STA        C.FILID,Y             ; PARAMETERS. VALIDITY IS CHECKED
                DEY                              ; AT VARIOUS POINTS FURTHER ALONG IN
                BPL        MOVPARM               ; THIS PROCESS.
CRENAM          LDY        #0                    ; MOVE LOCAL FILE NAME TO ENTRY BUFFER.
                LDA        (PATHNML),Y           ; GET LENGTH OF LOCAL NAME
                TAY
CRENAM1         LDA        (PATHNML),Y
```

```
                STA        DFIL+D.STOR,Y
                DEY                             ; (MOVE ALL, INCLUDING LENGTH BYTE.)
                BPL        CRENAM1
                LDA        C.FILID              ; MOVE FILE AND AUX ID.
                STA        DFIL+D.FILID
                LDA        C.AUXID
                STA        DFIL+D.AUXID
                LDA        C.AUXID+1
                STA        DFIL+D.AUXID+1
                LDA        #READEN+WRITEN+RENAMEN+DSTROYEN
                STA        DFIL+D.ATTR
                LDA        D.HEAD               ; SAVE FILE'S HEADER ADDRESS TOO.
                STA        DFIL+D.DHDR
                LDA        D.HEAD+1
                STA        DFIL+D.DHDR+1
                JSR        TWRPROT1             ; CAN WE WRITE TO THIS DISKETTE?
                BCS        CRERR1
                LDA        C.STOR               ; NOW TEST STORAGE TYPE FOR TREE TYPE FILES
                CMP        #4                   ; NOTE: THIS IS HARD CODED SINCE ALL TREES ARE LESS THAN 4 ***********
                BCC        SEED                 ; BRANCH IF SOME TYPE OF TREE (SEED, SAPLING...)
                JMP        NOTREE               ; GO TEST FOR SOME OTHER TYPE (SUCH AS DIRECTORY).
                PAGE
*
SEED            LDX        #SEEDTYP             ; START OUT ASSUMING A SEED FILE
                LDA        C.EOFHH              ; TEST FOR OUT OF RANGE PREALLOCATION
                BEQ        SEED1                ; (HOPEFULLY BRANCH ALWAYS)
OVFLOW          LDA        #OVRERR              ; REPORT UNABLE TO SATISFY REQUEST.
                SEC                             ; INDICATE ERROR
                RTS
*
SEED1           LDA        C.EOFHL              ; CALCULATE THE NUMBER OF
                STA        DFIL+D.EOF+2         ; BLOCKS NEEDED FOR PRE-ALLOCATION
                LSR        A
                TAY                             ; Y HOLDS THE NUMBER OF INDEX BLOCKS NEEDED
                STA        DATBLKH
                LDA        C.EOFLH              ; (CARRY UNDISTURBED FROM LAST SHIFT)
                STA        DFIL+D.EOF+1
                ROR        A                    ; WE NOW HAVE THE LOW ORDER COUNT OF NEEDED DATA BLOCKS
                STA        DATBLKL
                LDA        C.EOFLL
                STA        DFIL+D.EOF           ; (CARRY IN TACT FROM LOW COUNT)
                BNE        INCDATA              ; BUMP THE COUNT ON DATA BLOCKS IF REQUEST
                BCC        TSTSAP               ; IS NOT A MULTIPLE OF 512.
INCDATA         INC        DATBLKL
                BNE        TSTSAP
                INY                             ; MUST INCREASE NUMBER OF INDEXES ALSO.
                INC        DATBLKH
TSTSAP          TYA                             ; IF NON ZERO, THEN IT'S AT LEAST A SAPLING.
                BNE        SAPLING
                LDA        DATBLKL              ; TO QUALIFY AS AN HONEST SEED,
                BNE     TSTSEED           ; THEN ONE OR LESS DATA BLOCKS REQUESTED
                INC     DATBLKL           ; (MUST BE AT LEAST ONE BLOCK ALLOCATED
                BNE     CREALC            ; TYPE IS SEED. BRANCH ALWAYS
TSTSEED         CMP     #1                ; IF GREATER THAN ONE, IT'S NOT A SEED.
                BEQ     CREALC            ; IT IS A SEED. CONTINUE CREATION
                INX                       ; THE TYPE IS SAPLING.
                INY                       ; ONE INDEX BLOCK IS NEEDED.
                BNE     CREALC            ; BRANCH ALWAYS
                PAGE
*
SAPLING         INX                       ; TYPE IS AT LEAST SAPLING.
                CMP     #1                ; NO MORE THAN ONE INDEX BLOCK FOR A SAPLING
                BNE     TREE
                LDA     DATBLKL           ; MUST BE SURE THIS IS REAL MAX SAPLING (128K FILE)
                BEQ     CREALC            ; BRANCH IF IT IS.
TREE            INY                       ; ACCOUNT FOR ADDITIONAL 2ND LEVEL INDEX
*
                INX                       ; TYPE IS TREE (2 LEVEL INDEX)
                INY                       ; ADD AN EXTRA INDEX BLOCK FOR TOP INDEX
CREALC          STY     INDXBLK           ; STORE INDEX BLOCK COUNT
                TXA                       ; PUT STORAGE TYPE IN DIRECTORY ENTRY
                ASL     A
                ASL     A
                ASL     A
                ASL     A
                ORA     DFIL+D.STOR
                STA     DFIL+D.STOR
                STX     LEVELS            ; SAVE NUMBER OF INDEX LEVELS FOR PREALLOCATION.
                TYA                       ; NOW FIGURE THE TOTAL NUMBER OF
                CLC                       ; BLOCKS NEEDED (DATA + INDEX BLOCKS)
                ADC     DATBLKL
                STA     DFIL+D.USAGE      ; (MIGHT AS WELL RECORD IT IN DIR
                STA     REQL              ; WHILE WE'RE AT IT.)
                LDA     DATBLKH
                ADC     #0                ; UPDATE HI BYTE TOO
                STA     DFIL+D.USAGE+1
                STA     REQH
                LDX     D.DEV             ; PASS ALONG THE DEVICE WE'RE TALKIN ABOUT.
                JSR     TSFRBLK           ; 'TEST FREE BLOCKS' FINDS OUT IF ENOUGH FREE SPACE EXISTS
                BCS     OVFLOW            ; BRANCH IF NOT ENOUGH SPACE.
                JSR     ALC1BLK           ; GO ALLOCATE FIRST BLOCK
                BCS     CRERR
                STA     DFIL+D.FRST       ; (RETURNS ACC=LOW Y=HIGH)
                STA     IDXADRL           ; SAVE AS ADDRESS FOR INCORE INDEX ALSO.
                STY     DFIL+D.FRST+1
                STY     IDXADRH
                JSR     ZERGBUF           ; GO CLEAN OUT GBUF
                JSR     GTTINDX           ; GET TEMPORARY SPACE FOR AN INDEX BLOCK
                JSR     ZTMPIDX           ; AND ZERO IT OUT.
                LDX     LEVELS
                DEX                                   ; TEST FOR NUMBER OF LEVELS NEEDED.
                BEQ        ENDCRE                     ; BRANCH IF SEED FILE.
                DEX                                   ; IS IT A SAPLING PRE-ALLOCATION.
                BEQ        SAPFILE
                LDY        INDXBLK                    ; LOAD NUMBER OF INDEX BLOCKS NEEDED
                DEY                                   ; REMOVE THE ONE JUST ALLOCATED.
                STY        REQL
                STY        INDXBLK
                JSR        ALCIDXS                    ; GO ALLOCATE INDEXES FOR LOWER INDEX BLOCKS.
                BCS        CRERR
                JSR        WRTDFRST                   ; GO WRITE TREE TOP INDEX BLOCK.
                BCS        CRERR                      ; BRANCH IF UNABLE TO DO THIS.
                LDA        #0                         ; INIT INDEX POINTER
                STA        TREPTR
                PAGE
FILLTREE        LDY        TREPTR
                LDA        (TINDX),Y                  ; GET ADDRESS OF LOWER BLOCK
                STA        IDXADRL
                INC        TINDX+1                    ; BUMP TO PAGE 2 TO GET HI ADDRESS.
                LDA        (TINDX),Y                  ; GET HIGH ADDRESS.
                STA        IDXADRH
                DEC        TINDX+1                    ; CLEAN UP AFTER SELF...
                DEC        INDXBLK                    ; IS THIS THE LAST BLOCK ALLOCATED?
                BEQ        LSTSAP                     ; YES, ALLOCATE PARTIAL FILLED INDEX BLOCK
                LDA        #0                         ; ALLOCATE ALL 256 INDEXES
                STA        REQL
                JSR        SAPINDX                    ; AND WRITE ZEROED DATA BLOCKS.
                BCS        CRERR                      ; STOP IF ERROR ENCOUNTERED.
                JSR        WRTINDX                    ; WRITE INDEX BLOCK
                BCS        CRERR                      ; HOPEFULLY NEVER TAKEN.
                INC        TREPTR
                JSR        RDFRST                     ; READ IN TOP INDEX AGAIN.
                BCC        FILLTREE                   ; BRANCH IF NO ERROR.
CRERR           SEC                                   ; JUST IN CASE IT WAS CLEAR.
                RTS                                   ; RETURN ERROR.
*
*
SAPFILE         EQU        *
LSTSAP          LDA        DATBLKL                    ; GET NUMBER OF DATA BLOCKS (LOW BYTE) REQUESTED.
                STA        REQL
                JSR        SAPINDX                    ; GO ALLOCATE DATA BLOCKS AND WRITE EM.
                BCS        CRERR
ENDCRE          JSR        WRTINDX                    ; GO WRITE INDEX BLOCK. (FOR SEED THIS IS DATA.)
                BCS        CRERR
                LDX        #3                         ; MOVE CREATION TIME FOR THIS ENTRY
TRETIME         LDA        DATELO,X
                STA        DFIL+D.CREDT,X
                DEX
                BPL        TRETIME
ENDCRE0         INC        H.FCNT                     ; ADD ONE TO TOTAL NUMBER OF FILES IN SPECIFIED DIRECTORY.
```

```
                BNE        ENDCRE1
                INC        H.FCNT+1
                LDX        #3                   ; ENSURE MOD
ENDCRX          LDA        DATELO,X             ; DATE/TIME
                STA        DFIL+D.MODDT,X       ; IS
                DEX                             ; INITIALIZED
                BPL        ENDCRX
ENDCRE1         LDX        D.DEV                ; UPDATE APPROPRIATE BIT MAP
                JSR        UPBMAP
                BCS        CRERR2               ; BRANCH ON BITMAP UPDATE ERR
                JSR        DREVISE              ; UPDATE DIRECTORY LAST
                RTS                             ; RETURN ERRORS OR OK RESULT
*
                PAGE
SAPINDX         JSR        ZTMPIDX              ; ZERO OUT ANY STUFF LEFT OVER.
                LDA        REQL                 ; PRESERVE REQUEST COUNT
                STA        TLINK
                JSR        ALCIDXS              ; GO ALLOCATE REQUESTED NUMBER OF BLOCKS.
                BCS        CRERR
                LDY        #0                   ; THEN WRITE ZEROS TO DATA BLOCKS.
                STY        SAPTR                ; USE AS POINTER TO INDEX BLOCK
                LDA        (TINDX),Y            ; GET DATA BLOCK ADDRESS (LOW BYTE).
                STA        BLOKNML
                INC        TINDX+1
                LDA        (TINDX),Y            ; GET HIGH ADRRESS OF PRE-ALLOCATED DATA BLOCK.
                STA        BLOKNMH
                DEC        TINDX+1              ; (RESET BUFFER ADDRESS)
                JSR        WRTGBUF              ; WRITE DATA BLOCK
                BCS        CRERR
                LDA        TLINK                ; GET NUMBER REQUESTED AGAIN
                STA        REQL
DATINIT         LDY        SAPTR                ; GET POINTER TO INDEX BLOCK AGAIN.
                INY                             ; ANTICIPATE DOIN' THE NEXT DATA BLOCK
                DEC        REQL                 ; DO WE INDEED HAVE ANOTHER BLOCK TO WRITE.
                BEQ        DATDONE              ; NO, ALL DONE (CARRY CLEAR).
                STY        SAPTR                ; USE AS POINTER TO INDEX BLOCK
                LDA        (TINDX),Y            ; GET DATA BLOCK ADDRESS (LOW BYTE).
                STA        BLOKNML
                INC        TINDX+1              ; BUMP HI ADDR OF INDEX BUFFER TO ACCESS HIGH ADDR.
                TAX                             ; WAS LOW ADDRESS A ZERO?
                BNE        DATIT1               ; IF NOT, NO NEED TO CHECK VALIDITH OF HI BYTE
                CMP        (TINDX),Y
                BNE        DATIT1               ; BOTH BYTES CAN'T BE ZERO.
                LDA        #ALCERR
                JSR        SYSDEATH
DATIT1          LDA        (TINDX),Y            ; GET HIGH ADRRESS OF PRE-ALLOCATED DATA BLOCK.
                STA        BLOKNMH
                DEC        TINDX+1              ; (RESET BUFFER ADDRESS)
                LDA        #GBUF/256
                STA        DBUFPH               ; RESET TO ADDR TO GBUF JUST TO BE SURE.
                JSR        REPEATIO             ; WRITE DATA BLOCK
                BCC        DATINIT
DATDONE         RTS                             ; RETURN STATUS (CARRY SET IF ERROR)
*
REPEATIO        EQU        *
                LDA        #RPTCMD
                STA        DHPCMD
                JMP        RPEATIO1
*
ZERGBUF         LDY        #0                   ; ZERO OUT THE GENERAL PURPOSE BUFFER
                TYA
ZGBUF           STA        GBUF,Y               ; WIPE OUT BOTH PAGES
                STA        GBUF+$100,Y          ; WITH SAME LOOP.
                INY
                BNE        ZGBUF
                RTS
*
*
ZTMPIDX         LDY        #0                   ; ZERO OUT TEMPORARY INDEX BLOCK
                TYA
ZINDX1          STA        (TINDX),Y            ; THIS HAS TO BE DONE A
                INY                             ; TIME SINCE IT'S INDIRECT.
                BNE        ZINDX1
                INC        TINDX+1
ZINDX2          STA        (TINDX),Y
                INY
                BNE        ZINDX2
                DEC        TINDX+1              ; RESTORE PROPER ADDRESS
CRERR2          RTS
                PAGE
NOTREE          CMP        #DIRTYP              ; IS A DIRECTORY TO BE CREATED?
                BEQ        ISDIR                ; YES, DO SO...
                JMP        NOTDIR               ; NO, TRY NEXT TYPE.
*
ISDIR           LDA        C.EOFHH              ; CAN'T CREATE A DIRECTORY LARGER THAN
                ORA        C.EOFHL              ; 127 BLOCKS (THAT'S HUGE!)
                BEQ        ISDIR1               ; BRANCH IF WITHIN LIMITS, OTHEWISE
DIROVR          LDA        #OVRERR              ; REQUESTED DIRECTORY SIZE CAN'T BE
                SEC                             ; CREATED. SET CARRY TO INDICATE ERROR.
                RTS
*
ISDIR1          LDA        C.EOFLH              ; CALCULATE HOW MANY BLOCKS WILL
                LSR        A                    ; BE NEEDED FOR THIS NEW DIRECTORY.
                TAY                             ; (SAVE INITIAL COUNT IN Y)
                LDA        C.EOFLL              ; IF REQUESTED EOF IS NOT AN EVEN BLOCK
                BNE        DADD1                ; SIZE, THEN ROUND UP.
                BCC        TSDIRSZ              ; BRANCH IF ROUNING UNNECESSARY.
DADD1           INY                             ; ADD ONE TO BLOCK COUNT.
TSDIRSZ         TYA                             ; TEST TO BE SURE SIZE IS GREATER THAN ZERO
                BEQ        DADD1                ; IF ZERO THEN SIZE=1
                STA     DFIL+D.USAGE        ; SAVE NUMBER OF BLOCKS TO BE USED.
                STA     REQL
                ASL     A                   ; NOW SAVE ADJUSTED END OF FILE
                STA     DFIL+D.EOF+1
                LDA     #0
                STA     DFIL+D.EOF
                STA     DFIL+D.EOF+2
                STA     REQH                ; REQUESTED NUMBER OF BLOCKS NEVER EXCEEDS 128.
                JSR     TSFRBLK             ; TEST TO BE SURE ENOUGH DISK SPACE IS FREE.
                BCS     DIROVR              ; BRANCH IF REQUEST TOO LARGE.
                JSR     ZERGBUF             ; CLEAR CRAP FROM GBUF.
                JSR     ALC1BLK             ; GET ADDRESS OF FIRST (HEADER) BLOCK.
                BCS     CRERR2
                STA     DFIL+D.FRST
                STA     TLINK
                STY     DFIL+D.FRST+1
                STY     TLINK+1             ; (TLINK IS FOR REVERSE LINKAGE.)
                LDA     SOSTMPL             ; STORE SOS STAMP IN NEW DIRECTORY
                STA     GBUF
                LDA     SOSTMPH
                STA     GBUF+1
                LDY     #4                  ; MOVE OTHER VARIOUS THINGS
                BNE     DRSTUF1             ; BRANCH ALWAYS
DRSTUF          LDA     D.ENTBLK,Y          ; MOVE OWNING ENTRY'S
                STA     GBUF+HRBLK+4,Y      ; BLOCK ADDRESSES AND NUMBER TO NEW HEADER.
DRSTUF1         LDA     SOSVER,Y            ; MOVE VERSION, COMPATABLITY,
                STA     GBUF+HVER+4,Y       ; ATTRIBUTES, AND ENTRY SIZE
                DEY
                BPL     DRSTUF
                LDA     H.ENTLN             ; OVER WRITE LAST BYTE MOVED IN ABOVE LOOP WITH
                STA     GBUF+HRELN+4        ; THE PARENT DIRECTORY ENTRY LENGTH.
                LDA     DFIL+D.STOR         ; SET HEADER TYPE AND NAME
                TAY
                ORA     #HEDTYP*16
                STA     GBUF+HNLEN+4
                TYA                         ; (AND WHILE WE'RE AT IT SET DIRECTORY TYPE)
                ORA     #DIRTYP*16
                STA     DFIL+D.STOR
*
MVHNAME         LDA     DFIL+D.STOR,Y
                STA     GBUF+HNLEN+4,Y      ; MOVE HEADER NAME
                DEY
                BNE     MVHNAME
                LDX     #3                  ; GET CURRENT DATE.
CRETIME         LDA     DATELO,X
                STA     GBUF+HCRDT+4,X      ; SAVE AS HEADER CREATION TIME
                STA     DFIL+D.CREDT,X      ; AND DATE OF FILE CREATE.
                DEX
                BPL     CRETIME
                LDA     #$76
                STA        GBUF+HPENAB+4        ; DUMMY PASSWORD
                DEC        REQL                 ; TEST FOR ONE BLOCK DIRECTORY
                BEQ        DIRCREND             ; IT IS, FINISH UP.
                JSR        DIRWRT               ; GO WRITE FIRST DIRECTORY BLOCK AND ALLOCATE NEXT
                BCS        DERROR               ; PASS BACK ERROR.
                JSR        ZERGBUF              ; CLEAN OUT GENERAL BUFFER AGAIN.
CRNXTDIR        LDA        TLINK                ; MOVE LAST BLOCK ADDRESS
                STA        GBUF                 ; AS BACKWARD LINK.
                LDA        TLINK+1
                STA        GBUF+1
                LDA        FLINK                ; MAKE FORWARD LINK INTO CURRENT ADDRESS
                STA        TLINK
                LDA        FLINK+1
                STA        TLINK+1
                DEC        REQL                 ; IS THIS THE LAST BLOCK?
                BEQ        DIRCREND
                JSR        DIRWRT               ; WRITE THIS BLOCK AND ALLOCATE NEXT.
                BCS        DERROR
                LDA        #0                   ; ZERO OUT FORWARD LINK
                STA        GBUF+2
                STA        GBUF+3
                BEQ        CRNXTDIR             ; BRANCH ALWAYS
*
DIRCREND        JSR        DIRWRT1              ; WRITE LAST BLOCK OF THIS DIRECTORY
                BCS        DERROR
                JMP        ENDCRE0              ; FINISH UP WRITING OWNER DIRECTORY STUFF.
*
DIRWRT          JSR        ALC1BLK              ; GET ADDRESS OF NEXT BLOCK.
                BCS        DERROR
                STA        GBUF+2
                STY        GBUF+3               ; SAVE LINK ADDRESS
                STA        FLINK
                STY        FLINK+1
DIRWRT1         LDA        TLINK                ; GET ADDRESS OF CURRENT BLOCK
                STA        BLOKNML
                LDA        TLINK+1
                STA        BLOKNMH
                JMP        WRTGBUF              ; GO WRITE IT OUT
                PAGE
*
ERRGBUF         EQU        *
DERROR          RTS
*
*
SOSTMPL         DFB        $0                   ; THE FOLLOWING TWO BYTES ARE THE 'SOS STAMP'
SOSTMPH         DFB        $0
*
SOSVER          DFB        0,0,0,$27,13
*
*
```

```
RNDTAB          EQU        *
ENTCALC         LDA        #GBUF/256            ; SET HIGH ADDRESS OF DIRECTORY ENTRY INDEX POINTER
                STA        DRBUFPH
                LDA        #4                   ; CALCULATE ADDRESS OF ENTRY BASED
                LDX        D.ENTNUM             ; ON THE ENTRY NUMBER
ECALC0          CLC
ECALC1          DEX                             ; ADDR=GBUF+((ENTNUM-1)*ENTLEN)
                BEQ        ECALC2
                ADC        H.ENTLN
                BCC        ECALC1
                INC        DRBUFPH              ; BUMP HI ADDRESS
                BCS        ECALC0               ; BRANCH ALWAYS.
*
ECALC2          STA        DRBUFPL              ; SAVE NEWLY CALCULATED LOW ADDRESS
                RTS
                PAGE
DERROR2         RTS
*
DREVISE         LDA        DATELO               ; IF NO CLOCK,
                BEQ        DREVISE1             ; THEN DON'T TOUCH MOD T/D
                LDX        #3                   ; MOVE LAST MODIFICATION DATE/TIME TO ENTRY BEING UPDATED.
MODTIME         LDA        DATELO,X
                STA        DFIL+D.MODDT,X
                DEX
                BPL        MODTIME
*
DREVISE1        LDA        DFIL+D.ATTR          ; MARK ENTRY AS BACKUPABLE
                ORA        BKBITFLG             ; BIT 5 = BACKUP NEEDED BIT
                STA        DFIL+D.ATTR
                LDA        D.DEV                ; GET DEVICE NUMBER OF DIRECTORY
                STA        DEVNUM               ; TO BE REVISED.
                LDA        D.ENTBLK             ; AND ADDRESS OF DIRECTORY BLOCK
                STA        BLOKNML              ; THAT CONTAINS THE ENTRY.
                LDA        D.ENTBLK+1
                STA        BLOKNMH
                JSR        RDGBUF               ; READ BLOCK INTO GENERAL PURPOSE BUFFER.
                BCS        ERRGBUF
                JSR        ENTCALC              ; FIX UP POINTER TO ENTRY LOCATION WITHIN GBUF.
                LDY        H.ENTLN              ; NOW MOVE 'D.' STUFF TO DIRECTORY.
                DEY
MVDENT          LDA        DFIL+D.STOR,Y
                STA        (DRBUFPL),Y
                DEY
                BPL        MVDENT
                LDA        D.HEAD               ; IS THE ENTRY BLOCK THE SAME AS THE
                CMP        BLOKNML              ; ENTRY'S HEADER BLOCK?
                BNE        SVENTDIR             ; NO, SAVE ENTRY BLOCK
                LDA        D.HEAD+1             ; MAYBE, TEST HIGH ADDRESSES
                CMP        BLOKNMH
                BEQ        UPHEAD               ; BRANCH IF THEY ARE THE SAME BLOCK.
```

```
SVENTDIR        JSR        WRTGBUF              ; WRITE UPDATED DIRECTORY BLOCK
                BCS        DERROR2              ; RETURN ANY ERROR.
                LDA        D.HEAD               ; GET ADDRESS OF HEADER BLOCK
                STA        BLOKNML
                LDA        D.HEAD+1
                STA        BLOKNMH
                JSR        RDGBUF               ; READ IN HEADER BLOCK FOR MODIFICATION
                BCS        DERROR2
UPHEAD          LDY        #1                   ; UPDATE CURRENT NUMBER OF FILES IN THIS DIRECTORY
UPHED1          LDA        H.FCNT,Y
                STA        GBUF+HCENT+4,Y       ; (CURRENT ENTRY COUNT)
                DEY
                BPL        UPHED1
                LDA        H.ATTR               ; ALSO UPDATE HEADER'S ATTRIBUTES.
                STA        GBUF+HATTR+4
                JSR        WRTGBUF              ; GO WRITE UPDATED HEADER
DERROR1         RTS                             ; IMPLICITLY RETURN ANY ERRORS
*
                PAGE
*
NOTDIR          LDA        #TYPERR              ; NOT TREE OR DIRECTORY- NOT A RECOGNIZED TYPE!
TSTERR          SEC
                RTS                             ; DO NOTHING.
*
*
TSTSOS          LDA        GBUF                 ; TEST SOS STAMP
                CMP        SOSTMPL
                BNE        TSTERR
                LDA        GBUF+1
                CMP        SOSTMPH
                BNE        TSTERR
                LDA        GBUF+4               ; TEST FOR HEADER
                AND        #$E0
                CMP        #HEDTYP*16
                BNE        TSTERR               ; BRANCH IF NOT SOS HEADER (NO ERROR NUMBER)
                CLC                             ; INDICATE NO ERROR
                RTS
*
                CHN        FNDFIL,4,1
NE              TSTERR
                LDA        GBUF+4               ; TEST FOR HEADER
                AND        #$E0
                CMP        #HEDTYP*16
                BNE        TSTERR               ; BRANCH IF NOT SOS HEADER (NO ERROR NUMBER)
                CLC                             ; INDICATE NO ERROR
                RTS
*
                CHN        FNDFIL,4,1
O               ERROR
                RTS
```

```
*
                CHN        FNDFIL,4,1
                ENTRY      TOO.
                LDY        #D.MODDT+3
RIPTIME         LDA        DATELO,X
                STA        (DRBUFPL),Y
                DEY
                DEX
                BPL        RIPTIME             ;MOVE ALL FOR BYTES...
RUPDATE         JSR        WRTGBUF             ;WRITE UPDATED ENTRY BACK TO DISK. (ASSUMES BLOKNM UNDISTURBEDD)
                BCS        DERROR1             ;GIVE UP ON ANY ERROR.
                LDY        #D.DHDR             ;NOW COMPARE CURRENT BLOCK NUMBER TO THIS
                LDA        (DRBUFPL),Y         ; ENTRY'S HEADER BLOCK
                INY
                CMP        BLOKNML             ;ARE LOW ADDRESSES THE SAME?
                STA        BLOKNML             ;(SAVE IT IN CASE IT'S NOT)
                BNE        RIPPLE2             ;BRANCH IF ENTRY DOES NOT RESIDE IN SAME BLOCK AS HEADER.
                LDA        (DRBUFPL),Y         ;CHECK HIGH ADDRESS JUST TO BE SURE.
                CMP        BLOKNMH
                BEQ        RIPPLE              ;THEY ARE THE SAME, CONTINUE RIPPLE TO ROOT DIRECTORY.
RIPPLE2         LDA        (DRBUFPL),Y         ;THEY AREN'T THE SAME, READ IN THIS DIRECTORY'S HEADER.
                STA        BLOKNMH
                JSR        RDGBUF
                BCC        RIPPLE              ;CONTINUE IF READ WAS GOOD.
DERROR1         EQU        *
                RTS
                PAGE
*
NOTDIR          LDA        #TYPERR             ;NOT TREE OR DIRECTORY- NOT A RECOGNIZED TYPE!
TSTERR          SEC
                RTS                            ;DO NOTHING.
*
*
TSTSOS          LDA        GBUF                ;TEST SOS STAMP
                CMP        SOSTMPL
                BNE        TSTERR
                LDA        GBUF+1
                CMP        SOSTMPH
                BNE        TSTERR
                LDA        GBUF+4              ;TEST FOR HEADER
                AND        #$E0
                CMP        #HEDTYP*16
                BNE        TSTERR              ;BRANCH IF NOT SOS HEADER (NO ERROR NUMBER)
DRVISDNE        CLC                            ;INDICATE NO ERROR./
                RTS
*
                CHN        FNDFIL,4,1


*************************************************************************
```

```
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: CREATE
*************************************************************************

```

```
==============================================================================
DOCUMENT :SOS1.3.3of5.THREE:SOS.EQUATES.TXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: EQUATES
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

*
                ENTRY      BFMGR
*
* BFM INITIALIZATION ENTRIES
* (INIT CODE FOUND IN INIT.SRC)
*
                ENTRY      BFMFCB1             ; FCB PAGE 1 ADDR
                ENTRY      BFMFCB2             ; AND PAGE 2
                ENTRY      FCBZPP
                ENTRY      SISTER
                ENTRY      PATHBUF
                ENTRY      VCB
                ENTRY      WORKSPC
                ENTRY      PFIXPTR
                ENTRY      BMAPAGE
                ENTRY      BMBPAGE
                ENTRY      FCBADDRH
                ENTRY      BMAMADR
                ENTRY      BMBMADR
*
*
                EXTRN      LEVEL               ; FILE LEVEL (LOW BYTE)
                EXTRN      OPMSGRPLY           ; OPERATOR MESSAGE
                EXTRN      DATETIME            ; THANKS TOM...
                EXTRN      DMGR                ; THANKS BOB...
                EXTRN      REQBUF              ;   "
                EXTRN      REQFXBUF            ;   "
                EXTRN      GETBUFADR           ;  "
                EXTRN      RELBUF              ;   "
                EXTRN      BLKDLST             ;   "
                EXTRN      SERR
                EXTRN      BACKMASK
*
* ERRORS
*
                EXTRN      SYSERR
*
                EXTRN      BADPATH             ; INVALID PATHNAME SYNTAX
                EXTRN      FCBFULL             ; FILE CONTROL BLOCK FULL
                EXTRN      BADREFNUM           ; INVALID REFNUM
                EXTRN      PATHNOTFND           ; PATHNAME NOT FOUND
                EXTRN      VNFERR               ; VOLUME NOT FOUND
                EXTRN      FNFERR               ; FILE NOT FOUND
                EXTRN      DUPERR               ; DUPLICATE FILE NAME ERROR
                EXTRN      DUPVOL               ; DUPLICATE VOLUME CAN'T BE LOGGED IN.
                EXTRN      OVRERR               ; NOT ENOUGH DISK SPACE FOR PREALLOCATION
                EXTRN      DIRFULL              ; DIRECTORY FULL ERROR
                EXTRN      CPTERR               ; FILE INCOMPATIBLE SOS VERSION
                EXTRN      TYPERR               ; NOT CURRENTLY SUPPORTED FILE TYPE
                EXTRN      EOFERR               ; POSITION ATTEMPTED BEYOND END OF FILE
                EXTRN      POSNERR              ; ILLEGAL POSITION (L.T. 0 OR G.T. $FFFFFF)
                EXTRN      ACCSERR              ; FILE ACCESS R/W REQUEST CONFLICTS WITH ATTRIBUTES.
                EXTRN      BTSERR               ; USER SUPPLIED BUFFER TOO SMALL
                EXTRN      FILBUSY              ; EITHER WRITE WAS REQUESTED OR WRITE ACCESS ALREADY ALLOCATED.
                EXTRN      NOTSOS               ; NOT A SOS DISKETTE
                EXTRN      BADLSTCNT            ; INVALID VALUE IN LIST PARAMETER
                EXTRN      XDISKSW              ; DISK SWITCHED
                EXTRN      NOTBLKDEV            ; NOT A BLOCK DEVICE
                EXTRN      XNOWRITE             ; DISK/MEDIA IS HARDWARE WRITE PROTECTED
                EXTRN      XIOERROR             ; INFORMATION ON BLOCK DEVICE NOT ACCESSABLE
                EXTRN      DIRERR               ; DIRECTORY ENTRY COUNT INCONSISTENT WITH ACTUAL ENTRIES
                EXTRN      BITMAPADR            ; BIT MAP DISK ADDRESS IMPOSSIBLE
*
* FATAL ERRORS
*
                EXTRN      SYSDEATH
*
                EXTRN      VCBERR               ; VOLUME CONTROL BLOCK NOT USABLE
                EXTRN      ALCERR               ; ALLOCATION BLOCKS INVALID
                EXTRN      TOOLONG              ; PATHNAME BUFFER OVERFLOW
                PAGE
*
* CONSTANTS
*
DLIMIT          EQU        $2F                  ; DELIMITER IS CURRENTLY AN ASCII '/'
SEEDTYP         EQU        1
SAPTYP          EQU        2
TRETYP          EQU        3
DIRTYP          EQU        $D
HEDTYP          EQU        $E
RDCMD           EQU        $0
WRTCMD          EQU        $1
RPTCMD          EQU        $9
STATCMD         EQU        $02                  ; REQUEST STATUS OF BLOCK DEVICE. (BIT 0 = WRITE PROTECTED)
STATSUB         EQU        $0
PRETIME         EQU        $20                  ; COMMAND NEEDS CURRENT DATE/TIME STAMP
PREREF          EQU        $40                  ; COMMAND REQUIRES FCB ADDRESS AND VERIFICATION
PREPATH         EQU        $80                  ; COMMAND HAS PATHNAME TO PREPROCESS
SISTER          EQU        $1400
*
```

```
* VOLUME STATUS CONSTANTS (BITS)
*
DSWITCH         EQU        $40                   ; FOR DISK SWITCHED ERROR RECOVERY.
*
* FILE STATUS CONSTANTS
*
DATALC          EQU        $1                    ; DATA BLOCK NOT ALLOCATED.
IDXALC          EQU        $2                    ; INDEX NOT ALLOCATED
TOPALC          EQU        $4                    ; TOP INDEX NOT ALLOCATED
STPMOD          EQU        $8                    ; STORAGE TYPE MODIFIED
USEMOD          EQU        $10                   ; FILE USAGE MODIFIED
EOFMOD          EQU        $20                   ; END OF FILE MODIFIED
DATMOD          EQU        $40                   ; DATA BLOCK MODIFIED
IDXMOD          EQU        $80                   ; INDEX BLOCK MODIFIED
FCBMOD          EQU        $80                   ; HAS FCB/DIRECTORY BEEN MODIFIED? (FLUSH)
*
* FILE ATTRIBUTES CONSTANTS
*
READEN          EQU        $1                    ; READ ENABLED
WRITEN          EQU        $2                    ; WRITE ENABLED
NLINEN          EQU        $10                   ; NEW LINE ENABLED
BKBITVAL        EQU        $20                   ; FILE NEEDS BACKUP IF SET (BKBITFLG)
RENAMEN         EQU        $40                   ; RENAME OK WHEN ON.
DSTROYEN        EQU        $80                   ; DESTROY OK WHEN ON.
                PAGE
* HEADER INDEX CONSTANTS
*
HNLEN           EQU        $0                    ; HEADER NAME LENGTH (OFFSET INTO HEADER)
*HNAME EQU $1 ; HEADER NAME
HPENAB          EQU        $10                   ; PASSWORD ENABLE BYTE
HPASS           EQU        $11                   ; ENCODED PASSWORD
HCRDT           EQU        $18                   ; HEADER CREATION DATE
* HCRTM EQU $1A ; HEADER CREATION TIME
HVER            EQU        $1C                   ; SOS VERSION THAT CREATED DIRECTORY
HCMP            EQU        $1D                   ; BACKWARD COMPATIBLE WITH SOS VERSION
HATTR           EQU        $1E                   ; HEADER ATTRIBUTES- PROTECT ETC.
* HENTLN EQU $1F ; LENGTH OF EACH ENTRY
* HMENT EQU $20 ; MAXIMUM NUMBER OF ENTRIES/BLOCK
HCENT           EQU        $21                   ; CURRENT NUMBER OF FILES IN DIRECTORY
HRBLK           EQU        $23                   ; OWNER'S DIRECTORY ADDRESS
HRENT           EQU        $25                   ; OWNER'S DIRECTORY ENTRY NUMBER
HRELN           EQU        $26                   ; OWNER'S DIRECTORY ENTRY LENGTH
VBMAP           EQU        HRBLK
VTBLK           EQU        HRENT                 ; (USED FOR ROOT DIRECTORY ONLY)
*
* VOLUME CONTROL BLOCK INDEX CONSTANTS
*
VCBSIZE         EQU        $20                   ; CURRENT VCB IS 32 BYTES PER ENTRY (VER 0)
VCBNML          EQU        0                     ; VOLUME NAME LENGTH BYTE
VCBNAM          EQU        1                     ; VOLUME NAME
```

```
VCBDEV          EQU        $10                  ; VOLUME'S DEVICE
VCBSTAT         EQU        $11                  ; VOLUME STATUS. (80=FILES OPEN. 40=DISK SWITCHED.)
VCBTBLK         EQU        $12                  ; TOTAL BLOCKS ON THIS VOLUME
VCBTFRE         EQU        $14                  ; NUMBER OF UNUSED BLOCKS
VCBROOT         EQU        $16                  ; ROOT DIRECTORY (DISK) ADDRESS
*VCBMORG EQU $18 ; MAP ORGANIZATION (NOT SUPPORTED BY V 0)
*VCBMBUF EQU $19 ; BIT MAP BUF NUM
VCBDMAP         EQU        $1A                  ; FIRST (DISK) ADDRESS OF BITMAP(S)
VCBCMAP         EQU        $1C                  ; RELATIVE ADDRESS OF BIT MAP WITH SPACE (ADD TO VCBDMAP)
*VCBMNUM EQU $1D ; RELATIVE BIT MAP CURRENTLY IN MEMORY
VCBOPNC         EQU        $1E                  ; CURRENT NUMBER OF OPEN FILES.
VCBSWAP         EQU        $1F                  ; $8X IF VOLUME SWAPPED; $00 IF UNSWAPPED WHERE X=LOW ORDER BYTE OF VCB
ADR/16
*
* FILE CONTROL BLOCK INDEX CONSTANTS
*
FCBREFN         EQU        0                    ; FILE REFERENCE NUMBER (POSITION SENSITIVE)
FCBDEVN         EQU        1                    ; DEVICE (NUMBER) ON WHICH FILE RESIDES
*FCBHEAD EQU 2 ; BLOCK ADDRESS OF FILE'S DIRECTORY HEADER
*FCBDIRB EQU 4 ; BLOCK ADDRESS OF FILE'S DIRECTORY
FCBENTN         EQU        6                    ; ENTRY NUMBER WITHIN DIRECTORY BLOCK
FCBSTYP         EQU        7                    ; STORAGE TYPE - SEED, SAPLING, TREE, ETC.
FCBSTAT         EQU        8                    ; STATUS - INDEX/DATA/EOF/USAGE/TYPE MODIFIED.
FCBATTR         EQU        9                    ; ATTRIBUTES - READ/WRITE ENABLE, NEWLINE ENABLE.
FCBNEWL         EQU        $A                   ; NEW LINE TERMINATOR (ALL 8 BITS SIGNIFICANT).
FCBBUFN         EQU        $B                   ; BUFFER NUMBER
FCBFRST         EQU        $C                   ; FIRST BLOCK OF FILE
FCBIDXB         EQU        $E                   ; BLOCK ADDRESS OF INDEX (0 IF NO INDEX)
FCBDATB         EQU        $10                  ; BLOCK ADDRESS OF DATA
FCBMARK         EQU        $12                  ; CURRENT FILE MARKER.
FCBEOF          EQU        $15                  ; LOGICAL END OF FILE.
FCBUSE          EQU        $18                  ; ACTUAL NUMBER OF BLOCKS ALLOCATED TO THIS FILE.
FCBSWAP         EQU        $1A                  ; $8N = SWAPPED, $00 = UNSWAPPED VOLUME ("N" = VCB ENTRY NUMBER)
FCBLEVL         EQU        $1B                  ; LEVEL AT WHICH THIS FILE WAS OPENED
FCBDIRTY        EQU        $1C                  ; FCB MARKED AS MODIFIED
                PAGE
*
* ZERO PAGE STUFF
*
PAR             EQU        $A0
COMMAND         EQU        PAR
C.DNAMP         EQU        PAR+1
C.PATH          EQU        PAR+1
C.REFNUM        EQU        PAR+1
C.ISNEWL        EQU        PAR+2
C.OUTEOF        EQU        PAR+2
C.BASE          EQU        PAR+2
C.MRKPTR        EQU        PAR+2
C.OUTBUF        EQU        PAR+2
C.NWPATH        EQU        PAR+3
```

```
C.FILIST        EQU        PAR+3
C.NEWL          EQU        PAR+3
C.OUTVOL        EQU        PAR+3
C.OUTREF        EQU        PAR+3
C.XLIST         EQU        PAR+3
C.MAXPTH        EQU        PAR+3
C.MARK          EQU        PAR+3
C.NEWEOF        EQU        PAR+3
C.BYTES         EQU        PAR+4
C.FILSTLN       EQU        PAR+5
C.OUTBLK        EQU        PAR+5
C.OPLIST        EQU        PAR+5
C.XLEN          EQU        PAR+5
C.FILID         EQU        PAR+6
C.OUTCNT        EQU        PAR+6
C.OPLSTLN       EQU        PAR+7
C.AUXID         EQU        PAR+7
C.STOR          EQU        PAR+9
C.EOFLL         EQU        PAR+$A
C.EOFLH         EQU        PAR+$B
C.EOFHL         EQU        PAR+$C
DEBUPTR         EQU        PAR+$D               ; NOTE SAME AS BELOW
C.EOFHH         EQU        PAR+$D
* C.SPARE EQU PAR+$E
*
DEVICE          EQU        $C0
DHPCMD          EQU        DEVICE
UNITNUM         EQU        DEVICE+1
DSTATREQ        EQU        DEVICE+2
DBUFPL          EQU        DEVICE+2
DBUFPH          EQU        DBUFPL+1
DSTATBFL        EQU        DEVICE+3             ; TO PASS BACK BUSY, WRITE PROTECT, READ PROTECT.
DSTATBFH        EQU        DSTATBFL+1
RQCNTL          EQU        DEVICE+4
RQCNTH          EQU        RQCNTL+1
BLOKNML         EQU        DEVICE+6
BLOKNMH         EQU        BLOKNML+1
BRDPTR          EQU        DEVICE+8             ; (AND 9)
*
DVNAMP          EQU        DEVICE+1             ; USED FOR 'VOLUME' TO CALL
DVDNUM          EQU        DEVICE+3             ; 'GET.DNUM' IN DEVICE MANAGER.
*
SISBPH          EQU        SISTER+DBUFPH
SISDSTAT        EQU        SISTER+DSTATBFH
SSBRDPH         EQU        SISTER+BRDPTR+1
*
                PAGE
*
* ZERO PAGE TEMPORARIES
*
```

```
ZTEMPS          EQU        $B0
PATHNML         EQU        ZTEMPS
PATHNMH         EQU        PATHNML+1
USRBUF          EQU        ZTEMPS
TPATH           EQU        ZTEMPS+2
WRKPATH         EQU        ZTEMPS+4
TINDX           EQU        ZTEMPS+2
DRBUFPL         EQU        ZTEMPS+4
DRBUFPH         EQU        DRBUFPL+1
VCBPTR          EQU        ZTEMPS+6
BMADR           EQU        ZTEMPS+8
FCBPTR          EQU        ZTEMPS+$A
DATPTR          EQU        ZTEMPS+$C
POSPTR          EQU        ZTEMPS+$E
*
MAXTEMPS        EQU        $F
SISTEMPS        EQU        SISTER+ZTEMPS
SSTIDXH         EQU        SISTER+TINDX+1
SISPATH         EQU        SISTER+C.PATH+1
SSNWPATH        EQU        SISTER+C.NWPATH+1
SISUSRBF        EQU        SISTER+USRBUF+1
SISOUTBF        EQU        SISTER+C.OUTBUF+1
SISTPATH        EQU        SISTER+TPATH+1
SISBMADR        EQU        SISTER+BMADR+1
SISFCBP         EQU        SISTER+FCBPTR+1
SISDATP         EQU        SISTER+DATPTR+1
SISPOSP         EQU        SISTER+POSPTR+1
*
*
* ADDRESSES
*
PATHBUF         EQU        $1000                ; NOTE: THIS IS $100 BYTES LONG.
VCB             EQU        $1100
GBUF            EQU        $1200                ; THRU $13FF
*
* INITIALIZATION EQUATES
*
BFMFCB1         EQU        $1C                  ; FCB PAGE 1 ADDR
BFMFCB2         EQU        $1D                  ; FCB PAGE 2 ADDR
BMAPAGE         EQU        <$B800               ; BIT MAP A ADDR
BMBPAGE         EQU        <$BA00               ; BIT MAP B ADDR
FCBZPP          EQU        FCBPTR
*
*
*
                PAGE
                DSECT
                ORG        $0                   ; (THE FOLLOWING DO NOT NEED TO BE ON ZERO PAGE. 7/16/80 JRH.)
DATBLKL         DS         1
DATBLKH         DS         1
```

```
IDXADRL         DS         1                    ; DISK ADDRESS OF INDEX BLOCK
IDXADRH         DS         1
REQL            DS         1
REQH            DS         1
INDXBLK         DS         1
LEVELS          DS         1
TOTENT          DS         1
ENTCNTL         DS         1
ENTCNTH         DS         1
CNTENT          DS         1
NOFREE          DS         1
BMCNT           DS         1
SAPTR           DS         1
TREPTR          DS         1
TLINK           DS         2
FLINK           DS         2
PATHCNT         DS         1
PFIXPTR         DS         2
BMPTR           DS         1
BASVAL          DS         1
HALF            DS         1
*
*
                PAGE
*
* BIT MAP INFO TABLES (A & B)
*
BMTABSZ         EQU        $6
BMTAB           DS         1
BMBUFBNK        DS         1
BMASTAT         DS         1
BMADEV          DS         1
BMAMADR         DS         1
BMADADR         DS         2
BMACMAP         DS         1                    ; SIMILAR TO VCBCMAP
BMBSTAT         DS         1
BMBDEV          DS         1
BMBMADR         DS         1
                DS         2                    ; BMBDADR
                DS         1                    ; BMBCMAP
*
FCBADDRH        DS         1                    ; FILE CONTROL BLOCK'S BUFFER ADDRESS.
FCBANKNM        DS         1                    ; AND BANK (SISTER PAGE) BYTE.
TPOSLL          DS         1
TPOSLH          DS         1
TPOSHI          DS         1
RWREQL          DS         1
RWREQH          DS         1
BULKCNT         DS         1
NLCHAR          DS         1
```

```
NPATHDEV        DS         3                    ; FOR NEW PATHNAME DEVICE AND DIRECTORY HEADER ADDRESS
IOACCESS        DS         1                    ; USED TO DETERMINE IF A CALL HAS BEEN MADE TO THE DISK DEVICE HANDLER
DEVNUM          DS         1                    ; CURRENT DEVICE TO BE ACCESSED.
TOTDEVS         DS         1                    ; USED FOR ACCESSING DRIVES IN NUMERIC ORDER
CMDTEMP         DS         1                    ; USED FOR TESTING REFNUM, TIME, AND DSKSWTCH (PRE)PROCESSING.
DATELO          DS         1                    ; DATE AND TIME MUST RESIDE ON ZERO PAGE.
DATEHI          DS         1
TIMELO          DS         1
TIMEHI          DS         1
*
DUPLFLAG        DS         1                    ; USED FOR DIFFERENCE BETWEEN VNFERR AND DUPVOL BY SYNPATH
ZPGTEMP         DS         1                    ; A ONE-BYTE UNSTABLE TEMPORARY
VCBENTRY        DS         1                    ; POINTER TO CURRENT VCB ENTRY
*
                DEND
*
                CHN        PATH,4,1

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: EQUATES
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.3of5.THREE:SOS.FNDFIL.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: FNDFIL
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
*
*
FINDFILE        JSR        LOOKFILE             ; SEE IF FILE EXISTS
                BCS        NOFIND               ; BRANCH IF AN ERROR WAS ENCOUNTERED
MOVENTRY        LDY        H.ENTLN              ; MOVE ENTIRE ENTRY INFO TO A SAFE AREA
MOVENT1         LDA        (DRBUFPL),Y
                STA        DFIL+D.STOR,Y
                DEY
                BPL        MOVENT1
                LDA        #0                   ; TO INDICATE ALL IS WELL
NOFIND          RTS                             ; RETURN CONDITION CODES.
                PAGE
*
*
LOOKFILE        JSR        PREPROOT             ; FIND VOLUME AND SET UP OTHER BORING STUFF
                BCS        FNDERR               ; PASS BACK ANY ERROR ENCOUNTERED
                LDY        #0                   ; TEST TO SEE IF ONLY ROOT WAS SPECIFIED.
                LDA        (PATHNML),Y
                BNE        LOOKFIL0             ; BRANCH IF MORE THAN ROOT.
                LDA        #GBUF/256            ;  OTHERWISE, REPORT A BADPATH ERROR
                STA        DRBUFPH              ; (BUT FIRST CREATE A PHANTOM ENTRY FOR OPEN)
                LDA        #4
                STA        DRBUFPL
                LDY        #D.AUXID             ; FIRST MOVE IN ID, AND DATE STUFF.
PHANTM1         LDA        (DRBUFPL),Y
                STA        DFIL,Y
                DEY
                CPY        #D.CREDT-1
                BNE        PHANTM1
PHANTM2         LDA        ROOTSTUF-D.FILID,Y
                STA        DFIL,Y
                DEY
                CPY        #D.FILID-1
                BNE        PHANTM2
                LDA        #DIRTYP*$10          ; FAKE DIRECTORY FILE
                STA        DFIL+D.STOR
                LDA        #BADPATH             ; (CARRY IS SET)
                RTS
*
```

```
ROOTSTUF        DFB        0,2,0,4
                DFB        0,0,8,0
*
LOOKFIL0        LDA        #0                    ; RESET FREE ENTRY INDICATOR
                STA        NOFREE
                SEC                              ; INDICATE THAT THE DIRECTORY TO BE SEARCHED HAS HEADER IN THIS BLOCK
LOOKFIL1        LDA        #0                    ; RESET ENTRY COUNTER
                STA        TOTENT
                JSR        LOOKNAM               ; LOOK FOR NAME POINTED TO BY 'PATHNML'
                BCC        NAMFOJMP              ; BRANCH IF NAME WAS FOUND.
                LDA        ENTCNTL               ; HAVE WE LOOKED AT ALL OF THE
                SBC        TOTENT                ;  ENTRIES IN THIS DIRECTORY?
                BCC        DCRENTH               ; MAYBE, CHECK HI COUNT.
                BNE        LOOKFIL2              ; NO, READ NEXT DIRECTORY BLOCK
                CMP        ENTCNTH               ; HAS THE LAST ENTRY BEEN LOOKED AT (ACC=0)
                BEQ        ERRFNF                ; YES, GIVE 'FILE NOT FOUND' ERROR.
                BNE        LOOKFIL2              ; BRANCH ALWAYS.
DCRENTH         DEC        ENTCNTH               ; SHOULD BE AT LEAST 1
                BPL        LOOKFIL2              ; (THIS SHOULD BE BRANCH ALWAYS...)
ERRDIR          LDA        #DIRERR               ; REPORT DIRECTORY MESSED UP.
FNDERR          SEC                              ; INDICATE ERROR HAS BEEN ENCOUNTERED.
                RTS
NAMFOJMP        JMP        NAMFOUND              ; AVOID BRANCH OUT OF RANGE
*
                PAGE
LOOKFIL2        STA        ENTCNTL               ; KEEP RUNNING COUNT
                LDA        #GBUF/256             ; RESET INDIRECT POINTER
                STA        DRBUFPH
                LDA        GBUF+2                ; GET LINK TO NEXT DIRECTORY BLOCK
                BNE        NXTDIR0               ; (IF THERE IS ONE)
                CMP        GBUF+3                ; ARE BOTH ZERO, I.E. NO LINK?
                BEQ        ERRDIR                ; IF SO, THEN NOT ALL ENTRIES WERE ACCOUNTED FOR.
NXTDIR0         STA        BLOKNML
                LDA        GBUF+3
                STA        BLOKNMH
                JSR        RDGBUF                ; GO READ THE NEXT LINKED DIRECTORY IN.
                BCC        LOOKFIL1              ; BRANCH IF NO ERROR.
                RTS                              ; RETURN ERROR (IN ACCUMULATOR).
TELFREEX        JMP        TELFREE
*
FNF0X           JMP        FNF0                  ; AVOID BRANCH OUT OF RANGE
*
CFLAG           DS         1                     ; AM I CREATING?
TTSAVE          DS         2                     ; CURRENT BLOCK ADDR
BLOKSAVE        DS         2                     ; PARENT DIR ADDR
*
ERRFNF          LDA        NOFREE                ; WAS ANY FREE ENTRY FOUND?
                BNE        FNF0X
                LDA        GBUF+2                ; TEST LINK
                BNE        TELFREEX
                CMP        GBUF+3               ; IF BOTH ARE ZERO, THEN GIVE UP
                BNE        TELFREEX             ; BRANCH IF NOT LAST DIR BLOCK
                LDA        CFLAG                ; DOING A CREATE?
                BEQ        FNF0X                ; NO, SIMPLY REPORT NOT FOUND
*
* EXTEND THE DIRECTORY BY A BLOCK
*
                LDA        BLOKSAVE             ; BUT NOT
                ORA        BLOKSAVE+1           ;    IF A ROOT DIRECTORY!
                BEQ        FNF0X                ;        FORU BLOCKS HARD CODED
                LDA        TTLINK               ; FETCH CURRENT DIRECTORY
                STA        TLINK                ; ADDR (GBUF)
                LDA        TTLINK+1             ; AND ALLLOCATE A NEW
                STA        TLINK+1              ; BY LINKING TO CURRENT
                JSR        DIRWRT
                BCS        FNF0                 ; RATS! NO SPACE SAY "DIRFULL"
*
* SAVE CURRENT BLOCK ADDR
*
                LDA        TTLINK
                STA        TTSAVE
                LDA        TTLINK+1
                STA        TTSAVE+1
*
* FETCH DESCENDENT
*
                LDA        GBUF+2
                STA        BLOKNML
                LDA        GBUF+3
                STA        BLOKNMH
                JSR        ZERGBUF              ; INIT THE NEW DIR BLOCK
*
* AND INSERT BACK POINTER
* TO "CURRENT BLOCK"
*
                LDA        TTSAVE
                STA        GBUF
                LDA        TTSAVE+1
                STA        GBUF+1
                JSR        WRTGBUF
                BCS        ERTS
*
* UPDATE DIR'S HEADER IN PARENT
*
                LDA        BLOKSAVE
                STA        BLOKNML              ; PREPARE TO READ PARENT
                LDX        BLOKSAVE+1
                STX        BLOKNMH
                JSR        RDGBUF               ; FETCH PARENT
                LDY        #D.USAGE             ; BUMP BLOCKS USED BY HEADER
                LDA       (DEBUPTR),Y
                SEC
                ADC       #0                     ; BY JUST ONE BLOCK
                STA       (DEBUPTR),Y
                INY
                LDA       (DEBUPTR),Y            ; TWO BYTE BLOCKS USED
                ADC       #0
                STA       (DEBUPTR),Y
                LDY       #D.EOF+1               ; INCREASE EOF BY $200
                LDA       (DEBUPTR),Y
                CLC
                ADC       #2
                STA       (DEBUPTR),Y
                INY
                LDA       (DEBUPTR),Y
                ADC       #0
                STA       (DEBUPTR),Y
                JSR       WRTGBUF                ; REWRITE PARENT DIR BLOCK
                LDA       TTSAVE+1               ; REFETCH CURRENT DIR BLOCK
                STA       BLOKNMH
                LDA       TTSAVE
                STA       BLOKNML
                JSR       RDGBUF                 ; BACK FROM THE SHADOWS AGAIN
                JMP       ERRFNF                 ; VOILA! WE HAVE EXTENDED THE DIRECTORY!
*
TELFREE         STA       D.ENTBLK
                LDA       GBUF+3
                STA       D.ENTBLK+1             ; ASSUME FIRST ENTRY OF NEXT BLOCK
                LDA       #1                     ;  IS FREE FOR USE.
                STA       D.ENTNUM
                STA       NOFREE                 ;  MARK D.ENTNUM AS VALID (FOR CREATE)
FNF0            LDY       #0                     ; TEST FOR 'FILE NOT FOUND' VERSUS 'PATH NOT FOUND'
                LDA       (PATHNML),Y
                TAY
                INY
                LDA       (PATHNML),Y            ; IF NON-ZERO THEN 'PATH NOT FOUND'
ERRPATH1        SEC                              ; IN EITHER CASE, INDICATE ERROR.
                BEQ       FNF1
                LDA       #PATHNOTFND            ; REPORT NO SUCH PATH.
ERTS            RTS
FNF1            LDA       #FNFERR                ; REPORT FILE NOT FOUND.
                RTS
                PAGE
*
NAMFOUND        LDA       (PATHNML),Y            ; (Y=0)
                SEC
                ADC       PATHNML                ; TEST FOR LAST NAME IN PATH
                TAY                              ; IF ZERO, THEN THAT WAS LAST NAME
                CLC                              ; TO INDICATE SUCCESS
                LDA       PATHBUF,Y
                BEQ        FILFOUND
*NOW CHANGE THE PATHNAME POINTER TO POINT AT THE NEXT NAME IN THE PATH
                STY        PATHNML
                LDA        DRBUFPL              ; SAVE PARENTS
                STA        DEBUPTR              ; ENTRY POINTER
                LDA        DRBUFPH
                STA        DEBUPTR+1            ; IN CASE ENTRY ON PAGE 2
                LDA        BLOKNML              ; ADDRESS (DIR EXTEND)
                STA        BLOKSAVE
                LDA        BLOKNMH
                STA        BLOKSAVE+1
                LDY        #D.STOR              ; BE SURE THIS IS A DIRECTORY ENTRY
                LDA        (DRBUFPL),Y          ; HIGH NIBBLE WILL TELL
                AND        #$F0
                CMP        #DIRTYP*16           ; IS IT A SUB-DIRECTORY?
                BNE        ERRPATH1             ; REPORT THE USER'S MISTAKE
                LDY        #D.FRST              ; GET ADDRESS OF FIRST SUB-DIRECTORY BLOCK
                LDA        (DRBUFPL),Y
                STA        BLOKNML              ; (NO CHECKING IS DONE HERE FOR A VALID
                INY                             ;  BLOCK NUMBER... )
                STA        D.HEAD               ; SAVE AS FILE'S HEADER BLOCK TOO.
                LDA        (DRBUFPL),Y
                STA        BLOKNMH
                STA        D.HEAD+1
                JSR        RDGBUF               ; READ SUB-DIRECTORY INTO GBUF
                BCS        FNDERR1              ; RETURN IMMEDIATELY ANY ERROR ENCOUNTERED.
                LDA        GBUF+HCENT+4         ; GET THE NUMBER OF FILES
                STA        ENTCNTL              ;  CONTAINED IN THIS DIRECTORY
                LDA        GBUF+HCENT+5
                STA        ENTCNTH
                LDA        GBUF+HCMP+4          ; TEST BACKWARD COMPATIBILITY
                BEQ        MOVHEAD
ERRCOMP         LDA        #CPTERR              ; TELL THEM THIS DIRECTORY IS NOT COMPATABLE
NONAME          EQU        *
FNDERR1         SEC
                RTS
MOVHEAD         JSR        MOVHED0              ; MOVE INFO ABOUT THIS DIRECTORY
                JMP        LOOKFIL0             ; DO NEXT LOCAL PATHNAME
*
MOVHED0         LDX        #$A                  ; MOVE INFO ABOUT THIS DIRECTORY
MOVHED1         LDA        GBUF+HCRDT+4,X
                STA        H.CREDT,X
                DEX
                BPL        MOVHED1
                RTS
*
                PAGE
*
*
FILFOUND        EQU        *
```

```
ENTADR          LDA        H.MAXENT             ; FIGURE OUT WHICH IS ENTRY NUMBER THIS IS.
                SEC
                SBC        CNTENT               ; MAX ENTRIES - COUNT ENTRIES + 1 = ENTRY NUMBER
                ADC        #0                   ; (CARRY IS/WAS SET)
                STA        D.ENTNUM
                LDA        BLOKNML
                STA        D.ENTBLK
                LDA        BLOKNMH              ; AND INDICATE BLOCK NUMBER OF THIS DIRECTORY.
                STA        D.ENTBLK+1
                CLC
                RTS
*
LOOKNAM         LDA        H.MAXENT             ; RESET COUNT OF FILES PER BLOCK
                STA        CNTENT
                LDA        #GBUF/256
                STA        DRBUFPH
                LDA        #4
LOKNAM1         STA        DRBUFPL              ; RESET INDIRECT POINTER TO GBUF
                BCS        LOKNAM2              ; BRANCH IF THIS BLOCK CONTAINS A HEADER
                LDY        #D.STOR
                LDA        (DRBUFPL),Y          ; GET LENGTH OF NAME IN DIRECTORY
                BNE        ISNAME               ; BRANCH IF THERE IS A NAME.
                LDA        NOFREE               ; TEST TO SEE IF A FREE ENTRY HAS BEEN DECLARED.
                BNE        LOKNAM2              ; YES BUMP TO NEXT ENTRY
                JSR        ENTADR               ; SET ADDRESS FOR CURRENT ENTRY
                INC        NOFREE               ; INDICATE A FREE SPOT HAS BEEN FOUND
                BNE        LOKNAM2              ; BRANCH ALWAYS.
*
ISNAME          AND        #$F                  ; STRIP TYPE (THIS IS CHECKED BY 'FILFOUND')
                INC        TOTENT               ; (BUMP COUNT OF VALID FILES FOUND)
                CMP        (PATHNML),Y          ; ARE BOTH NAMES OF THE SAME LENGTH?
                BNE        LOKNAM2              ; NO, BUMP TO NEXT ENTRY
                TAY
CMPNAME         LDA        (DRBUFPL),Y          ; COMPARE NAMES LETTER BY LETTER
                CMP        (PATHNML),Y
                BNE        LOKNAM2
                DEY                             ; HAVE ALL LETTERS BEEN COMPARED?
                BNE        CMPNAME              ; NO, CONTINUE..
                CLC                             ; BY GOLLY, WE GOT US A MATCH!
                RTS
*
LOKNAM2         DEC        CNTENT               ; HAVE WE CHECKED ALL POSSIBLE ENTRIES IN THIS BLOCK?
                BEQ        NONAME               ; YES, GIVE UP.
                LDA        H.ENTLN              ; ADD ENTRY LENGTH TO CURRENT POINTER
                CLC
                ADC        DRBUFPL
                BCC        LOKNAM1              ; BRANCH IF WE'RE STILL IN THE FIRST PAGE.
                INC        DRBUFPH              ; LOOK ON SECOND PAGE
                CLC                             ; CARRY SHOULD ALWAYS BE CLEAR BEFORE LOOKING AT NEXT.
                BCC        LOKNAM1              ; BRANCH ALWAYS...
                PAGE
*
*
PREPROOT        JSR       FINDVOL              ; FIND CORRECT VOLUME AND DEVICE NUMBER
                BCC       ROOT1                ; BRANCH IF IT WAS FOUND.
ROOT0           JSR       LOOKVOL              ; OTHERWISE LOOK ON ALL DEVICES.
                BCS       SRITZ                ; CAN'T FIND IT.
ROOT1           LDA       #0                   ; ZERO OUT DIRECTORY TEMPS
                LDY       #42                  ; (DECIMAL)
CLRDSP          STA       D.DEV,Y
                DEY
                BPL       CLRDSP
                LDY       #VCBDEV              ; SET UP DEVICE NUMBER
                LDA       (VCBPTR),Y
                STA       DEVNUM
                STA       D.DEV                ; FOR FUTURE REFERENCE
                INY
                LDA       (VCBPTR),Y           ; GET CURRENT STATUS OF THIS VOLUME
                STA       V.STATUS
                LDY       #VCBROOT             ; GET BLOCK ADDRESS OF ROOT DIRECTORY TOO.
                LDA       (VCBPTR),Y
                STA       BLOKNML
                STA       D.HEAD               ; PRESERVE AS HEADER
                INY
                LDA       (VCBPTR),Y
                STA       BLOKNMH
                STA       D.HEAD+1
                JSR       RDGBUF               ; GO READ IN ROOT
                BCC       ROOT2                ; BRANCH IF NO ERROR
                PHA                            ; SAVE ERROR CODE
                LDY       #VCBSTAT             ; CHECK THIS BUGGER FOR AN OPEN FILE.
                LDA       (VCBPTR),Y
                ASL       A                    ; (SHIFT OPEN STATUS INTO CARRY)
                PLA                            ; GET ERROR CODE AGAIN
                BCS       ROOTERR              ;  BRANCH IF ERROR NEEDS TO BE REPORTED
                BNE       ROOT0                ; OTHERWISE, LOOK ELSEWHERE (BRANCH ALWAYS).
*
ROOT2           JSR       CHKROOT              ; VERIFY ROOT NAME
                BEQ       ROOT3                ; BRANCH IF MATCHED.
                LDY       #VCBSTAT             ; TEST FOR OPEN FILES ON THIS VOLUME BEFORE
                LDA       (VCBPTR),Y           ;  LOOKING FOR IT ELSEWHERE.
                BPL       ROOT0
                JSR       USRREQ               ;  REQUEST USER MOUNT VOLUME
                BCC       ROOT1                ;  USER SAID S/HE DID-- CHECK IT
                LDA       #VNFERR              ; REPORT VOLUME NOT FOUND ERR IF REFUSE TO INSERT
SRITZ           RTS
*
                PAGE
ROOT3           LDY       #$F                  ; (NOTE: X CONTAINS THE LENGTH OF THE ROOT NAME)
ROOTINFO        LDA       GBUF+HCRDT+3,Y       ; SAVE HEADER INFO.
```

```
                STA        V.STATUS,Y
                DEY
                BNE        ROOTINFO             ; LOOP TIL ALL 15 BYTES MOVED
                LDA        H.FCNT
                STA        ENTCNTL
                LDA        H.FCNT+1
                STA        ENTCNTH
                TXA                             ; NOW THAT ROOT IS IDENTIFIED, ADJUST
                SEC                             ;  PATH NAME POINTER TO NEXT NAME IN THE PATH
                ADC        PATHNML
                STA        PATHNML
                CLC                             ; INDICATE NO ERROR
ROOTERR         RTS
*
*
CHKROOT         LDY        #0                   ; GET LENGTH OF NAME
                LDA        (PATHNML),Y
                TAY
                TAX                             ; SAVE IN X FOR LATTER ADJUSTMENT TO PATH POINTER
                EOR        GBUF+4
                AND        #$F                  ; DOES PATHNAME HAVE SAME LENGTH AS DIRECTORY NAME?
                BNE        NOTROOT              ; BRANCH IF NOT
CKROOT1         LDA        (PATHNML),Y          ; COMPARE CHARACTER BY CHARACTER
                CMP        GBUF+4,Y
                BNE        NOTROOT
                DEY
                BNE        CKROOT1              ; LOOP UNTIL ALL CHARACTERS MATCH
NOTROOT         RTS
*
                PAGE
FINDVOL         LDA        #VCB/256             ; SEARCH VCB FOR VOLUME NAME
                STA        VCBPTR+1
                LDA        #0
                STA        D.DEV
                STA        VCBPTR
FNDVOL1         PHA                             ; SAVE LAST SEARCH POSITION
                TAX
                LDY        #0                   ; (INDEX TO PATHNAME POINTER)
                LDA        VCB,X                ; GET LENGTH OF VOLUME NAME TO COMPARE
                BEQ        NXTVCB               ; BRANCH IF VCB ENTRY IS EMPTY
                CMP        (PATHNML),Y          ; ARE NAMES OF SAME LENGTH?
                BNE        NXTVCB               ; NO, INDEX NEXT VCB
                CLC                             ; SCAN NAME BACKWARDS
                TAY
                TXA
                ADC        VCB,X
                TAX                             ; NOW BOTH INDEXES POINT TO LAST CHARACTER OF THE NAMES TO COMPARE
VOLNAM          LDA        (PATHNML),Y
                CMP        VCB,X
                BNE        NXTVCB
                DEX
                DEY
                BNE        VOLNAM               ; CHECK ALL CHARACTERS
                PLA                             ; SINCE A MATCH IS FOUND
                STA        VCBPTR               ;  SET UP INDEX TO VCB ENTRY
                TAX
                LDA        VCB+VCBSWAP,X        ;  BRANCH IF
                BEQ        FOUNDVOL             ;  VOLUME NOT SWAPPED
                JSR        SWAPIN               ;  IF USER REALLY WANTS IT, THEN BRING IN IF SWAPPED
                BCC        FOUNDVOL             ;  BRANCH IF SUCCESS
                LDA        #XIOERROR            ;  USER REFUSES TO MOUNT
                RTS
FOUNDVOL        CLC                             ; INDICATE VOLUME FOUND
                RTS
*
NXTVCB          PLA                             ; GET CURRENT INDEX AGAIN.
                CLC
                ADC        #VCBSIZE             ; VCB ENTRY LENGTH.
                BCC        FNDVOL1              ; BRANCH IF THER IS ANOTHER TO CHECK
                RTS                             ; RETURN WITH CARRY SET TO SHOW FAILURE.
                PAGE
*
*
LOOKVOL         LDX        #12                  ; (1) COUNT+(12)DEVICE LIST
LOOKVOL1        LDA        BLKDLST,X            ;  EXTRN
                STA        SCRTCH,X             ;  MY CHANGEABLE COPY
                DEX
                BPL        LOOKVOL1             ;  WORK BACKWARDS SO
                STA        TOTDEVS              ;  ENTRY ZERO IS TOTAL DEVICES LISTED
                INX                             ;  MAKE XREG = ZERO
LOKDEV1         INX
                STX        SCRTCH
                LDA        SCRTCH,X
                CMP        D.DEV
                BEQ        NXTDEV               ; DON'T LOOK AGAIN ON A DRIVE THAT HAS BEEN CHECKED
                STA        DEVNUM               ; CHECK FOR DEVICE ALREADY LOGGED IN A VCB
                JSR        DEVVCB               ; (CARRY CLEAR IF IT'S THERE)
                BCC        LOKVOL1
                LDA        #0                   ; FIND A FREE VCB TO LOG THIS GUY IN
ENTVCB          TAX                             ; INDEX TO NEXT VCB ENTRY
                LDA        VCB,X
                BEQ        FREEVCB              ; FOUND A FREE SPOT.
                TXA                             ; NOW INDEX TO NEXT, AND KEEP LOOKIN
                CLC
                ADC        #VCBSIZE             ; (EACH VCB ENTRY IS 32 BYTES)
                BCC        ENTVCB               ; BRANCH IF MORE TO FIND
                LDA        #0
ENTVCB2         EQU        *                    ; SEE IF WE CAN REPLACE A DEVICE
                TAX
                LDA        VCB+VCBSTAT,X        ; VCB HAS FILES OPEN?
                BEQ        FREEVCB              ; NO, USE IT!
                TXA
                CLC
                ADC        #VCBSIZE             ; SEARCH NEXT VCB ENTRY
                BCC        ENTVCB2
                RTS                             ; FAILED TO FIND A FREE VCB ENTRY
*
CHKVLOG         LDY        #0                   ; MAKE SURE VOLUME WAS ACTUALLY LOGGED IN
                LDA        (VCBPTR),Y
                BNE        FOUNDVOL             ; AH, MADE IT...
                LDA        #DUPVOL              ; WELL, NOT QUITE, THIS VOLUME CAN'T BE LOGGED
                SEC
                RTS
                PAGE
*
FREEVCB         STX        VCBPTR               ; NOW THIS IS THE POINTER TO A FREE VCB
                LDA        #2                   ; ROOT DIRECTORIES ALWAYS AT BLOCK 2
                LDX        #0
                BEQ        GETROOT              ; BRANCH ALWAYS
LOKVOL1         LDY        #VCBSTAT             ; MAKE SURE NO FILES ARE ACTIVE ON
                LDA        (VCBPTR),Y           ;  THE VOLUME BEFORE LOGGING IT IN.
                BMI        SNSWIT               ;  BRANCH IF FILES ACTIVE
                LDY        #VCBROOT+1           ; GET ADDRESS OF ROOT DIRECTORY
                LDA        (VCBPTR),Y           ; HIGH FIRST.
                TAX
                DEY                             ; THEN LOW.
                LDA        (VCBPTR),Y
GETROOT         JSR        GETROT0
                BCC        LOKVOL2              ; BRANCH IF SUCCESSFULLY READ.
                LDA        #0                   ; OTHERWISE, TAKE THIS DEVICE OUT OF VCB
                TAY
                STA        (VCBPTR),Y           ; (VOLUME 'OFF LINE')
                BEQ        NXTDEV               ; BRANCH ALWAYS
*
LOKVOL2         JSR        LOGVCB               ; GO UPDATE VCB TO INCLUDE CURRENT VOLUME INFO
                BCS        NXTDEV               ;  IF NOT A SOS DISKETTE, SKIP TO NEXT DEVICE
                JSR        CHKROOT              ; GO COMPARE TO SEE IF WE FOUND WHAT WE'RE
                BEQ        CHKVLOG              ;  LOOKING FOR...
*
NXTDEV          LDX        SCRTCH               ; LOOK AT OTHER DEVICES?
                CPX        TOTDEVS
                BCC        LOKDEV1              ; YES.
                LDA        #VNFERR              ; REPORT VOLUME NOT FOUND.
                RTS
*
SNSWIT          EQU        *                    ;  SENSE DSWITCH
                LDY        #VCBDEV
                LDA        (VCBPTR),Y
                STA        DEVNUM               ;  MAKE SURE DEVICE NUMBER IS CURRENT
                JSR        TWRPROT1             ;  USES DEVNUM
                LDA        DSWGLOB              ;  DISK SWITCH GLOBAL
                BEQ        NXTDEV               ;  BRANCH IF NO DISK SWITCH
                JSR        VERFYVOL             ;  COMPARES VCBPTR VS. DEVNUM CONTENTS
                BCC        NXTDEV               ;  BRANCH IF DISK HAS NOT BEEN SWITCHED
                JSR        CHKROOT              ;  COMPARES PATHNML VS. GBUF
                BNE        NXTDEV               ;  IGNORE IF NOT WHAT WE ARE LOOKING FOR
                LDX        #0                   ;  LOOK FOR FREE
                JSR        SNSWIT1
                BCS        NXTDEV               ;  ANY ERRORS LOGGING IN THE NEW VOLUME
                JMP        CHKVLOG              ;  MAKE SURE THE NEW VOLUME IS LOGGED
SNSWIT1         LDA        VCB,X                ;  VCB ENTRY
                BEQ        SNSWIT2              ;  BRANCH IF FOUND
                TXA
                CLC
                ADC        #VCBSIZE             ;  LOOK AT NEXT VCB AREA
                TAX
                BCC        SNSWIT1
                RTS                             ;  CAN'T BE LOGGED IN!
SNSWIT2         LDA        #0
                STA        DUPLFLAG             ;  TURN OFF DUPLICATE VOLUME FLAG
                STX        VCBPTR
                JSR        LOGVCB1              ;  PARTIALLY LOG IN THE NEW VOLUME
                BCS        NONSOS               ;  CS MEANS NONSOS ERROR
                LDA        DUPLFLAG             ;  WAS IT A DUPLICATE VOLUME?
                BNE        SNSWIT6              ;  BRANCH IF YES
                LDY        #VCBSWAP             ;  BY MAKING SWAP BYTE NON ZERO
                LDA        #1
                STA        (VCBPTR),Y           ;  SO SWAPOUT WON'T AFFECT
                LDA        DEVNUM               ;  A REG PASSES DEVNUM TO SWAPOUT
                JSR        SWAPOUT              ;  OLD ACTIVE MOUNT MUST BE SWAPPED
                BCC        SNSWIT3
                LDA        #XIOERROR            ;  USER REFUSED TO REPLACE OLD VOLUME
                RTS
SNSWIT3         LDY        #VCBSWAP             ;  NOW LOG IN THE NEW ALL THE WAY
                LDA        #0
                STA        (VCBPTR),Y
SNSWIT4         JSR        VERFYVOL             ;  DON'T BOTHER TO ASK IF NEW VOLUME IS ALREADY MOUNTED
                BCC        SNSWIT5              ;  BRANCH IF NEW VOLUME ON LINE
                JSR        USRREQ               ;  ASK USER TO REMOUNT NEW VOLUME
                BCC        SNSWIT4              ;  USER SAYS THEY DID: CHECK IT OUT
                LDA        #VNFERR
SNSWIT5         RTS
SNSWIT6         LDA        #DUPVOL
                SEC
                RTS
                PAGE
*
NONSOS          LDA        #NOTSOS              ; TELL EM IT'S NOT A SOS DISK (COULD BE PASCAL)
                RTS                             ;  CARRY SHOULD ALREADY BE SET
*
```

```
*
DEVVCB          LDA        #0                   ; SCAN VCB FOR DEVICE SPECIFIED IN 'DEVNUM'
DVCB1           TAX                             ; FIRST TEST FOR VALID VCB.
                LDA        VCB,X
                BEQ        DVCB2
                LDA        VCB+VCBSWAP,X        ;  SWAPPED VOLUMES DON'T COUNT
                BNE        DVCB2                ;  AS LOGGED IN
                LDA        VCB+VCBDEV,X         ; GET DEVICE NUMBER
                CMP        DEVNUM               ; TEST AGAINST REQUESTED DEVICE
                BEQ        FOUNDEV              ; YES, SET UP A POINTER TO IT
DVCB2           TXA                             ; BUMP TO NEXT VCB
                CLC
                ADC        #VCBSIZE
                BCC        DVCB1                ; BRANCH IF MORE TO LOOK AT.
                RTS                             ; RETURN CARRY SET TO INDICATE NOT FOUND
*
TSTDUPVOL       LDX        VCBPTR               ; PRESERVE CURRENT ADDR OF FREE VCB
                LDA        #0                   ; LOOK FOR A CURRENTLY LOGGED ON VOLUME OF THE SAME NAME.
TSDUPV1         STA        VCBPTR
                JSR        CMPVCB
                BCS        TSDUPV2              ; BRANCH IF NO MATCH.
                LDY        #VCBSTAT             ; TEST FOR ANY OPEN FILES.
                LDA        (VCBPTR),Y
                BMI        FOUNDDUP             ; TELL THE SUCKER HE CAN'T LOOK AT THIS VOLUME!
                LDA        #0                   ; TAKE DUPLICATE OFF LINE IF NO OPEN FILES.
                TAY
                STA        (VCBPTR),Y
                BEQ        NODUPVOL             ; RETURN THAT ALL IS OK TO LOG IN NEW.
TSDUPV2         LDA        VCBPTR
                CLC
                ADC        #VCBSIZE             ; BUMP TO NEXT ENTRY.
                BCC        TSDUPV1
NODUPVOL        EQU        *
FOUNDEV         CLC
FNDDUP1         STX        VCBPTR
                RTS
*
FOUNDDUP        STA        DUPLFLAG             ; A DUPLICATE HAS BEEN DETECTED.
                SEC                             ; INDICATE ERROR
                LDA        VCBPTR               ;  SAVE ADDRESS OF DUPLICATE
                STA        VCBENTRY
                BCS        FNDDUP1              ; BRANCH ALWAYS TAKEN
                PAGE
*
*
LOGVCB          LDY        #VCBNML              ; IS THIS A PREVIOUSLY LOGGED IN VOLUME
                LDA        (VCBPTR),Y           ; (ACC=0?)
                BEQ        LOGVCB1              ; NO, GO AHEAD AND PREPARE VCB.
                JSR        CMPVCB               ; DOES VCB MATCH VOLUME READ?
                BCC        VCBLOGD              ; YES, DON'T DISTURB IT.
```

```
LOGVCB1         LDA        #0                   ; ZERO OUT VCB ENTRY
                LDY        #VCBSIZE-1
ZERVCB          STA        (VCBPTR),Y
                DEY
                BPL        ZERVCB
                JSR        TSTSOS               ; MAKE SURE IT'S A SOS DISKETTE.
                BCS        VCBLOGD              ; IF NOT, RETURN CARRY SET.
                JSR        TSTDUPVOL            ; FIND OUT IF A DUPLICATE WITH OPEN FILES ALREADY EXISTS
                BCS        NOTLOG0
                LDA        GBUF+4               ; MOVE VOLUME NAME TO VCB
                AND        #$F                  ; STRIP ROOT MARKER
                TAY
                PHA
MOVOLNM         LDA        GBUF+4,Y
                STA        (VCBPTR),Y
                DEY
                BNE        MOVOLNM
                PLA                             ; GET LENGTH AGAIN
                STA        (VCBPTR),Y           ; SAVE THAT TOO.
                LDY        #VCBDEV              ; SAVE DEVICE NUMBER ALSO.
                LDA        DEVNUM
                STA        (VCBPTR),Y
                JSR        CLEARBMS             ;  MARKS THIS DEVICES OLD BITMAPS AS INVALID (A REG PASSED)
                LDA        GBUF+VTBLK+4         ; AND TOTOL NUMBER OF BLOCKS ON THIS UNIT,
                LDY        #VCBTBLK
                STA        (VCBPTR),Y
                LDA        GBUF+VTBLK+5
                INY
                STA        (VCBPTR),Y
                LDY        #VCBROOT
                LDA        BLOKNML              ; AND ADDRESS OF ROOT DIRECTORY
                STA        (VCBPTR),Y
                INY
                LDA        BLOKNMH
                STA        (VCBPTR),Y
                LDY        #VCBDMAP
                LDA        GBUF+VBMAP+4         ; AND LASTLY, THE ADDRESS
                STA        (VCBPTR),Y           ;  OF THE FIRST BITMAP
                LDA        GBUF+VBMAP+5
                INY
                STA        (VCBPTR),Y
                CLC                             ; INDICATE THAT IT WAS LOGGED IF POSIBLE.
VCBLOGD         RTS
NOTLOG0         JMP        NOTLOG1
                PAGE
CMPVCB          LDA        GBUF+4               ; COMPARE VOLUME NAME IN VCB
                AND        #$F
                LDY        #VCBNML              ;  WITH NAME IN DIRECTORY
                CMP        (VCBPTR),Y           ; ARE THEY SAME LENGTH
                BNE        NOTSAME
                TAY
VCBCMP1         LDA        GBUF+4,Y
                CMP        (VCBPTR),Y
                BNE        NOTSAME
                DEY
                BNE        VCBCMP1
                CLC                             ; INDICATE MATCH.
                RTS
*
VERFYVOL        LDX        #0                   ; READ IN ROOT DIRECTORY HEADER.
                LDA        #2
                JSR        GETROT0
                BCS        NOVRFY1              ; PASS BACK WHATEVER OTHER ERROR OCCURS.
                JSR        CMPVCB               ; TEST ROOT WITH VOLUME NAME IN VCB.
                BCC        NOVRFY               ;  BRANCH IF ROOT MATCHES VCB
                LDA        #0                   ;  OTHERWISE, PASS BACK FOREIGN VOLUME ERROR (SOS OR UCSD)
NOVRFY          RTS                             ; RETURN RESULTS IN CARRY.
NOVRFY1         LDA        #VNFERR              ;  NOTHING IN DRIVE
                RTS
*
GETROT0         STA        BLOKNML
                STX        BLOKNMH              ; STORE ADDRESS AND READ IN ROOT
                JSR        RDGBUF
                BCC        RETROT2              ; BRANCH IF SUCCESSFULLY READ.
NOTSAME         EQU        *
                SEC                             ; INDICATE ERROR
RETROT2         RTS
*
NOTLOG1         LDX        VCBPTR               ;  LOAD THE VCB ADDRESS
                LDA        VCBENTRY             ;  OF THE DUPLICATE VOLUME
                STA        VCBPTR
                STX        VCBENTRY             ;  AND SAVE THE FREE VCB SPACE ADDR
                LDY        #VCBDEV              ;  IS DUPLICATE ON SAME DEVICE?
                LDA        DEVNUM
                CMP        (VCBPTR),Y
                BNE        NOTLOG2              ;  BRANCH IF NOT
                JSR        SWAPIN               ;  SWAP IN IF NECESSARY
                LDA        #0
                STA        DUPLFLAG             ;  NO MORE DUPLICATE VOLUME STATUS
                LDA        VCBPTR               ;  MAKE CHKROOT WORK IN A MOMENT
                STA        PATHNML              ;  THIS IS INCREDIBLY GROSS
;  BUT IS A RESULT OF MAKING VOLUME A SPECIAL
;  CASE OF SEARCHING ALL DEVICES FOR
;  A KNOWN VOLUME
                CLC
                RTS
NOTLOG2         LDA        VCBENTRY             ;  REACH HERE IF REAL DUPLICATE VOLUME
                STA        VCBPTR               ;  RESOTRE FREE VCB PTR
                CLC
                RTS                             ;  DUPLICATE VOLUME PRETENDS TO BE NO ERROR
                PAGE
*
TSFRBLK         LDY        #VCBTFRE+1
                LDA        (VCBPTR),Y           ; FIND OUT IF ENOUGH FREE BLOCKS
                DEY                             ; ARE AVAILABLE TO ACCOMODATE REQEST.
                ORA        (VCBPTR),Y           ; BUT FIRST FIND OUT IF WE GOT A PROPER COUNT FOR THIS VOLUME.
                BNE        CMPFREB              ; BRANCH IF COUNT IS NON-ZERO
                DEY                             ; IF ZERO, THEN COUNT MUST BE TAKEN
                LDA        (VCBPTR),Y           ;  GET HIGH TOTAL BLKS
                TAX                             ;  SAVE IT
                DEY                             ;  GET LOW
                LDA        (VCBPTR),Y           ;  TOTAL BLKS
                BNE        TSFR01
                DEX                             ;  ADJUST FOR BITMAP BLOCK BOUNDARY
TSFR01          TXA
                LSR        A                    ; DIVIDE BY 16. THE RESULT IS THE NUMBER
                LSR        A                    ;  OF BIT MAPS TO BE SEARCHED.
                LSR        A
                LSR        A
                STA        BMCNT                ; SAVE IT.
                LDA        #0                   ; START COUNT AT ZERO.
                STA        SCRTCH
                STA        SCRTCH+1
                LDA        #$FF                 ; MARK 'FIRST FREE' TEMP AS UNKNOWN
                STA        NOFREE
                LDY        #VCBDEV              ; MAKE SURE BIT MAP IS UP TO DATE
                LDA        (VCBPTR),Y           ; GET DEVICE NUMBER
                TAX                             ; PASS TO 'UPBMAP' IN X
                JSR        UPBMAP               ; (NOTHING HAPPENS IF IT DON'T HAFTA.)
                BCS        TFBERR               ; BRANCH IF WE GOT TROUBLE,
                LDY        #VCBDMAP             ; GET ADDRESS OF FIRST BIT MAP.
                LDA        (VCBPTR),Y
                STA        BLOKNML
                INY                             ; (FOR HIGH ADDRESS)
                LDA        (VCBPTR),Y
                STA        BLOKNMH
BMAPRD          JSR        RDGBUF               ; USE G(ENERAL)BUFF(ER) FOR TEMPORARY
                BCS        TFBERR               ;  SPACE TO COUNT FREE BLOCKS (BITS)
                JSR        COUNT                ; GO COUNT EM
                DEC        BMCNT                ; WAS THAT THE LAST BIT MAP?
                BMI        CHGVCB               ; IF SO, GO CHANGE FCB TO AVOID DOING THIS AGAIN!
                INC        BLOKNML              ; NOTE: THE ORGANIZATION OF THE BIT MAPS
                BNE        BMAPRD               ;  ARE CONTIGUOUS FOR SOS VERSION 0
                INC        BLOKNMH              ;  IF SOME OTHER ORGANIZATION IS IMPLEMENTED, THIS CODE
                JMP        BMAPRD               ;  MUST BE CHANGED!
                PAGE
*
CHGVCB          LDY        #VCBCMAP             ; MARK WHICH BLOCK HAD FIRST FREE SPACE
                LDA        NOFREE
                BMI        DSKFULL              ; BRANCH IF NO FREE SPACE WAS FOUND.
                STA        (VCBPTR),Y
                LDY        #VCBTFRE+1          ; UPDATE THE FREE COUNT.
                LDA        SCRTCH+1            ; GET HIGH COUNT BYTE
                STA        (VCBPTR),Y          ; UPDATE VOLUME CONTROL BLOCK.
                DEY
                LDA        SCRTCH
                STA        (VCBPTR),Y          ; AND LOW BYTE TOO...
CMPFREB         LDA        (VCBPTR),Y          ; COMPARE TOTAL AVAILABLE
                SEC
                SBC        REQL                ;  FREE BLOCKS ON THIS VOLUME.
                INY
                LDA        (VCBPTR),Y
                SBC        REQH
                BCC        DSKFULL
                CLC
                RTS
DSKFULL         LDA        #OVRERR
                SEC
TFBERR          RTS
                PAGE
*
COUNT           LDY        #0                  ; BEGIN AT THE BEGINNING.
FRCONT          LDA        GBUF,Y              ; GET BIT PATTERN
                BEQ        FRCNT1              ; DON'T BOTHER COUNTING NOTHIN'
                JSR        CNTFREE
FRCNT1          LDA        GBUF+$100,Y         ; DO BOTH PAGES WITH SAME LOOP
                BEQ        FRCNT2
                JSR        CNTFREE
FRCNT2          INY
                BNE        FRCONT              ; LOOP TILL ALL 512 BYTES COUNTED
                BIT        NOFREE              ; HAS FIRST BLOCK WITH FREE SPACE BEEN FOUND YET?
                BPL        FRCNT3              ; BRANCH IF IT HAS.
                LDA        SCRTCH              ; TEST TO SEE IF ANY BLOCKS WERE COUNTED
                ORA        SCRTCH+1
                BEQ        FRCNT3              ; BRANCH IF NONE COUNTED.
                LDY        #VCBTBLK+1
                LDA        (VCBPTR),Y          ; SHOW THIS MAP IS FIRST WITH FREE SPACE
                SEC                            ;  CORRECT FOR EXACT MULTIPLES OF $1000
                SBC        #$01
                LSR        A
                LSR        A
                LSR        A
                LSR        A
                SEC                            ; SUBTRACT COUNTDOWN FROM TOTAL BIT MAPS
                SBC        BMCNT
                STA        NOFREE
FRCNT3          RTS
*
CNTFREE         ASL        A                   ; COUNT THE NUMBER OF BITS IN THIS BYTE.
                BCC        CFREE1
                INC        SCRTCH
                BNE        CFREE1
                INC        SCRTCH+1
CFREE1          TAX
                BNE        CNTFREE              ; LOOP UNTIL ALL BITS COUNTED.
                RTS

                CHN        ALLOC,4,1

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: FNDFIL
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.3of5.THREE:SOS.PATH.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: PATH
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
*
*
*
BFMGR           LDX        COMMAND             ; WHAT CALL?
*
*
*
                LDA        DISPTCH,X           ; TRANSLATE INTO COMMAND ADDRESS
                ASL        A                   ; (BIT 7 INDICATES IT'S GOT A PATHNAME TO PREPROCESS)
                STA        CMDTEMP
                AND        #$3F                ; (BIT 6 IS REFNUM PREPROCESS, 5 IS FOR TIME, SO STRIP EM.)
                TAX
                LDA        CMDTABLE,X          ; MOVE ADDRESS FOR INDIRECT JUMP.
                STA        CMDADR
                LDA        CMDTABLE+1,X        ; (HIGH BYTE)
                STA        CMDADR+1
                LDA        #<VCB
                STA        VCBPTR+1            ; INSURE DEFAULT HI ADDRESS TO VCB BEFORE CALLS
                LDA        #BKBITVAL           ; INIT "BACKUP BIT FLAG"
                STA        BKBITFLG            ; TO SAY "FILE MODIFIED"
                LDY        #MAXTEMPS           ; ZERO OUT SISTER PAGE FOR TEMPS
                LDA        #0
                STA        SERR                ; MAKE GLOBAL ERROR SAY "NONE"
                STA        DSWGLOB             ; "DISK SWITCH GLOBAL"
                STA        DUPLFLAG            ; "DUPLICATE VOLUME ON LINE"
                STA        CFLAG               ; SET "CREATE" TO NO
                STA        BLOKSAVE
                STA        BLOKSAVE+1          ; SET PARENT DIRECTORY TO NULL
CLRSIS          STA        SISTEMPS,Y
                DEY
                BPL        CLRSIS              ; CARRY IS UNDISTURBED BY THIS LOOP
                BCC        NOPATH
                JSR        SETPATH             ; GO PROCESS PATHNAME BEFORE CALLING COMMAND
                BCS        ERRORSYS            ; BRANCH IF BAD NAME.
NOPATH          ASL        CMDTEMP             ; TEST FOR REFNUM PREPROCESSING
                BCC        NOPREREF
                JSR        FINDFCB             ; GO SET UP POINTERS TO FCB AND VCB OF THIS FILE.
                BCS        ERRORSYS            ; BRANCH IF ANY ERRORS ARE ENCOUNTERED.
NOPREREF        ASL        CMDTEMP             ; LASTLY CHECK FOR NECESSITY OF TIME STAMP.
```

```
                BCC        TSWVRFY
                LDX        #DATELO              ; PASS Z PAGE ADDRESS OF WHERE TO RETURN DATE/TIME
                JSR        DATETIME             ; (NO ERROR POSIBLE)
TSWVRFY         LDX        COMMAND              ; TEST FOR NECESSITY OF VOLUME VERIFICATION
                LDA        #PREPATH+PREREF+PRETIME ; TO ENSURE VCB IS SET
                AND        DISPTCH,X
                BEQ        EXECUTE
                LDY        #VCBSTAT
                LDA        (VCBPTR),Y
                AND        #DSWITCH             ; WAS THE VOLUME PREVIOUSLY SWITCHED?
                BEQ        EXECUTE
                DEY                             ; GET DEVICE NUMBER
                LDA        (VCBPTR),Y
                STA        DEVNUM
DVERIFY         JSR        VERFYVOL             ; SEE IF PROPER VOLUME NOW ON LINE
                BCC        CLRDSWT              ; BRANCH IF YES
                JSR        USRREQ               ; OTHERWISE REQUEST IT BE PUT ON LINE
                BCC        DVERIFY              ; USER SEZ S/HE DID: CHECK IT OUT
                LDA        #VNFERR              ; VOLUME NOT FOUND IF USER REFUSES
                BNE        ERRORSYS             ; REPORT ERROR (BRANCH ALWAYS)
CLRDSWT         LDY        #VCBSTAT             ; GET VOLUME
                LDA        (VCBPTR),Y           ; STATUS
                AND        #$FF-DSWITCH         ; TURN OFF DISK SWITCH
                STA        (VCBPTR),Y           ; SO WE WON'T VERIFY NEXT TIME
EXECUTE         JSR        GOCMD                ; EXECUTE COMMAND
                BCC        GOODOP               ; BRANCH IF SUCCESSFUL
                CMP        #XDISKSW             ; DISK SWITCH?
                BNE        ERRORSYS             ; NO, REPORT SOME OTHER
                LDY        #VCBSTAT             ; MARK VCB WITH SWITCH
                LDA        (VCBPTR),Y
                AND        #$FF-DSWITCH         ; TO ENSURE VOLUME VERIFIED
                BPL        ERRCMD               ; NO FILES OPEN SO DSWITCH CANT APPLY
                ORA        #DSWITCH
ERRCMD          STA        (VCBPTR),Y
                JMP        BFMGR                ; TRY THE COMMAND AGAIN
*
ERRORSYS        JSR        SYSERR
GOODOP          RTS                             ; GOOD RETURN
*
GOCMD           JMP        (CMDADR)
*
                PAGE
*
CMDTABLE        EQU        *
                DW         CREATE
                DW         DESTROY
                DW         RENAME
                DW         SETINFO
                DW         GETINFO
                DW         VOLUME
                DW         SETPREFX
                DW         GETPREFX
                DW         OPEN
                DW         NEWLINE
                DW         READ
                DW         WRITE
                DW         CLOSE
                DW         FLUSH
                DW         SETMARK
                DW         GETMARK
                DW         SETEOF
                DW         GETEOF
*
DISPTCH         EQU        *
                DFB        PREPATH+PRETIME+0     ; CREATE
                DFB        PREPATH+PRETIME+1     ; DESTROY
                DFB        PREPATH+PRETIME+2     ; RENAME
                DFB        PREPATH+PRETIME+3     ; SETINFO
                DFB        PREPATH+4             ; GETINFO
                DFB        5                     ; VOLUME
                DFB        6                     ; SETPREFIX, PATHNAME MOVED TO PREFIX BUFFER
                DFB        7                     ; GETPREFIX
                DFB        PREPATH+8             ; OPEN
                DFB        PREREF+$9             ; NEWLINE
                DFB        PREREF+$A             ; READ
                DFB        PREREF+$B             ; WRITE
                DFB        PRETIME+$C            ; CLOSE
                DFB        PRETIME+$D            ; FLUSH, REFNUM MAY BE ZERO TO FLUSH ALL.
                DFB        PREREF+$E             ; SETMARK
                DFB        PREREF+$F             ; GETMARK
                DFB        PREREF+$10            ; SET EOF
                DFB        PREREF+$11            ; GET EOF
*
                PAGE
*
SETPATH         LDA        C.PATH                ; FOR A REGULAR PATHNAME,
                STA        TPATH                 ; SET UP TEMP POINTER TO PROCESS
                LDA        C.PATH+1              ; PATHNAME AND CHECK FOR SYNTAX ERRORS
                STA        TPATH+1
                LDA        SISPATH
                STA        SISTPATH              ; (LEAVE CALL PARAMETERS ALONE!)
* DROP INTO 'SYNPATH'
*
SYNPATH         LDA        #>PATHBUF             ; SET UP DEFAULT ADDRESS FOR
                STA        PATHNML               ; SYNTAXED PATHNAME -
                STA        WRKPATH               ; LENGTH, NAME, LENGTH, NAME, ETC...
                LDA        #<PATHBUF
                STA        PATHNMH
                STA        WRKPATH+1             ; (ASSUMES FULL PATHNAME, NO PREFIX).
                LDX        #0                    ; USE INDEXED INDIRECT FOR SOURCE PATHNAME
                TXA                                   ; SET BEGINNING OF PATH
                STA        (PATHNML,X)                ; TO ZERO TO INDICATE NOTHING PROCESSED.
                TAY
                LDA        (TPATH,X)                  ; GET TOTAL LENGTH OF SOURCE PATHNAME
                BMI        ERRSYN
                BEQ        ERRSYN
                STA        PATHCNT                    ; (THIS IS USED AS A 'COUNT-DOWN')
                JSR        INCTPTH                    ; INCREMENT SOURCE POINTER
                LDA        (TPATH,X)                  ; GET FIRST CHARACTER OF PATHNAME
                CMP        #DLIMIT                    ; IS IT A FULL PATHNAME (NO PREFIX)?
                BEQ        BUMPATH                    ; YES, WE'RE READY TO DO IT.
                CMP        #$2E                       ; IS IT A DRIVE NAME '.'?
                BNE        ADPREFIX                   ; NO, ADD PREFIX TO BEGINNING
DRIVENAM        LDA        (TPATH,X)                  ; MOVE DRIVE NAME FOR VOLUME CALL
                CMP        #DLIMIT                    ; HAVE WE MOVED ENTIRE NAME?
                BEQ        PREVOLM                    ; YES, PROCESS IT.
                INY                                   ; (IF THIS IS THE FIRST, MAKE ROOM FOR LENGTH OF NAME)
                STA        (WRKPATH),Y
                JSR        INCTPTH                    ; BUMP POINTER TO GIVEN NAME.
                DEC        PATHCNT
                BNE        DRIVENAM
                BEQ        PREVOLM1
*
                PAGE
PREVOLM         JSR        INCTPTH                    ; MAKE IT SO POINTING PAST DELIMITER.
                DEC        PATHCNT
PREVOLM1        TYA                                   ; SAVE LENGTH OF DRIVE NAME.
                STA        (WRKPATH,X)
                LDA        #>PATHBUF                  ; POINT AT PATHNAME BUFFER FOR DEVICE ID CALL.
                STA        DVNAMP
                LDA        #<PATHBUF
                STA        DVNAMP+1
                LDA        #0                         ; MAKE VIRTUAL POINT AT SWITCHED IN BANK.
                STA        SISTER+DVNAMP+1
                JSR        SRCHDEV                    ; GO IDENTIFY WHICH VOLUME
                BCC        PREVOLM2                   ; BRANCH IF NO ERROR
                CMP        #VNFERR                    ; WAS IT REPORTED AS 'VOLUME NOT FOUND'?
                BNE        SPTHERR                    ; NO SOME OTHER ERROR WAS ENCOUNTERED.
                LDX        DUPLFLAG                   ; YES, WAS IT NOT FOUND BECAUSE SOME OTHER 'OPEN' VOLUME HAS SAME NAME?
                BEQ        SPTHERR                    ; NO, IT SIMPLY WASN'T FOUND.
                LDA        #DUPVOL                    ; (CARRY IS SET)
                RTS
*
PREVOLM2        LDY        #0                         ; (X CONTAINS AN INDEX TO VCB)
                LDA        VCB,X                      ; GET VOLUME NAME LENGTH.
                STA        PATHBUF,Y
SPATH2          INX                                   ; MOVE VOLUME NAME INTO PATH NAME BUFFER IN
                INY                                   ; PLACE OF DISK DEVICE NAME ('.D1' OR SIMULAR)
                LDA        VCB,X
                STA        PATHBUF,Y
                CPY        PATHBUF              ; HAVE ALL CHARACTERS BEEN MOVED?
                BNE        SPATH2
                LDX        #0                   ; RESET X FOR INDEXING
                STX        PATHNML
                LDA        #<PATHBUF
                STA        PATHNMH
                LDA        PATHCNT              ; IS THAT ALL THERE IS?
                BNE        SPATH3               ; NO, MORE TO COME...
                CLC
                JMP        ENDPATH
*
SPATH3          INY                             ; BUMP TO END OF NAME+1
                STY        WRKPATH              ; RESET WORKPATH POINTER TO CURRENT.
                LDA        #0                   ; RESET PATHNAME BUFFER POINTER.
                LDY        #<PATHBUF
                BNE        NOPREFX              ; BRANCH ALWAYS...
*
ERRSYN          LDA        #BADPATH             ; RETURN SYNTAX ERROR
SPTHERR         SEC
                RTS
*
ADPREFIX        LDA        PFIXPTR              ; GET POINTER TO BEGINNING OF THE
                LDY        PFIXPTR+1            ; PREFIX.
NOPREFX         STA        PATHNML
                STY        PATHNMH              ; IF NO PRESET PREFIX, THIS IS THE SAME AS
                BNE        FRSTCHAR             ; PATHBUF ADDRESS. (BRANCH ALWAYS TAKEN)
*
                PAGE
*
BUMPATH         DEC        PATHCNT              ; FIRST ADJUST COUNT
                CLC                             ; (JUST IN CASE OF LAST CHARACTER)
                BEQ        ENDPATH              ; (MUST OF HAD TRAILING SPACES)
                JSR        INCTPTH
FRSTCHAR        LDY        #0                   ; INIT COUNT FOR THIS PORTION OF THE
                TYA                             ; PATHNAME. ALSO PRESET LENGTH TO ZERO IN
                STA        (WRKPATH,X)          ; CASE OF TRAILING SPACES.
                LDA        (TPATH,X)            ; GET CHARACTER.
                AND        #$7F                 ; IGNORE HIGH BIT.
                CMP        #$20                 ; IS IT A LEADING SPACE?
                BEQ        BUMPATH              ; IF SO, IGNORE IT.
                CMP        #$5B                 ; IS IT GREATER THAN (UPPER CASE) A 'Z'?
                BCC        ALFA1                ; NO, MAKE SURE IT'S AN ALPHA CHARACTER
                AND        #$5F                 ; YES, ASSUME IT'S LOWER CASE, AND UPSHIFT
                CMP        #$5B                 ; WAS IT TRULY LOWER CASE?
                BCS        ERRSYN               ; NO, GIVE ERROR.
*
ALFA1           CMP        #$41                 ; IS IT LESS THAN 'A'?
                BCC        ERRSYN               ; YES! IT'S CRAP...
                BCS        SAVPATH              ; NO, IT'S GOOD. SAVE IT.
*
```

```
NXTCHAR         LDA        (TPATH,X)            ; GET THE NEXT CHARACTER.
                AND        #$7F                 ; THESE CHARACTERS MAY BE ALPHA, NUMERIC,
                CMP        #$5B                 ; OR A PERIOD - ONLY THE FIRST HAD TO BE ALPHA
                BCC        ALFA2                ; BRANCH IF LESS THAN 'Z'
                AND        #$5F                 ; UPSHIFT LOWER CASE.
                CMP        #$5B                 ; NOW IS IT VALID?
                BCS        ERRSYN               ; NOPE.
*
ALFA2           CMP        #$41                 ; IS IT GREATER THAN 'A'?
                BCS        SAVPATH              ; YUP, IT IS WORTH SAVIN.
                CMP        #$3A                 ; >9?
                BCS        TSTDLIM              ; YES
                CMP        #$30                 ; NO, <0?
                BCS        SAVPATH              ; NO, IT'S VALID NUMERIC.
TSTDLIM         CMP        #DLIMIT              ; IS IT THE DELIMITER?
                BEQ        ENDPATH              ; YES. CARRY SET INDICATES MORE TO COME.
                CMP        #$2E                 ; IS IT A '.' (PERIOD)?
                BNE        ERRSYN               ; NO, IT'S AN ERROR (#@&##@!)
SAVPATH         CLC
                INY                             ; BUMP NAME LENGTH
                STA        (WRKPATH),Y
                DEC        PATHCNT              ; IF ZERO, THAT WAS THE LAST CHARACTER
                BEQ        ENDPATH              ; (CARRY CLEAR INDICATES END OF PATH)
                INC        TPATH                ; BUMP POINTER TO SOURCE PATHNAME.
                BNE        NXTCHAR
                INC        TPATH+1              ; HIGH ORDER, WHEN NECESSARY.
                BNE        NXTCHAR              ; BRANCH ALWAYS.
                PAGE
*
ENDPATH         TYA                             ; GET CURRENT NAME LENGTH
                STA        (WRKPATH,X)          ; AND PUT IT IN FRONT OF NAME
                BCC        LSTNAME              ; BRANCH IF THAT WAS THE LAST OF PATH
                CMP        #$10                 ; WAS THE NAME ILLEGALLY LONG?
                BCS        ERRSYN1              ; YES, REPORT IT.
                LDY        #0
                SEC                             ; ADJUST WORK POINTER TO END OF PREVIOUS NAME.
                ADC        WRKPATH
                STA        WRKPATH              ; REPLACE OLD POINTER.
                BCC        BUMPATH              ; DO NEXT NAME.
                LDA        #TOOLONG             ; THIS IS A NEVER ERROR!
                JSR        SYSDEATH             ; (NEVER RETURNS).
*
LSTNAME         BEQ        TSTVALD
                CMP        #$10                 ; MAKE SURE LAST ISN'T TOO LONG
                BCS        ERRSYN1
                INY                             ; PUT A ZERO AT END OF PROCESSED PATHNAME
                LDA        #0
TSTVALD         STA        (WRKPATH),Y
                LDA        (PATHNML,X)          ; SURE THERE IS A PATHNAME
                BEQ        ERRSYN1              ; IF NOT, REPORT ERROR.
                CLC                             ; INDICATE NO ERROR.
                RTS
*
ERRSYN1         JMP        ERRSYN
*
INCTPTH         INC        TPATH                ; POINT AT NEXT CHARACTER
                BNE        INCPTH1
                INC        TPATH+1
INCPTH1         RTS
*
                PAGE
SETPREFX        JSR        SETPATH              ; CALL IS MADE HERE SO A 'NUL' PATH MAY BE DETECTED.
                BCC        SETPRFX1             ; BRANCH IF PATHNAME OK
                TAX                             ; SAVE ERROR CODE
                LDY        #0
                LDA        (C.PATH),Y           ; TEST FOR A NUL PATHNAME
                BEQ        RESETPFX             ; BRANCH IF PREFIX TO BE RESET.
                TXA                             ; RESTORE ERROR CODE
                RTS
RESETPFX        STA        PFIXPTR
                CLC
                RTS
SETPRFX1        LDA        PATHNML              ; MAKE SURE NAME STARTED WITH A '/' DELIMITER.
                BNE        ERRSYN1              ; BRANCH IF IT DID.
                LDY        WRKPATH              ; FIND THE END OF THE INPUT PREFIX
                CLC                             ; ADD LAST LOCAL NAME LENGTH TO FIND TRUE END.
                LDA        (PATHNML),Y
                BNE        SETPRFX3
                DEY
                TYA
                BNE        SETPRFX4
SETPRFX3        ADC        WRKPATH
                TAY
SETPRFX4        EOR        #$FF                 ; GET COMPLIMENT TO FIND BEGINNING ADDRESS.
                STA        PFIXPTR              ; OF NEW PREFIX IN THE PREFIX BUFFER
                STA        WRKPATH              ; (PREFIX ALWAYS ENDS AT THE LAST BYTE OF BUFFER)
MOVPRFX         LDA        (PATHNML),Y
                STA        (WRKPATH),Y          ; MOVE IN NEW PREFIX
                DEY
                BPL        MOVPRFX
                CLC                             ; AND WE'RE FINISHED!
                RTS                             ; NO ERRORS POSIBLE FROM THIS ROUTINE.
*
                PAGE
*
GETPREFX        CLC                             ; CALCULATE HOW BIG A BUFFER IS NEEDED TO
                LDA        PFIXPTR              ; PASS THE PREFIX BACK TO THE USER.
                EOR        #$FF                 ; (EVEN IF NO PREFIX, 1 BYTE IS NEEDED TO SHOW 0 LENGTH)
                ADC        #2                   ; ADD 2 FOR LEADING AND ENDING "/".
                CMP        C.MAXPTH             ; IS THERE ENOUGH SPACE IN USER'S BUFFER?
                BCC        SENDPRFX             ; BRANCH IF YES
                LDA        #BTSERR              ; TELL USER BUFFER IS TOO SMALL.
                RTS                             ; (CARRY IS SET TO INDICATE ERROR.)
*
SENDPRFX        LDY        #0                   ; SAVE TOTAL LENGTH OF STRING TO BE RETURNED
                STA        (C.PATH),Y
                TAY
                DEY                             ; DISCOUNT TRAILING DELIMITER.
                BEQ        NULPREFX             ; BRANCH IF PREFIX IS SET TO NUL.
                INY
                LDX        PFIXPTR              ; GET BEGINNING ADDRESS OF PREFIX AGAIN
                DEX
                STX        WRKPATH
                LDA        #<PATHBUF
                STA        WRKPATH+1
SNDLMIT         LDA        #DLIMIT              ; PLACE DELIMITER BEFORE, BETWEEN, AND AFTER LOCAL NAMES.
                STA        (C.PATH),Y
SNDPRFX1        DEY
                BEQ        GOTPRFX              ; BRANCH IF ALL OF PREFIX IS TRANSFERED.
                LDA        (WRKPATH),Y
                STA        (C.PATH),Y           ; ASSUME IT'S A CHARACTER.
                AND        #$F0                 ; NOW TEST TO SEE IF IT WAS A LOCAL LENGTH.
                BEQ        SNDLMIT              ; BRANCH IF IT WAS.
                BNE        SNDPRFX1             ; GO MOVE NEXT CHAR IF IT WASN'T (ALWAYS TAKEN).
NULPREFX        TYA                             ; RETURN NUL STRING.
                STA        (C.PATH),Y
GOTPRFX         CLC                             ; INDICATE NO ERROR.
                RTS
                PAGE
*
FINDFCB         LDA        FCBADDRH             ; INITIALIZE INDIRECT POINTER TO
                STA        FCBPTR+1             ; FILE CONTROL BLOCK (ALLOCATED WHEN SYSTEM
                LDA        #0                   ; WAS FIRST BOOTED).
                STA        FCBPTR               ; NOTE: ALWAYS STARTS ON PAGE BOUNDARY.
                LDA        FCBANKNM             ; SET SISTE PAGE BYTE TOO...
                STA        SISFCBP
                LDY        C.REFNUM             ; GET REQUESTED REFERENCE
                BMI        ERRNOTBLK            ; BRANCH IF IT'S NOT A BLOCK DEVICE REFERENCE
                DEY                             ; (SHOULD BE IN THE RANGE OF 1-16 BEFORE DECREMENT)
                CPY        #$10                 ; IS IT A VALID REFNUM?
                BCS        REEFER               ; NO, THE USER'S SMOKIN DOPE!
                TYA                             ; TO FIND ASSOCIATED FILE CONTROL STUFF,
                ASL        A                    ; MULTIPLY (REFNUM-1) BY 32.
                ASL        A
                ASL        A
                ASL        A
                ASL        A
                BCC        SVFCBLO              ; BRANCH IF IT'S WITHIN FIRST HALF OF FCB
                INC        FCBPTR+1             ; BUMP TO SECOND HAVE (REFNUM>8)
SVFCBLO         STA        FCBPTR               ; SAVE LOW ADDRESS OF REFERENCED FCB
```

```
                LDA       C.REFNUM            ; NOW VERIFY THAT FILE IS OPEN.
                LDY       #FCBREFN
                CMP       (FCBPTR),Y          ; SHOULD BE EQUAL!
                BNE       ERRNOREF            ; BRANCH IF THEY'RE NOT
FNDFCBUF        LDY       #FCBBUFN            ; IT'S A LEGAL FILE, NOW SET UP
                LDA       (FCBPTR),Y          ; INDIRECT POINTERS TO DATA
GTBUFFRS        LDX       #DATPTR             ; (AND INDEX) BUFFER(S) IN ZERO PAGE
                JSR       GETBUFADR           ; GET BUFFER ADDRESS UNLESS
                BCS       REEFER1             ; BOB HAS BEEN SMOKIN DOPE...
                LDA       #2                  ; (ASSUME AN INDEX BLOCK BUFFER IS ALSO PRESENT)
                ADC       DATPTR+1
                STA       TINDX+1
                LDA       DATPTR
                STA       TINDX
                LDA       SISDATP
                STA       SSTIDXH
                LDY       #FCBDEVN
                LDA       (FCBPTR),Y          ; MAKE SURE DEVICE
                STA       D.DEV               ; NUMBER TEMPS MATCH
                STA       DEVNUM              ; CURRENT FILE'S DEVICE
                LDA       #0                  ; LOOK AT ALL VOLUMES LOGGED IN
FNDFVOL         TAX
                LDA       VCB+VCBDEV,X        ; GET VOLUMES DEVICE NUMBER
                CMP       (FCBPTR),Y          ; HVE WE FOUND A MATCH.
                BNE       FNDFV1
                LDY       #FCBSWAP            ; SWAP BYTES
                LDA       VCB+VCBSWAP,X       ; MISMATCH
                CMP       (FCBPTR),Y          ; MEANS FILE BELONGS
                BNE       FNDFV.1             ; TO ANOTHER VOLUME
                LDA       VCB,X               ; IS THIS AN OPEN DEVICE?
                BEQ       FNDFV.1             ; NO, TRY ANOTHER VOLUME
                JSR       FVOLFOUND           ; YES, SAVE VCB ADDRESS
                LDA       VCB+VCBSWAP,X       ; SWAPPED?
                BEQ       REEFER1             ; NO, RETURN CALMLY TO USER
                JSR       SWAPIN              ; YES, SWAP ME IN
                BCC       REEFER1             ; RETURN WITHOUT ERROR
                LDA       #XIOERROR           ; USER REFUSED TO MOUNT PROPER VOLUME
                RTS
*
FNDFV.1         LDY       #FCBDEVN            ; RELOAD Y WITH DEVICE INDEX
FNDFV1          TXA
                CLC
                ADC       #VCBSIZE
                BCC       FNDFVOL             ; LOOP UNTIL FOUND
                LDA       #VCBERR             ; OTHERWISE DIE A SYSTEM DEATH!
                JSR       SYSDEATH
                PAGE
*
ERRNOREF        LDA       #0                  ; DROP A ZERO INTO THIS FCB TO
                STA       (FCBPTR),Y          ; SHOW FREE FCB
```

```
*
REEFER          LDA        #BADREFNUM           ; TELL USER THAT REQUESTED REFNUM
                SEC                             ; IS ILLEGAL (OUT OF RANGE) FOR THIS CALL.
REEFER1         RTS
*
ERRNOTBLK       LDA        #NOTBLKDEV           ; TELL USER THAT SPECIFIED DEVICE IS NOT A BLOCK DEVICE
                SEC
                RTS
*
SVCBADR         EQU        *
FVOLFOUND       STX        VCBPTR
                LDA        #VCB/256
                STA        VCBPTR+1
                CLC                             ; INDICATE LEGAL REFNUM
                RTS
                PAGE
* NAME    : GETDNUM
* FUNCTION: GET DEVICE NUMBER
* INPUT   : DVNAMP SETUP
* OUTPUT  : DEVNUM IN 'SCRTCH'
*         : 'BPL' IF NOT BLOCK DEV
*         : 'BCS' IF NO DEVICE
* VOLATILE: ALL REGS
*
GETDNUM         EQU        *
                LDA        #>SCRTCH+1           ; SET UP POINTER TO SCRATCH AREA
                STA        DVDNUM               ; TO RECIEVE DEVICE NUMBER.
                LDA        #SCRHIGH
                STA        DVDNUM+1
                LDA        #0                   ; PLACE A ZERO IN BANK BYTE SINCE
                STA        SISTER+DVDNUM+1      ; IT'S NOT IN A BANK.
                STA        VCBPTR+1
                LDA        #4                   ; THE 'GET.DNUM' COMMAND.
                STA        DHPCMD
                JSR        RPEATIO0             ; CALL BOB FOR THE INFO.
                RTS                             ; RETURN WITH DEVMGR CC'S
                PAGE
*
* NAME    : SRCHDEV
* FUNCTION: SEARCH FOR A VOLUME
*
SRCHDEV         EQU        *
                JSR        GETDNUM              ; GET DEVNUM
                BCS        VOLERR1              ; BRANCH IF ANY ERROR OTHER THAN NOTBLOCKDEV
                BPL        ERRNOTBLK            ; BRANCH IF NOT A BLOCK DEVICE
                LDA        #0                   ; NOW SEARCH FOR A VOL WITH THE
                STA        NFOPEN               ; INIT TEMP VCB POINTER
VOLOOK          TAX                             ; SAME DEVNUM AS SCRTCH
                LDA        VCB+VCBSTAT,X        ; ANY FILES OPEN?
                BNE        VLOOK00              ; BRANCH IF SOME FILE OPEN
                STX        NFOPEN               ; ELSE SAVE THE VCB ENTRY PTR
VLOOK00         EQU        *
                LDA        VCB+VCBSWAP,X        ; VOLUME SWAPPED OUT?
                BNE        VNOTEQ               ; YES, CANT BE THE ACTIVE VOL
                LDA        VCB+VCBDEV,X
                EOR        SCRTCH+1
                BEQ        VLOOK0               ; BRANCH IF MATCH.
VNOTEQ          LDA        VCB,X                ; IS THIS A FREE VCB?
                BNE        VLOOK2               ; BRANCH IF NOT FREE, OTHEWISE TAKE NEXT BRANCH.
VLOOK0          EOR        VCB,X                ; TEST FOR A VOLUME NAME LENGTH
                BEQ        VLOOK1               ; BRANCH IF VCB FREE
                JSR        SVCBADR              ; SAVE CURRENT ADDRESS OF VCB.
                LDA        VCB+VCBSTAT,X        ; TEST FOR ANY OPEN FILES.
                BPL        VLOOK3               ; LOG THE VOLUME IN JUST TO BE SURE
                LDA        SCRTCH+1             ; SET UP
                STA        DEVNUM               ; DEVICE NUMBER ARGUMENT
                TXA                             ; SAVE PTR TO VCB
                PHA                             ; ON STACK
                JSR        VERFYVOL             ; COMPARES VCBPTR TO DEVNUM CONTENTS
                BCC        VNOSWIT
                CMP        #VNFERR              ; SEE IF NOTHING IN DRIVE
                BEQ        VLOOK7               ; BRANCH IF NOTHING IN DRIVE
                JSR        TSTSOS               ; IS THE VOLUME AN UNRECOGNIZED SOS OR (UCSD OR DOS)?
                BCS        KNOTSOS              ; DEFINITELY NOT SOS FORMAT
                LDX        #0                   ; START VCB SCAN AT BEGINNING
                JSR        SNSWIT1              ; FIND A FREE VCB AND LOG IN THE NEW GUY
                BCS        VNOSWIT1             ; CAN'T LOG IN NEW GUY--KEEP OLD
                PLA
                LDX        VCBPTR               ; PASS BACK X AS NEW VCB
                RTS
*
NFOPEN          DS         1                    ; TEMP VCB PTR FOR VCB W/ NO FILES OPEN
*
VNOSWIT         CLC                             ; RETURN IT TO USER
                PLA                             ; REMEMBER OLD VCB PTR
                TAX                             ; AND PASS BACK TO USER
                RTS
; RETURN TO CALLER X=POINTER TO VCB.
*
VOLERR1         SEC                             ; RETURN SOME VOLUME ERROR
                RTS
VNOSWIT1        CMP        #DUPVOL
                BNE        VLOOK7               ; REPORT OTHER ERROR FROM LOGGING IN NEW VOL AS VNF
                TAX
                PLA                             ; MAKE STACK CORRECT
                TXA                             ; RESTORE ERROR CODE
                SEC
                RTS                             ; IF DUPLICATE VOLUME ERROR, RETURN FACT TO USER
KNOTSOS         PLA                             ; MAKE STACK CORRECT
                LDA        #NOTSOS              ; FOR THE PASCAL FOLK
                RTS                                     ; NOTSOS MEANS UCSD OR DOS OR BAD SOS VOLUME
*
VLOOK7          PLA                                     ; THROW AWAY OLD VCB PTR
                JMP        NOVOLM                       ; AND REPORT VOLUME NOT FOUND
*
VLOOK1          JSR        SVCBADR                      ; SAVE ADDRESS OF FREE VCB.
VLOOK2          TXA                                     ; BUMP TO NEXT VOLUME ENTRY.
                CLC
                ADC        #VCBSIZE
                BCC        VOLOOK                       ; BRANCH IF MORE TO CHECK.
                LDX        VCBPTR+1                     ; FREE VCB YET FOUND?
                BNE        VLOOK3                       ; BRANCH IF YES
                LDX        NFOPEN                       ; SAVE POSSIBLE FREE VCB
                JSR        SVCBADR                      ; AND SAVE PTR PERMANENTLY
VLOOK3          LDA        VCBPTR+1                     ; WAS A FREE VCB FOUND?
                BEQ        NOVOLM                       ; BRANCH IF VOLUME CAN'T BE LOGGED IN.
                LDA        SCRTCH+1                     ; GET DEVICE NUMBER
                STA        DEVNUM                       ; SAVE DEVICE NUMBER.
                LDA        #1                           ; FAKE OUT 'LOKVOL'
                STA        SCRTCH                       ; TO THINK TO LOOK ONLY ONCE.
                STA        TOTDEVS
                LDA        #<VCB
                STA        VCBPTR+1
                STA        PATHNMH                      ; (TO MAKE HARMLESS)
                LDA        #0
                STA        SISTER+PATHNMH
                LDX        VCBPTR
                STX        PATHNML
                STA        VCB,X                        ; FORCE CURRENT VOLUME OFF LINE, THEN LOG WHATS THERE.
                JSR        FREEVCB                      ; GO READ ROOT DIRECTORY.
                BCS        RTVOLNAM                     ; RETURN ANY ERRORS
                LDX        VCBPTR                       ; MAKE SURE VOLUME WAS LOGGED IN
                LDA        VCB,X
                BEQ        NOVOLM                       ; RETURN ERROR
                RTS                                     ; ELSE RETURN NORMALLY
NOVOLM          LDA        #VNFERR                      ; TELL USER 'NO VOLUME'
                SEC
RTVOLNAM        TAX                                     ; SAVE REAL ERROR WHILE DUPLICATE IS CHECKED
                LDA        DUPLFLAG
                BEQ        RTV1                         ; BRANCH IF NOT DUPLICATE
                LDX        #DUPVOL
RTV1            TXA                                     ; RECALL ERROR
                RTS

                CHN        VOLUME,4,1

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: PATH
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.3of5.THREE:SOS.PRINT.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: PRINT
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       'SOS 1.1 BLOCK FILE MANAGER' L
* 01-FEB-82
                REL
                IBUFSIZ    1
                SBUFSIZ    40
                INCLUDE    SOSORG,6,1,254
                ORG        ORGBFM               ; BITMAPS $B800-$BBFF
ZZORG           EQU        *
                REP        60
*          (C) COPYRIGHT 1981 BY APPLE COMPUTER INC.
*                     ALL RIGHTS RESERVED
                REP        60
                MSB        OFF
                LST        VSYM
                CHN        EQUATES,4,1
                CHN        ALLOC
                INCLUDE    POSN/OPEN
                INCLUDE    READ/WRITE,2,,4
                INCLUDE    CLOSE/EOF,2,,4
                INCLUDE    DESTROY,2,,4
                INCLUDE    SWAPOUT/IN,2,,4

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: PRINT
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.3of5.THREE:SOS.UMGR.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: UMGR.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                SBTL       "SOS 1.1  UTILITY MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGUMGR
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*  UTILITY MANAGER
*
*  THIS MODULE HANDLES THE FOLLOWING SOS CALLS:
*    SET.FENCE,   GET.FENCE
*    SET.TIME,    GET.TIME
*    JOYSTICK,    COLDSTRT
*
*  IN ADDITION, IT CONTAINS THE ROUITNE DATETIME WHICH
*  PROVIDES THE DATE AND TIME FOR THE BLOCK FILE MANAGER.
*
                REP        60
*
                ENTRY      UMGR
                ENTRY      DATETIME
                ENTRY      BCDBIN
                ENTRY      COLDSTRT
*
                ENTRY      PCLOCK
*
                EXTRN      SYSBANK
                EXTRN      CEVPRI
                EXTRN      SYSERR
                EXTRN      BADSCNUM
                EXTRN      BADJMODE
                EXTRN      XNORESRC
                EXTRN      ALLOCSIR
                EXTRN      DEALCSIR
*
U.TPARMX        EQU        $C0
U.REQCODE       EQU        U.TPARMX
```

```
PRIORITY        EQU        U.TPARMX+1
J.MODE          EQU        U.TPARMX+1
J.VALUE         EQU        U.TPARMX+2
TIME            EQU        U.TPARMX+1
MEMORY          EQU        U.TPARMX+1
*
BITON2          EQU        $04
BITON5          EQU        $20
BITON6          EQU        $40
BITON7          EQU        $80
BITOFF5         EQU        $DF
*
Z.REG           EQU        $FFD0
E.REG           EQU        $FFDF
B.REG           EQU        $FFEF
                PAGE
                REP        35
*
* UTILITY SWITCH
*
                REP        35
*
*
UMGR            EQU        *
                LDA        E.REG                     ;SELECT $C000 I/O SPACE
                ORA        #BITON6
                STA        E.REG
*
                LDA        U.REQCODE
                CMP        #USWCNT
                BCS        UMGRERR
                ASL        A
                TAX
                LDA        USWTBL+1,X
                PHA
                LDA        USWTBL,X
                PHA
                RTS
*
UMGRERR         LDA        #>BADSCNUM
                JSR        SYSERR
*
*  UTILITY SWITCH TABLE
*
USWTBL          EQU        *
                DW         SET.FENCE-1
                DW         GET.FENCE-1
                DW         SET.TIME-1
                DW         GET.TIME-1
                DW         JOYSTICK-1
                DW         COLDSTRT-1
USWCNT          EQU        *-USWTBL/2
                PAGE
                REP        60
*
* SET.FENCE(IN.PRIORITY) SYSTEM CALL
*
* GET.FENCE(OUT.PRIORITY) SYSTEM CALL
*
* THESE TWO CALLS ALLOW THE CALLER TO EITHER RETRIEVE OR SET
* THE CURRENT SYSTEM EVENT PRIORITY THRESHOLD.  BY RAISING
* THE FENCE, A USER MAY INHIBIT THE EXECUTION OF EVENTS WHOSE
* PRIORITY IS EQUAL TO OR LESS THAN THE VALUE OF THE SYSTEM
* FENCE.
*
                REP        60
*
*
SET.FENCE       EQU        *
                LDA        PRIORITY
                STA        CEVPRI
                RTS                                ; NORMAL EXIT
*
*
GET.FENCE       EQU        *
                LDA        CEVPRI
                LDY        #0
                STA        (PRIORITY),Y
                RTS                                ; NORMAL EXIT
                PAGE
                REP        60
*
*  SET.TIME(IN.TIME)
*  GET.TIME(OUT.TIME)
*
*  THESE SYSTEM CALLS ALLOW THE USER TO SET AND READ THE
*  SYSTEM'S CLOCK.  THE TIME IS EXPRESSED AS AN EIGHTEEN
*  DIGIT ASCII STRING IN THE FORM "YYYYMMDDWHHMMSSMMM".
*
*    YYYY  YEAR       [1900-1999]
*      MM  MONTH        [01-12]
*      DD  DAY          [01-31]
*       W  WEEKDAY       [1-7]  1 => SUNDAY
*      HH  HOUR         [00-23]
*      MM  MINUTE       [00-59]
*      SS  SECOND       [00-59]
*     MMM  MILLISECOND [000-999]
*
*  THE CLOCK CHIP AUTOMATICALLY MAINTAINS THE TIME AND
*  DATE FROM MILLISECONDS TO MONTHS.  IT DOES NOT MAINTAIN
*  THE YEAR, HOWEVER, NOR DOES IT RECOGNIZE 29 FEBRUARY
*  IN LEAP YEARS.  THE SOFTWARE SETS THE DAY AND MONTH
*  LATCHES TO THE DON'T CARE STATE AND USES THE REMAINING
*  EIGHT BITS TO HOLD A TWO DIGIT BCD YEAR.  THE CLOCK
*  MUST BE RESET AT THE BEGINNING OF EACH YEAR AND ON
*  29 FEBRUARY IN LEAP YEARS.
*
*  SET.TIME ASSUMES THAT THE DATE IS VALID AND CORRECT.
*  THE CENTURY IS IGNORED AND MILLISECONDS ARE ALWAYS SET
*  TO ZERO.  GET.TIME ALWAYS SETS THE CENTURY TO 19.
*
                REP        60
*
*
*  TEMPORARY ZERO PAGE
*
PCLK            EQU        $D0                  ;POINTER TO SAVED PCLOCK
WKDAY           EQU        $D2
CKSUM           EQU        $D3
CLKTEMP         EQU        $18D4                ;THROUGH $18DD - ABSOLUTE
*
*  CLOCK LOCAL DATA
*
PCLOCK          DS         $0A                  ;PSEUDO CLOCK REGISTERS
RETRY           DS         $01
*
*  CLOCK HARDWARE ADDRESSES
*
CLOCK           EQU        $C070
CSEC            EQU        $02
CMIN            EQU        $03
CMON            EQU        $07
LDAY            EQU        $0E
CRESET          EQU        $12
STATUS          EQU        $14
*
WKMON           DFB        8,11,11,7,9,12
                DFB        7,10,13,8,11,13
*
*
SET.TIME        EQU        *
                LDX        #$00
                LDY        #$12
                LDA        #'0'
                BNE        STIM011
*
STIM010         INX
                LDA        (TIME),Y             ;CONVERT TIME FROM
STIM011         AND        #$0F                 ;  ASCII TO BCD AND
                STA        PCLOCK,X             ;  TRANSFER TO PCLOCK
                DEY
                CPY        #$07
                BEQ        STIM010
                LDA        (TIME),Y
                ASL        A
                ASL        A
                ASL        A
                ASL        A
                ORA        PCLOCK,X
                STA        PCLOCK,X
                DEY
                BPL        STIM010
*
                LDA        PCLOCK+7             ;CALCULATE WEEKDAY
                JSR        BCDBIN
                TAX
                LDA        PCLOCK+8
                JSR        BCDBIN
                TAY
                LSR        A
                LSR        A
                STA        WKDAY
                TYA
                AND        #$03
                BNE        STIM015
                CPX        #3
                BCS        STIM015              ; <SRS 82.162>
                DEY
STIM015         CLC
                TYA
                ADC        WKDAY
                ADC        WKMON-1,X
                STA        WKDAY
                LDA        PCLOCK+6
                JSR        BCDBIN
                CLC
                ADC        WKDAY
                SEC
STIM016         SBC        #7
                CMP        #8
                BCS        STIM016
                STA        PCLOCK+5
*
                LDA        #$D0
                STA        PCLK                 ;POINT (PCLK) TO 8F:FFD0
                LDA        #$FF
                STA        PCLK+1
                LDA        #$8F
                STA        $1401+PCLK
                LDA        #$A5
                STA        CKSUM                ;INITIALIZE CHECKSUM
                LDY        #$00
*
STIM020         LDA        PCLOCK,Y             ;SAVE PCLOCK
                STA        (PCLK),Y             ;  BEHIND 6522
                EOR        CKSUM
                STA        CKSUM
                INY
                CPY        #$0A
                BCC        STIM020
                STA        (PCLK),Y             ;SAVE CHECKSUM
*
                LDA        Z.REG
                PHA                             ;SAVE ZERO PAGE
                LDA        E.REG
                PHA                             ;SAVE ENVIRONMENT
                ORA        #BITON7              ;  AND SET 1 MHZ
                STA        E.REG
*
                LDY        #STATUS
                STY        Z.REG
                LDA        CLOCK                ;DOES CLOCK EXIST?
                BMI        STIM050              ;  NO
*
                LDX        #CRESET
                STX        Z.REG
                LDA        #$FF                 ;RESET ALL COUNTERS
                STA        CLOCK
                STA        CLOCK
*
                LDX        #CSEC-1
STIM030         INX
                PHP
                SEI                             ;DISABLE INTERRUPTS
STIM040         STX        Z.REG
                LDA        CLOCK                ;(DUMMY READ FOR STATUS)
                LDA        PCLOCK,X
                STA        CLOCK                ;SET CLOCK COUNTER
                LDA        CLOCK                ;(DUMMY READ FOR STATUS)
                STY        Z.REG
                LDA        CLOCK                ;CHECK STATUS BIT
                BNE        STIM040
                PLP                             ;RESTORE INTERRUPTS
                CPX        #CMON
                BCC        STIM030
*
                LDX        #LDAY
                STX        Z.REG
                LDA        PCLOCK+8
                ORA        #$CC                 ;STUFF YEAR INTO DAY
                STA     CLOCK                 ;  AND MONTH LATCHES
                INC     Z.REG
                LDA     PCLOCK+8
                LSR     A
                LSR     A
                ORA     #$CC
                STA     CLOCK
*
STIM050         PLA
                STA     E.REG                 ;RESTORE ENVIRONMENT
                PLA
                STA     Z.REG                 ;  AND ZERO PAGE
                RTS
                PAGE
GET.TIME        EQU     *
                LDA     Z.REG                 ;SAVE ZERO PAGE
                PHA
                LDA     E.REG                 ;SAVE ENVIRONMENT
                PHA
                ORA     #BITON7
                STA     E.REG                 ;SET 1 MHZ
*
                LDY     #STATUS
                STY     Z.REG
                LDA     CLOCK                 ;DOES CLOCK EXIST?
                BMI     GTIM050               ;  NO
*
                LDA     #$10                  ;ALLOW $10 RETRYS
                STA     RETRY
GTIM010         LDX     #CMON+1
                PHP
                SEI                           ;DISABLE INTERRUPTS
*
GTIM020         DEX
                BMI     GTIM030               ;ALL DONE
                STX     Z.REG
                LDA     CLOCK                 ;COPY CLOCK COUNTERS
                STA     CLKTEMP,X             ;  TO TEMP REGISTERS
                STY     Z.REG
                LDA     CLOCK                 ;CHECK STATUS BIT
                BEQ     GTIM020
*
                PLP                           ;CLOCK READ ERROR
                DEC     RETRY
                BPL     GTIM010               ;TRY AGAIN
                BMI     GTIM050
*
GTIM030         PLP                           ;RESTORE INTERRUPTS
                LDX     #LDAY+1
                STX     Z.REG
                LDA        CLOCK                ;READ YEAR FROM DAY
                SEC                             ;  AND MONTH LATCHES
                ROL        A
                ROL        A
                DEC        Z.REG
                AND        CLOCK
                STA        CLKTEMP+8
*
                LDX        #$09
GTIM040         LDA        CLKTEMP,X            ;COPY CLOCK DATA
                STA        PCLOCK,X             ;  TO PSEUDO CLOCK
                DEX
                BPL        GTIM040
*
GTIM050         LDA        #$19
                STA        PCLOCK+9
*
                PLA
                STA        E.REG                ;RESTORE ENVIRONMENT
                PLA
                STA        Z.REG                ;  AND ZERO PAGE
*
                LDY        #$11
                LDX        #$00
GTIM060         LDA        PCLOCK,X             ;GET MOST SIGNIFICANT
                LSR        A                    ;  BCD DIGIT
                LSR        A
                LSR        A
                LSR        A
                ORA        #$30                 ;CONVERT TO ASCII
                STA        (TIME),Y
                INX
                DEY
                BMI        GTIM080
GTIM070         LDA        PCLOCK,X             ;GET LEAST SIGNIFICANT
                AND        #$0F                 ;  BCD DIGIT
                ORA        #$30                 ;CONVERT TO ASCII
                STA        (TIME),Y
                DEY
                CPY        #$07
                BNE        GTIM060
                INX
                BNE        GTIM070
GTIM080         RTS
                PAGE
                REP        60
*
*  SUBROUTINE DATETIME
*
*  THIS SUBROUTINE READS THE CLOCK AND WRITES A DATE/TIME
*  STAMP TO A FOUR BYTE BUFFER ON THE CALLER'S ZERO PAGE;
*  THE DATA FORMAT IS SHOWN BELOW.  ON ENTRY, X MUST POINT
*  TO THE BUFFER.  ON EXIT, ALL REGISTERS ARE CLOBBERED.
*  IF AN ERROR OCCURS, CARRY IS SET AND THE BUFFER IS
*  SET TO ZERO; OTHERWISE, CARRY IS CLEARED.
*
*   BITS: 7 6 5 4 3 2 1 0
*    X+0  M M M D D D D D
*    X+1  Y Y Y Y Y Y Y M
*    X+2   -  MINUTE   -
*    X+3   - - HOUR  - -
*
                REP        60
*
*  TEMPORARY STORAGE
*
OFFSET          DFB        0
ERRCNT          DFB        0
CLKREGS         DS         5
MIN             EQU        CLKREGS+0
HOUR            EQU        CLKREGS+1
DAY             EQU        CLKREGS+3
MON             EQU        CLKREGS+4
YEAR            EQU        CLKREGS+2
*
*
DATETIME        EQU        *
                STX        OFFSET
                LDA        Z.REG
                PHA                             ;SAVE ZERO PAGE
                LDA        E.REG
                PHA                             ;  AND ENVIRONMENT
                ORA        #BITON7+BITON6       ;SET 1 MHZ AND
                STA        E.REG                ;  ENABLE I/O SPACE
*
                LDY        #STATUS
                STY        Z.REG
                LDA        CLOCK                ;DOES CLOCK EXIST?
                BMI        DT030                ;  NO
*
                LDA        #8
                STA        ERRCNT               ;ALLOW 8 RETRYS
DT010           LDX        #CMON+1
                PHP
                SEI                             ;DISABLE INTERRUPTS
*
DT020           DEX
                CPX        #CMIN
                BCC        DT050
                STX        Z.REG
                LDA        CLOCK                ;READ THE CLOCK
                STA        CLKREGS-CMIN,X
                STY        Z.REG
                LDA        CLOCK                ;CHECK STATUS
                BEQ        DT020
*
                PLP                             ;CLOCK READ ERROR
                DEC        ERRCNT
                BPL        DT010
DT030           PLA
                STA        E.REG                ;RESTORE ENVIRONMENT
                PLA
                STA        Z.REG                ;  AND ZERO PAGE
                LDX        #CMON-CMIN
DT040           LDA        PCLOCK+CMIN,X
                STA        CLKREGS,X
                DEX
                BPL        DT040
                LDX        PCLOCK+8
                JMP        DT060
*
DT050           PLP                             ;READ YEAR FROM LATCHES
                LDA        #LDAY+1
                STA        Z.REG
                LDA        CLOCK
                SEC
                ROL        A
                ROL        A
                DEC        Z.REG
                AND        CLOCK
                TAX
*
                PLA
                STA        E.REG                ;RESTORE ENVIRONMENT
                PLA
                STA        Z.REG                ;  AND ZERO PAGE
*
DT060           TXA
                JSR        BCDBIN               ;CONVERT YEAR TO BINARY
                STA        YEAR
                LDA        MON                  ;CONVERT MONTH AND DAY
                JSR        BCDBIN               ;  TO BINARY THEN
                ASL        A                    ;  COMBINE WITH YEAR
                ASL        A                    ;  TO FORM DATE STAMP
                ASL        A
                ASL        A
                ASL        A
                STA        MON
                ROL        YEAR
                LDA        DAY
                JSR        BCDBIN
                ORA        MON
                LDX        OFFSET
                STA        0,X
                LDA        YEAR
                STA        1,X
                LDA        MIN                  ;CONVERT MINUTE
                JSR        BCDBIN
                STA        2,X
                LDA        HOUR                 ;CONVERT HOUR
                JSR        BCDBIN
                STA        3,X
                CLC
                RTS
                PAGE
                REP        60
*
*  SUBROUTINE BCDBIN
*
*  THIS SUBROUTINE CONVERTS A BYTE FROM BCD TO BINARY.
*  THE BYTE IS PASSED AND RETURNED IN A.  THERE IS NO
*  ERROR CHECKING.  Y IS DESTROYED AND X IS UNCHANGED.
*
                REP        60
*
BCDBIN          EQU        *
                PHA
                LSR        A                    ;ISOLATE TENS DIGIT FOR
                LSR        A                    ;  INDEXING THE TABLE
                LSR        A
                LSR        A
                TAY
                PLA
                AND        #$0F                 ;GET UNITS
                CLC
                ADC        TENS,Y               ;ADD IN TENS
                RTS
*
TENS            DFB        00,10,20,30,40,50,60,70,80,90
                PAGE
                REP        60
*
*  SOS CALL $64 -- JOYSTICK INPUT
*    JOYSTICK(IN.J.MODE; OUT.J.VALUE)
*
                REP        60
*
*
AD.INPUT        EQU        $D0
AD.TEMP         EQU        $D1
```

```
*
PA.SW0          EQU        $C061                ;PORT A, SWITCH 0
PA.SW1          EQU        $C063                ;PORT A, SWITCH 1
PB.SW0          EQU        $C062                ;PORT B, SWITCH 0
PB.SW1          EQU        $C060                ;PORT B, SWITCH 1
*
AD.SEL0         EQU        $C058                ;A/D SELECT CONTROLS
AD.SEL1         EQU        $C05E
AD.SEL2         EQU        $C05A
AD.CHRG         EQU        $C05C                ;A/D RAMP CHARGE /
AD.STRT         EQU        $C05D                ;    START TIMEOUT
AD.FLAG         EQU        $C066                ;A/D TIMEOUT FLAG
*
TCHARGE         EQU        500                  ;CHARGE TIME FOR A/D
TOFFSET         EQU        360                  ;OFFSET TIME TO A/D WINDOW
*
ANALOG          EQU        $F4A8                ;ROM ENTRY FOR ANALOG INPUT
ANLOG1          EQU        $F4AB                ;  INTERRUPT REENTRY
D.T2            EQU        $FFD8                ;TIMER
D.ACR           EQU        $FFDB                ;AUXILIARY CONTROL REGISTER
D.IFR           EQU        $FFDD                ;INTERRUPT FLAG REGISTER
*
ENSEL           EQU        $C0DC
ENSIO           EQU        $C0DE
*
*
JOYSTICK        EQU        *
                LDA        J.MODE               ;VALIDATE J.MODE
                CMP        #$08
                BCC        JS010
                LDA        #>BADJMODE
JS.ERR          JSR        SYSERR
*
JS010           JSR        AD.SETUP             ;SET UP RESOURCES
                BCS        JS.ERR
                LDA        J.MODE               ;READ PORT B OR PORT A?
                AND        #BITON2
                BNE        JS020
                LDA        PB.SW0               ;PORT B
                LDX        PB.SW1
                LDY        #$01
                BNE        JS030
JS020           LDA        PA.SW0               ;PORT A
                LDX        PA.SW1
                LDY        #$03
JS030           STY        AD.INPUT             ;SAVE INPUT SELECT
                AND        #BITON7
                BEQ        JS040
                LDA        #$FF
JS040           LDY        #$00
```

```
                STA        (J.VALUE),Y          ;RETURN SWITCH 0
                TXA
                AND        #BITON7
                BEQ        JS050
                LDA        #$FF
JS050           INY
                STA        (J.VALUE),Y          ;RETURN SWITCH 1
*
                LSR        J.MODE
                BCC        JS060
                LDA        AD.INPUT
                JSR        AD.READ              ;READ A/D
                LDY        #$02
                STA        (J.VALUE),Y          ;RETURN X AXIS
JS060           INC        AD.INPUT
                LSR        J.MODE
                BCC        JS070
                LDA        AD.INPUT
                JSR        AD.READ              ;READ A/D
                LDY        #$03
                STA        (J.VALUE),Y          ;RETURN Y AXIS
*
JS070           JSR        AD.CLNUP             ;CLEAN UP
                RTS                             ;  AND EXIT
                PAGE
                REP        60
*
*  SUBROUTINE AD.SETUP
*  THIS SUBROUTINE SETS UP THE ENVIRONMENT AND RESOURCES
*  FOR READING THE JOYSTICKS.  IF AN ERROR OCCURS, CARRY
*  IS SET AND AN ERROR NUMBER IS RETURNED IN A.
*  OTHERWISE, CARRY IS CLEARED.
*
                REP        60
AD.SETUP        EQU        *
                LDA        #JOYSIRSIZ
                LDX        #>JOYSIRTBL
                LDY        #<JOYSIRTBL
                JSR        ALLOCSIR             ;ALLOCATE RESOURCES
                BCC        ADS010
                LDA        #>XNORESRC
                RTS
ADS010          LDA        E.REG
                AND        #$7F                 ;SET 2 MHZ,
                ORA        #$43                 ;  ENABLE ROM, & I/O SPACE
                STA        E.REG
                PHP
                SEI
                LDA        D.ACR
                AND        #BITOFF5             ;SET UP TIMER
                STA     D.ACR
                PLP
                BIT     ENSEL               ;DISABLE ENSEL
                BIT     ENSIO               ;SET ENSIO FOR INPUT
                RTS
*
JOYSIRTBL       EQU     *
                DFB     $0C,0,0,0,0         ;ENSIO
                DFB     $0D,0,0,0,0         ;ENSEL
                DFB     $0E,0,0,0,0         ;6522 D.T2
JOYSIRSIZ       EQU     *-JOYSIRTBL
                REP     60
*
*  SUBROUTINE AD.CLNUP
*  THIS SUBROUTINE RESTORES THE ENVIRONMENT AND RELEASES
*  THE RESOURCES AFTER READING THE JOYSTICKS.
*
                REP     60
AD.CLNUP        EQU     *
                LDA     E.REG
                AND     #$3C                ;RESTORE RAM AT $C000 & $F000
                STA     E.REG
                LDA     #JOYSIRSIZ
                LDX     #>JOYSIRTBL
                LDY     #<JOYSIRTBL
                JSR     DEALCSIR            ;DEALLOCATE RESOURCES
                RTS
                PAGE
                REP     60
*
*  SUBROUTINE AD.READ
*  THIS SUBROUTINE READS A SPECIFIED A/D INPUT AND RETURNS
*  AN 8 BIT RESULT.  IT ASSUMES THAT THE A/D RESOURCES HAVE
*  BEEN ALLOCATED, THE I/O SPACE AND $F000 ROM HAVE BEEN
*  SELECTED, AND THE SYSTEM IS RUNNING IN 2 MHZ MODE.
*
*  PARAMETERS:
*    A:  A/D INPUT PORT (0-7)
*
*  RETURN VALUE:
*    A:  RESULT (0 - 255)
*    X, Y:  UNDEFINED
*
                REP     60
*
AD.READ         EQU     *
                LSR     A                   ;SELECT THE APPROPRIATE
                BIT     AD.SEL0             ;  A/D INPUT
                BCC     ADR010
                BIT     AD.SEL0+1
```

```
ADR010          LSR        A
                BIT        AD.SEL1
                BCC        ADR020
                BIT        AD.SEL1+1
ADR020          LSR        A
                BIT        AD.SEL2
                BCC        ADR030
                BIT        AD.SEL2+1
ADR030          PHP
*
ADR040          CLI
                BIT        AD.CHRG               ;CHARGE A/D CAPACITOR
                LDA        #>TCHARGE
                STA        D.T2
                LDA        #<TCHARGE
                STA        D.T2+1
                LDA        #BITON5
ADR050          BIT        D.IFR
                BEQ        ADR050
*
                SEI
                SEC
                LDA        #>TOFFSET
                STA        D.T2                  ;SET UP TIMER
                LDA        #<TOFFSET
                BIT        AD.STRT               ;START A/D TIMEOUT
                JSR        ANALOG                ;MEASURE CONVERSION TIME
                BCC        ADR070
*
ADR060          CLI                              ;PROCESS AN INTERRUPT
                SEI
                BIT        AD.FLAG               ;STILL TIMING?
                BPL        ADR040                ;  NO -- START OVER
                JSR        ANLOG1                ;  YES -- CONTINUE
                BCS        ADR060
*
ADR070          PLP
                EOR        #$FF                  ;NORMALIZE RESULT
                BMI        ADR080                ;RESULT < 0
                STA        AD.TEMP
                TYA
                EOR        #$FF
                LSR        AD.TEMP
                ROR        A
                LSR        AD.TEMP
                ROR        A
                LSR        AD.TEMP
                BNE        ADR090                ;RESULT > 255
                ROR        A
                ADC        #0
                RTS
ADR080          LDA        #0
                RTS
ADR090          LDA        #$FF
                RTS
                PAGE
                REP        60
*
*  SYSTEM COLD START
*
*  THIS ROUTINE IS CALLED TO TELL THE USER TO REBOOT THE
*  SYSTEM.  IT CLEARS THE SCREEN, DISPLAYS A MESSAGE,
*  OVERWRITES BANKED MEMORY, AND HANGS UNTIL THE USER
*  PERFORMS A HARD RESET.
*
                REP        60
*
*
COLDSTRT        EQU        *
                SEI                             ;SHUT DOWN INTERRUPTS
                LDA        #$40                 ;  AND IGNORE NMI
                STA        $FFCA
                LDA        #$67
                STA        E.REG                ;DISABLE RESET
                LDA        #$00
                STA        Z.REG                ;USE PAGE ZERO
*
                LDX        SYSBANK
                LDA        #$BF
                LDY        #$00
                STY        MEMORY
CS010           STA        MEMORY+1
                STX        B.REG
                LDA        #$A0
CS020           STA        (MEMORY),Y           ;SET MEMORY TO BLANKS
                DEY
                BNE        CS020
                DEC        MEMORY+1
                BNE        CS020
                DEX
                BPL        CS010
*
                LDY        #6
CS030           STA        $C050,Y              ;SELECT 40 COLUMN
                DEY                             ;  BLACK & WHITE TEXT
                BPL        CS030
*
                LDY        #BOOTLEN
CS040           LDA        BOOTMSG-1,Y          ;PRINT BOOT MESSAGE
                STA        BOOTADR-1,Y
                DEY
                BNE        CS040
*
                LDA        #$77
                STA        E.REG                ;ENABLE RESET
                JMP        *                    ;HANG UNTIL RESET
                PAGE
                MSB        ON
BOOTMSG         ASC        "INSERT SYSTEM DISKETTE & REBOOT"
BOOTLEN         EQU        *-BOOTMSG
BOOTADR         EQU        40-BOOTLEN/2+$628
                MSB        OFF
                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENUMGR
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR UMBR"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: UMGR.SRC
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.3of5.THREE:SOS.VOLUME.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: VOLUME
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
                REP        40
* NAME    : VOLUME
* FUNCTION: RETURN VOLUME INFO
* INPUT   : DEVICE NAME
* OUTPUT  : THE INFO
* VOLATILE: ALL REGS
                REP        40
*
VOLUME          EQU        *
                LDA        C.DNAMP             ; TRANSFER DEVICE NAME
                STA        DVNAMP              ; NAME FOR DMGR
                LDA        C.DNAMP+1
                STA        DVNAMP+1
                LDA        SISTER+C.DNAMP+1    ; AND XTND
                STA        SISTER+DVNAMP+1
                JSR        GETDNUM             ; GET DEVNUM
                BCC        VOL7                ; =>SOME KINDA ERROR
                RTS                            ; RETURN ERROR
VOL7            BMI        VOL2                ; =>IT'S GOOD...
                LDA        #NOTBLKDEV          ; NOT BLOCKED
                JMP        VOLERR              ; =>RETURN THE ERROR
*
* UNCONDITIONALLY READ ROOT DIRECTORY:
*
VOL2            EQU        *
                LDA        SCRTCH+1
                STA        DEVNUM              ; SETUP DEV NUMBER
                LDA        #2                  ; BLKNUM=2
                LDX        #0
                JSR        GETROT0             ; GET IT PLEASE
                LDA        #VNFERR             ; ERROR CODE
                BCC        VOL8                ; BRANCH IF NO ERROR ON READ
                RTS                            ; =>ERROR, PASS IT ON.
*
VOL8            LDA        #>VCB               ; SET VCBPTR TO THE
                STA        VCBPTR              ; FIRST OF THEM
                LDA        #<VCB
                STA        VCBPTR+1
*
```

```
* IS THIS VOLUME SOS OR OTHER?
*
                JSR        TSTSOS               ; WHICH KIND?
                BCC        VLOGGED              ; =>IT'S SOS
                JMP        VNOTSOS              ; =>NOT SOS
*
* IS THIS SOS VOLUME LOGGED IN?
*
VLOGGED         EQU        *
                JSR        CMPVCB               ; DOES VOLNAME MATCH?
                BCC        VFOUND               ; =>YES, WE KNOW ABOUT IT.
                JSR        VNXTVCB              ; BUMP TO NEXT
                BCC        VLOGGED              ; =>TRY 'EM ALL...
                BCS        VNEW                 ; =>NOT FOUND, IT'S NEW (BRANCH ALWAYS)
*
*
* IT'S BEEN LOGGED IN BEFORE:
*  IS IT SWAPPED IN OR OUT?
*
VFOUND          EQU        *
                LDY        #VCBSWAP             ; INDEX TO IT
                LDA        (VCBPTR),Y           ; SWAPPED?
                BPL        VFOUND1              ; =>IN. RETURN THE INFO
*
* SWAPPED OUT. BEFORE WE SWAP IT
*  IN, MAKE SURE IT BELONGS ON
*  THIS DEVICE!
*
                LDY        #VCBDEV              ; INDEX TO IT
                LDA        (VCBPTR),Y           ; GET ITS DEVICE
                CMP        DEVNUM               ; CORRECT DEVICE?
                BEQ        VSWAPIN              ; =>YES
                LDA        #DUPVOL              ; IF FOR ANOTHER DEV,
                JMP        VOLERR               ; THEN IT'S AN ERROR!
*
* NOW SWAP-IN THIS VOLUME:
*
VSWAPIN         EQU        *
                JSR        SWAPIN               ; SWAP IT IN
                JMP        VINFO                ; AND RETURN THE INFO
*
VFOUND1         LDY        #VCBDEV
                LDA        (VCBPTR),Y           ; SAME DEVICES?
                CMP        DEVNUM
                BEQ        VINFO                ; YES; RETURN THE INFORMATION
                LDY        #VCBSTAT
                LDA        (VCBPTR),Y           ; OPEN FILES?
                BPL        VFOUND2              ; BRANCH IF NOT
                LDA        #DUPVOL
                BNE        VOLERR               ; ELSE REPORT DUPLICATE VOLUME ERROR (BRANCH ALWAYS)
```

```
VFOUND2         LDY        #VCBNML              ; MOVE THE LOGIN TO THIS NEW DEVICE
                LDA        #0                   ; BY UNLOGGING THE OLD
                STA        (VCBPTR),Y           ; AND LOGGING IN THE NEW (DROP INTO VNEW)
                REP        40
*
* IT'S A BRAND NEW VOLUME.
*  GUESS WE'LL HAVE TO LOG IT IN:
*
VNEW            EQU        *
                LDA        DEVNUM               ; PASS A REG TO SWAPOUT
                JSR        SWAPOUT              ; SWAP ANY ACTIVE VOL ON THIS DEVICE
                BCC        VNEW1                ; BRANCH ON NO ERROR
                LDA        #XIOERROR
                RTS
VNEW1           LDA        #>VCB                ; FIND AN EMPTY VCB
                STA        VCBPTR
                LDA        #<VCB
                STA        VCBPTR+1
VFREE           LDY        #VCBNML
                LDA        (VCBPTR),Y           ; EMPTY VCB?
                BEQ        VLOGIN               ; ITS FREE, USE IT
                LDY        #VCBDEV
                LDA        (VCBPTR),Y           ; OR ONE WITH SAME DEVICE
                CMP        DEVNUM
                BNE        VFREEX               ; BRANCH IF NO DEVICE MATCH
                LDY        #VCBSTAT
                LDA        (VCBPTR),Y           ; AND NO OPEN FILES
                BPL        VLOGIN               ; BRANCH IF OK TO REUSE THIS VCB
                LDA        DEVNUM               ; THEN WE MUST SWAP OUT THIS VOLUME
                JSR        SWAPOUT
                BCC        VFREEX               ; SWAPOUT PROCEEDED OK
                LDA        #XIOERROR            ; ELSE REPORT ERROR
                RTS
VFREEX          JSR        VNXTVCB              ; TRY NEXT
                BCC        VFREE                ; MORE TO COME
* RAN OUT OF MT'S ... FIND W/O FILES
VNFIL           LDY        #VCBSTAT
                LDA        (VCBPTR),Y
                BPL        VLOGIN
                JSR        VNXTVCB
                BCC        VNFIL
* ALL OPEN ... REPORT VCBFULL
                LDA        #FCBFULL
                BNE        VOLERR
VLOGIN          EQU        *
                JSR        LOGVCB               ; AND LOGIN THIS ONE
                REP        40
*
* RETURN ALL THE NICE INFO:
*
```

```
VINFO           EQU        *
                LDA        #0
                LDY        #VCBTFRE             ; FETCH VOLUME FREE BLOCK COUNT
                STA        (VCBPTR),Y           ; FORCE RESCAN OF ALL
                INY                             ; BITMAPS
                STA        (VCBPTR),Y           ; TO MAKE SURE VCB INFO CURRENT
                STA        REQL                 ; FREE BLOCKS
                STA        REQH
                JSR        TSFRBLK
*
                LDX        VCBPTR               ; GET VCB INDEX
                LDY        #0
VINFO1          EQU        *
                LDA        VCB+VCBTBLK,X        ; MOVE TOTAL
                STA        (C.OUTBLK),Y         ; BLOCKS AVAIL
                INX
                INY
                CPY        #4                   ; AND FREE ONES TOO
                BNE        VINFO1
*
                LDY        #0                   ; NOW DO VOLNAME
                LDA        (VCBPTR),Y
                TAY
VINFO2          EQU        *
                LDA        (VCBPTR),Y
                STA        (C.OUTVOL),Y
                DEY
                BPL        VINFO2
                CLC
                BCC        VOLRET               ; =>DONE
*
VOLERR          EQU        *
                SEC
VOLRET          EQU        *
                RTS
                PAGE
                REP        40
* THIS ISN'T A SOS VOLUME. MARK
*  THE ACTIVE VOL THIS DEVICE
*  SO THAT IT GETS CHECKED LATER:
*
VNOTSOS         EQU        *
                LDY        #VCBDEV              ; IS VCB FOR THIS
                LDA        (VCBPTR),Y           ; DEVICE?
                CMP        DEVNUM
                BNE        VNS2
                LDY        #VCBSTAT             ; INDEX TO IT
                LDA        (VCBPTR),Y           ; GET STATUS
                BPL        VNS2                 ; =>NOT ACTIVE.
                ORA        #DSWITCH             ; SET 'SWITCHEROO'
                STA        (VCBPTR),Y           ; PUT IT BACK
*
VNS2            EQU        *
                JSR        VNXTVCB              ; GET NEXT VCB
                BCC        VNOTSOS              ; =>TRY 'EM ALL.
*
                LDA        #NOTSOS              ; GIVE THE ERROR
                BNE        VOLERR               ; (BRANCH ALWAYS)
                SKP        5
* NAME    : VNXTVCB
* FUNCTION: BUMP VCBPTR TO NEXT VCB
* INPUT   : NOTHING
* OUTPUT  : VCBPTR UPDATED
*         : 'BCC' IF MORE TO GO
*         : 'BCS' IF DONE
* VOLATILE: AC
*
VNXTVCB         EQU        *
                LDA        VCBPTR
                CLC
                ADC        #VCBSIZE             ; BUMP IT
                STA        VCBPTR
                RTS                             ; CARRY SET IF END OF PAGE

                CHN        CREATE,4,1

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: VOLUME
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.4of5.FOUR:SOS.CLOSE.EOF.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: CLOSE.EOF
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
*
*
CLOSE           LDA        C.REFNUM             ; CLOSE ALL?
                BNE        CLOSE1               ; NO, JUST ONE OF 'EM
                STA        CFERR                ; CLEAR GLOBAL CLOSE ERROR
                JSR        GFCBADR              ; SET UP POINTER TO FCB
CLOSALL         LDA        #0                   ; BEGIN AT THE BEGINNING.
CLSALL1         STA        FCBPTR               ; SAVE CURRENT LOW BYTE OF POINTER
                LDY        #FCBLEVL             ; FETCH THE LEVEL AT WHICH
                LDA        (FCBPTR),Y           ; FILE WAS OPENED
                CMP        LEVEL                ; TEST AGAINST CURRENT GLOBAL LEVEL
                BCC        NXTCLOS              ; DONT CLOSE IF FILES LEVEL IS < GLOBAL LEVEL
                LDY        #FCBREFN             ; INDEX TO REFERENCE NUMBER
                LDA        (FCBPTR),Y           ; IS THIS REFERENCE FILE OPEN?
                BEQ        NXTCLOS              ; NO, TRY NEXT.
                JSR        FLUSH2               ; CLEAN IT OUT...
                BCS        CLOSERR              ; RETURN FLUSH ERRORS
                JSR        CLOSE2               ; UPDATE FCB & VCB
                LDY        C.REFNUM
                BEQ        NXTCLOS              ; NO ERR IF CLOSE ALL
                BCS        CLOSERR
NXTCLOS         LDA        FCBPTR               ; BUMP POINTER TO NEXT FILE CONTROL BLOCK.
                CLC
                ADC        #$20
                BCC        CLSALL1              ; BRANCH IF WITHIN SAME PAGE.
                LDA        FCBPTR+1
                INC        FCBPTR+1             ; BUMP TO NEXT PAGE.
                CMP        FCBADDRH             ; HAVE WE CHECKED BOTH PAGES?
                BEQ        CLOSALL              ; YES, RETURN NO ERROR.
                CLC
                LDA        CFERR                ; ON FINAL CLOSE OF CLOSE ALL REPORT LOGGED ERRORS
                BEQ        C3                   ; BRANCH IF NO ERRORS
                SEC
C3              RTS
*
*
CFERR           DS         1                    ; GLOBAL ERROR FLAG FOR FLUSH AND CLOSE ALL
*
*
```

```
CLOSE1          JSR        FLUSH1               ; FLUSH FILE FIRST (INCLUDING UPDATING BIT MAP)
                BCS        CLOSERR
CLOSE2          LDY        #FCBBUFN
                LDA        (FCBPTR),Y
                JSR        RELBUF
                BCS        CLOSERR
                LDA        #0
                LDY        #FCBREFN
                STA        (FCBPTR),Y
                INY                             ; BUMP TO 'FCBDEVN'
                LDA        (FCBPTR),Y
                STA        DEVNUM               ; GO LOOK FOR ASSOCIATED VCB.
                JSR        DEVVCB
                LDX        VCBPTR               ; GET VCBPTR
                DEC        VCB+VCBOPNC,X        ; INDICATE ONE LESS FILE OPEN.
                BNE        CLOSEND              ; BRANCH IF THAT WASN'T THE LAST...
                LDA        VCB+VCBSTAT,X
                AND        #$7F                 ; STRIP 'FILES OPEN' BIT
                STA        VCB+VCBSTAT,X
CLOSEND         CLC
                RTS
CLOSERR         JMP        GLBERR               ; DON'T REPORT CLOSALL ERR NOW
*
                PAGE
*
FLUSH           LDA        C.REFNUM             ; FLUSH ALL?
                BNE        FLUSH1               ; NO, JUST ONE OF 'EM
                STA        CFERR                ; CLEAR GLOBAL FLUSH ERROR
                JSR        GFCBADR              ; SET UP POINTER TO FCB
FLSHALL         LDA        #0                   ; BEGIN AT THE BEGINNING.
FLSHAL1         STA        FCBPTR               ; SAVE CURRENT LOW BYTE OF POINTER
                LDY        #FCBREFN             ; INDEX TO REFERENCE NUMBER
                LDA        (FCBPTR),Y           ; IS THIS REFERENCE FILE OPEN?
                BEQ        NXFLUSH              ; NO, TRY NEXT.
                JSR        FLUSH2               ; CLEAN IT OUT...
                BCS        FLSHERR              ; RETURN ANY ERRORS
*
                BCS        CLOSERR
NXFLUSH         LDA        FCBPTR               ; BUMP POINTER TO NEXT FILE CONTROL BLOCK.
                CLC
                ADC        #$20
                BCC        FLSHAL1              ; BRANCH IF WITHIN SAME PAGE.
                LDA        FCBPTR+1
                INC        FCBPTR+1             ; BUMP TO NEXT PAGE.
                CMP        FCBADDRH             ; HAVE WE CHECKED BOTH PAGES?
                BEQ        FLSHALL              ; YES, RETURN NO ERROR.
FLUSHEND        CLC
                LDA        CFERR                ; ON LAST FLUSH OF A FLUSH(0)
                BEQ        F3                   ; BRANCH IF NO LOGGED ERRORS
                SEC                             ; REPORT ERROR NOW
```

```
F3              RTS
FLSHERR         JMP        GLBERR               ; FLUSH ALL OR ONE?
*
FLUSH2          JSR        FNDFCBUF             ; MUST SET UP ASSOCIATED VCB AN BUFFER LOCATIONS FIRST.
                BCC        FLUSH2A              ; BRANCH IF NO ERROR ENCOUNTERED.
                JMP        GLBERR               ; CHECK FOR CLOSE OR FLUSH ALL
*
FLUSH1          LDA        #0                   ; CLEAR
                STA        CFERR                ; GLOBAL ERROR FOR NORMAL REFNUM FLUSH
                JSR        FINDFCB              ; SET UP POINTER TO FCB USER REFERENCES
                BCS        FLSHERR              ; RETURN ANY ERRORS
FLUSH2A         LDY        #FCBATTR             ; TEST TO SEE IF FILE IS
                LDA        (FCBPTR),Y           ; MODIFIED. FIRST TEST WRITE ENABLED.
                AND        #WRITEN
                BEQ        FLUSHEND             ; BRANCH IF 'READ ONLY'
                LDY        #FCBDIRTY            ; SEE IF EOF HAS BEEN MODIFIED
                LDA        (FCBPTR),Y
                BMI        FLUSH2B              ; BRANCH IF IT HAS
                LDY        #FCBSTAT             ; NOW TEST FOR DATA MODIFIED.
                LDA        (FCBPTR),Y           ; (IN OTHER WORDS: WAS FILE ACTUALLY
                AND        #USEMOD+EOFMOD+DATMOD ; WRITTEN TO WHILE IT'S BEEN OPEN?)
                BEQ        FLUSHEND             ; BRANCH IF FILE NOT MODIFIED.
FLUSH2B         JSR        TWRPROT1             ; DISK SWITCH CHECKING
                LDA        DSWGLOB
                BEQ        FLUSH2C              ; BRANCH IF NO SWITCH
                LDA        #XDISKSW
                SEC
                RTS                             ; FORCES A VERIFIED RETRY
FLUSH2C         LDY        #FCBSTAT             ; NOW TEST FOR DATA MODIFIED.
                LDA        (FCBPTR),Y
                AND        #DATMOD              ; DOES CURRENT DATA BUFFER NEED TO BE
                BEQ        FLUSH3               ; WRITTEN? BRANCH IF NOT.
                JSR        WFCBDAT              ; IF SO, GO WRITE IT STUPID!
                BCS        FLSHERR
FLUSH3          LDY        #FCBSTAT             ; CHECK TO SEE IF THE INDEX BLOCK (TREE FILES ONLY)
                LDA        (FCBPTR),Y           ; NEEDS TO BE WRITTEN.
                AND        #IDXMOD
                BEQ        FLUSH4               ; BRANCH IF NOT...
                JSR        WFCBIDX
                BCS        FLSHERR              ; RETURN ANY ERRORS.
                PAGE
*
FLUSH4          LDY        #FCBENTN             ; NOW PREPARE TO UPDATE DIRECTORY
OWNRMOV         LDA        (FCBPTR),Y           ; NOTE: THIS CODE DEPENDS ON THE
                STA        D.DEV-FCBDEVN,Y      ; DEFINED ORDER OF THE FILE CONTROL
                DEY                             ; BLOCK AND THE TEMPORARY DIRECTORY AREA IN 'WORKSPC'! *************
                CPY        #FCBDEVN-1
                BNE        OWNRMOV
                LDA        D.HEAD               ; READ IN THE DIRECTORY HEADER FOR THIS FILE
                STA        BLOKNML
                LDA        D.HEAD+1
                STA        BLOKNMH
                LDA        D.DEV
                STA        DEVNUM
                JSR        RDGBUF               ; READ IT INTO THE GENERAL PURPOSE BUFFER
                BCS        FLSHERR              ; BRANCH IF ERROR.
                JSR        MOVHED0              ; MOVE HEADER INFO.
                LDA        D.ENTBLK             ; GET ADDRESS OF DIRECTORY BLOCK THAT
                LDY        D.ENTBLK+1           ; CONTAINS THE FILE ENTRY.
                CMP        D.HEAD               ; TEST TO SEE IF IT'S THE SAME BLOCK THAT
                BNE        FLSHEBLK             ; THE HEADER IS IN. BRANCH IF NOT.
                CPY        D.HEAD+1
                BEQ        FLUSH5               ; BRANCH IF HEADER BLOCK = ENTRY BLOCK.
FLSHEBLK        STA        BLOKNML
                STY        BLOKNMH
                JSR        RDGBUF               ; GET BLOCK WITH FILE ENTRY IN GENERAL BUFFER.
FLUSH5          JSR        ENTCALC              ; SET UP POINTER TO ENTRY
                JSR        MOVENTRY             ; MOVE ENTRY TO TEMP ENTRY BUFFER IN 'WORKSPC'
                LDY        #FCBUSE              ; UPDATE 'BLOCKS USED' COUNT.
                LDA        (FCBPTR),Y
                STA        DFIL+D.USAGE
                INY
                LDA        (FCBPTR),Y
                STA        DFIL+D.USAGE+1       ; HI BYTE TOO...
                LDY        #FCBEOF              ; AND MOVE IN END OF FILE MARK WHETHER
EOFUPDTE        LDA        (FCBPTR),Y           ; WE NEED TO OR NOT.
                STA        DFIL+D.EOF-FCBEOF,Y
                INY                             ; MOVE ALL THREE BYTES.
                CPY        #FCBEOF+3
                BNE        EOFUPDTE
                LDY        #FCBFRST             ; ALSO MOVE IN THE ADDRESS OF
                LDA        (FCBPTR),Y           ; THE FILE'S FIRST BLOCK SINCE
                INY                             ; IT MIGHT HAVE CHANGED SINCE THE FILE
                STA        DFIL+D.FRST          ; FIRST OPENED.
                LDA        (FCBPTR),Y
                STA        DFIL+D.FRST+1
                PAGE
                LDY        #FCBSTYP             ; AND THE LAST THING TO UPDATE IS
                LDA        (FCBPTR),Y           ; THE STORAGE TYPE.
                ASL        A                    ; (SHIFT IT INTO THE HI NIBBLE)
                ASL        A
                ASL        A
                ASL        A
                STA        SCRTCH
                LDA        DFIL+D.STOR          ; GET OLD TYPE BYTE (IT MIGHT BE THE SAME)
                AND        #$F                  ; STRIP OFF OLD TYPE
                ORA        SCRTCH               ; ADD IN THE NEW TYPE,
                STA        DFIL+D.STOR          ; AND PUT IT AWAY.
                JSR        DREVISE              ; GO UPDATE DIRECTORY!
                BCS        FLUSHERR
                LDY        #FCBDIRTY            ; MARK
                LDA        (FCBPTR),Y           ; FCB/DIRECTORY
                AND        #$FF-FCBMOD          ; AS
                STA        (FCBPTR),Y           ; UNDIRTY
                LDX        #0                   ; NOW CHECK TO SEE IF A BIT MAP
                LDA        D.DEV                ; IS LYING AROUND THAT SHOULD BE WRITTEN.
                CMP        BMADEV               ; IS IT IN MAP BUFFER A?
                BEQ        BMAPUP               ; YES, PUT IT ON THE DISK IF NECESSARY.
                LDX        #BMTABSZ             ; SET INDEX TO BIT MAP TABLE 'B'
                CMP        BMBDEV               ; NO, WHAT ABOUT BIT MAP BUFFER B?
                BNE        FLSHEND1             ; NOPE, ALL DONE.
BMAPUP          LDA        BMASTAT,X            ; TEST TO SEE IF IT'S BEEN MODIFIED.
                BPL        FLSHEND1             ; NOPE, ALL DONE AS I SAID.
                STX        BMTAB
                JSR        WRTBMAP              ; GO PUT IT AWAY.
                BCS        FLUSHERR
                LDX        BMTAB                ; MARK MAP AS UPDATED
                LDA        #0
                STA        BMASTAT,X
FLSHEND1        CLC
                RTS
FLUSHERR        EQU        *                    ; DROP INTO GLBERR
*
GLBERR          EQU        *                    ; REPORT ERROR IMMEDIATELY
* ONLY IF NOT A CLOSE ALL OR FLUSH ALL
                LDX        C.REFNUM
                BNE        GLBERR1              ; NOT AN 'ALL' SO REPORT NOW
                CLC
                STA        CFERR                ; SAVE FOR LATER
GLBERR1         RTS
*
*
GFCBADR         LDA        FCBANKNM             ; GET BANK THAT FCB IS IN
                STA        SISFCBP
                LDA        FCBADDRH             ; AND HIGH BYTE ADDRESS OF FILE CONTORL BLOCK.
                STA        FCBPTR+1
                RTS                             ; SILLY THAT IT'S SO SHORT...
*
SETERR          LDA        #ACCSERR
                SEC
EOFRETN         RTS
                PAGE
*
SETEOF          LDY        #FCBSTYP             ; ONLY KNOW HOW TO MOVE EOF OF TREE TYPE
                LDA        (FCBPTR),Y
                CMP        #TRETYP+1
                BCS        SETERR               ; BRANCH IF OTHER THAN TREE
                LDY        #FCBATTR             ; NOW CHECK TO INSURE WRITE IS ENABLED.
                LDA        (FCBPTR),Y
                AND        #WRITEN              ; CAN WE SET NEW EOF?
                BEQ        SETERR               ; NOPE, ACCESS ERROR.
                JSR        TSTWPROT             ; FIND OUT IF MOD IS POSIBLE (HARDWARE WRITE PROTECT)
                BCS        SETERR
                LDY        #FCBEOF+2            ; SAVE OLD EOF
                LDX        #2                   ; SO IT CAN BE SEEN
SETSAVE         LDA        (FCBPTR),Y           ; WHETHER BLOCKS NEED
                STA        OLDEOF,X             ; TO BE RELEASED
                DEY                             ; UPON
                DEX                             ; CONTRACTION
                BPL        SETSAVE              ; ALL THREE BYTES OF THE EOF
                JSR        ADJMARK              ; GET ADJUSTED END OF FILE ACCORDING TO 'C.BASE' INTO TPOS.
                BCS        EOFRETN              ; RETURN ANY ERROR IMMEDIATELY
                LDX        #2
NEOFPOS         LDA        TPOSLL,X             ; POSITION MARK TO NEW EOF
                STA        C.NEWEOF,X
                DEX
                BPL        NEOFPOS
                LDY        #FCBMARK+2           ; FIND OUT IF EOF < MARK.
                LDX        #2
NEOFTST         LDA        (FCBPTR),Y
                CMP        C.NEWEOF,X           ; COMPARE UNTIL NOT EQUAL OR CARRY CLEAR
                BCC        SETEOF1              ; BRANCH IF EOF>MARK
                BNE        SETEOF0              ; BRANCH IF EOF<MARK
                DEY
                DEX
                BPL        NEOFTST              ; LOOP ON ALL THREE BYTES
SETEOF0         JSR        RDPOSN               ; READ IN NEW POSITION.
                BCS        EOFRETN              ; RETURN ANY ERRORS.
SETEOF1         LDX        #2
                LDY        #FCBEOF+2            ; MOVE NEW EOF TO FCB.
SETEOF2         LDA        C.NEWEOF,X
                STA        (FCBPTR),Y
                DEY
                DEX
                BPL        SETEOF2              ; MOVE ALL THREE BYTES.
                JSR        FCBUSED              ; MARK FCB AS DIRTY (FOR FLUSH)
*
                LDX        #2                   ; POINT TO THIRD BYTE
PURTEST         LDA        OLDEOF,X             ; SEE IF EOF MOVED BACKWARDS
                CMP        C.NEWEOF,X           ; SO BLOCKS CAN
                BCC        PURTEST1             ; BE RELEASED (BRANCH IF NOT)
                BNE        PURGE                ; BRANCH IF BLOCKS TO BE RELEASED
                DEX
                BPL        PURTEST              ; ALL THREE BYTES
PURTEST1        JMP        FLSHEND1             ; NEW EOF NOT SMALLER
TRELEAS1        JMP        TRELEASE             ; OVERFLOW PREVENTER
*
PURGE           LDY        #FCBSTYP             ; FIND OUT WHAT TYPE OF TREE
                LDA        (FCBPTR),Y           ; TO PERFORM THE PROPER
                CMP        #SEEDTYP             ; STYLE OF BLOCK RELEASE
                BEQ        EOFOUT              ; SEED DON'T DEALLOCATE
                CMP        #TRETYP             ; FULL TREE?
                BEQ        TRELEAS1            ; BRANCH IF YES
*
* IF WE GET HERE, WE ARE RELEASING
* BLOCKS AT THE END OF A SAPLING FILE: CALCULATE CORRECT POSITION
* WITHIN THE INDEX BLOCK AND ALLOW SUBROUTINE
* PURGE LATTER BLOCKS TO DEALLOCATE
* ALL THE DATA BLOCKS THAT FOLLOW
*
                JSR        FNDBMAP             ; REFRESH THE RIGHT MAP FOR THIS VOLUME
                LDX        TPOSHI              ; PRELOAD
                LDY        TPOSLH              ;  THE THREE EOF
                LDA        TPOSLL              ;    BYTES
                BNE        PUR1                ; BRANCH IF NO BOUNDARY ADJUSTMENT NEEDED
                CPY        #0
                BNE        PUR2                ; MIDDLE BYTE ZERO MEANS NO CARRY
                CPX        #0                  ; ALL BYTES ZERO??
                BEQ        PUR1                ; BRANCH IF YES
                DEX
*
* THESE LINES IF CODE, SOMEWHAT CRYPTIC,
* CALCULATE THE POINT AT WHICH THE
* LAST BLOCK CONTAINING THE LAST BIT
* OF DATUM
*
* THE FOLLOWING IS ROUGHLY A /512
* ALGORITHM
*
PUR2            DEY
PUR1            TXA
                LSR        A
                TYA
                ROR        A
*
                JSR        PURLBLKS            ; MAKES A GOOD PTR TO DO THE RELEASING
                LDY        #FCBSTAT            ; MARK INDEX BLOCK
                LDA        (FCBPTR),Y          ; AS DIRTY
                ORA        #IDXMOD
                STA        (FCBPTR),Y
                LDA        PURUSE              ; INDICATE NEW NUMBER OF BLOCKS USED
                CLC
                ADC        #2                  ; ACCOUNT FOR CARDINAL AND INDEX
                LDY        #FCBUSE
                STA        (FCBPTR),Y          ; FILE LOW BYTE
                INY
                LDA        #0                  ; ANTICIPATE <257 BLOCKS
                BCC        PURHI
                LDA        #1                  ; >256 BLOCKS IN FILE
PURHI           STA        (FCBPTR),Y          ; HIGH BYTE BLOCKS USED
```

```
EOFOUT          CLC
                RTS                                 ; NO ERRORS POSSIBLE
*
PURLBLKS        EQU        *                        ; PURGE LATTER BLOCKS
* INPUT ARG: A REGISTER CONTAINING
* POINTER TO CURRENT DATA BLOCK WITHIN THE
* CURRENT INDEX BLOCK (TINDX)
* DEALLOCATE ALL LEGAL BLOCKS AFTER
* THE A REGISTER PTR. NO ERRORS POSSIBLE
*
                TAY                                 ; MAKE PROPER INDEX
                STY        PURUSE                   ; INDICATES NUMBER OF BLOCKS IN USE IN FILE
PURLOOP         INY                                 ; POINT TO A PTR TO DATA BLK TO DEALLOCATE
                BEQ        PURLRTS                  ; NO MORE BLOCKS IN INDEX
                INC        TINDX+1                  ; GET HIGH PART OF BLOCK ADDR
                LDA        (TINDX),Y
                TAX                                 ; X IS A PASSING PARM
                LDA        #0                       ; TELL INDEX BLOCK THAT THE DATA
                STA        (TINDX),Y                ; BLOCK IS NOW FREE
                TXA
                DEC        TINDX+1                  ; AND LOW PART
                ORA        (TINDX),Y
                BEQ        PURLOOP                  ; INDICATED ADDR WAS ZERO-ZERO
                LDA        (TINDX),Y                ; A REG IS ANOTHER PASSING PARM
                PHA
                LDA        #0
                STA        (TINDX),Y                ; AND SET LOW DATA ADDR AS FREED
                PLA
                STY        PURPLACE                 ; TEMP STORAGE
                JSR        DEALLOC                  ; DEALLOCATE BLOCK (ADDR: A (LOW), X ( HIGH)
                LDY        #VCBTFRE
                CLC
                LDA        (VCBPTR),Y               ; ADJUST NUMBER OF FREE BLOCKS ON VOLUME
                ADC        #1
                STA        (VCBPTR),Y
                INY
                LDA        (VCBPTR),Y               ; HIGH BYTE OF TOTAL FREE
                ADC        #0
                STA        (VCBPTR),Y
                LDY        PURPLACE
                JMP        PURLOOP
PURLRTS         RTS
PURUSE          DS         1                        ; CURRENT NUMBER OF BLOCKS USED
PURPLACE        DS         1                        ; CURRENT PLACE IN RELEASE-BLOCK CYCLE
TRELEASE        EQU        *
                JMP        EOFOUT                   ; RELEASE TWO LEVEL TREE CODE GOES HERE
*
GETEOF          LDY        #FCBEOF                  ; INDEX TO END OF FILE MARK
                LDX        #0                       ; WE'VE GOT INDIRECT BOTH WAYS (IN & OUT)
OUTEOF          LDA        (FCBPTR),Y
```

```
                STA        (C.OUTEOF,X)
                INY
                CPY        #FCBEOF+3
                BEQ        OFFRTS               ; BRANCH IF ALL THREE BYTES TRANSFERED.
                INC        C.OUTEOF             ; BUMP USER'S POINTER.
                BNE        OUTEOF
                INC        C.OUTEOF+1
                BNE        OUTEOF               ; BRANCH ALWAYS
*
                CHN        DESTROY,4,2

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: CLOSE.EOF
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.4of5.FOUR:SOS.DESTROY.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: DESTROY
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
*
NEWLINE         LDY        #FCBATTR             ; ADJUST NEWLINE STATUS FOR OPEN FILE.
                LDA        C.ISNEWL             ; ON OR OFF?
                BPL        OFFNEWL              ; BRANCH IF NEW LINE IS TO BE CLEARED.
                LDA        #NLINEN
                ORA        (FCBPTR),Y           ; SET NEW LINE BIT IN ATTRIBUTES
                STA        (FCBPTR),Y
                LDY        #FCBNEWL             ; AND MOVE IN NEW 'NEW-LINE' BYTE.
                LDA        C.NEWL
                STA        (FCBPTR),Y
                CLC
                RTS                             ; NO ERROR POSSIBLE.
*
OFFNEWL         LDA        #$FF-NLINEN
                AND        (FCBPTR),Y
                STA        (FCBPTR),Y           ; CLEAR NEW-LINE BIT.
OFFRTS          CLC                             ; THE NEW LINE CHARACTER DOES'T MATTER...
                RTS
                PAGE
*
GETINFO         JSR        FINDFILE             ; LOOK FOR FILE THEY WANT OT KNOW ABOUT.
                BCC        GTINFO1              ; BRANCH IF NO ERRORS.
                CMP        #BADPATH             ; WAS IT A ROOT DIRECTORY FILE?
                SEC                             ; (IN CASE OF NO MATCH)
                BNE        GINFOERR
                LDA        #$F0
                STA        DFIL+D.STOR          ; FOR GET INFO, REPORT PROPER STORAGE TYPE
                LDA        #0                   ; FORCE A COUNT OF FREE BLOCKS.
                STA        REQL
                STA        REQH
                JSR        TSFRBLK              ; (RETURNS IF IMMEDIATELY IF COUNT HAS PREVIOUSLY BEEN TAKEN)
                LDY        #VCBTFRE+1
                LDA        (VCBPTR),Y           ; RETURN TOTAL BLOCKS AND TOTAL IN USE.
                STA        REQH                 ; FIRST TRANSFER 'FREE' BLOCKS TO ZPAGE FOR LATER SUBTRACT
                DEY
                LDA        (VCBPTR),Y           ; TO DETERMINE THE 'USED' COUNT
                STA        REQL
                DEY
                LDA        (VCBPTR),Y           ; TRANSFER TO 'D.' TABLE AS AUX I.D.
                STA        DFIL+D.AUXID+1       ; (TOTAL BLOCK COUNT IS CONSIDERED AUX I.D. FOR THE VOLUME)
                TAX
                DEY
                LDA        (VCBPTR),Y
                STA        DFIL+D.AUXID
                SEC                             ; NOW SUBTRACT AND REPORT THE NUMBER OF BLOCKS 'IN USE'
                SBC        REQL
                STA        DFIL+D.USAGE
                TXA
                SBC        REQH
                STA        DFIL+D.USAGE+1
GTINFO1         LDY        #0                   ; TRANSFER BYTES FROM THERE INTERNAL ORDER TO CALL SPEC VIA 'INFTABL'
TRANSLATION
GTINFO2         LDA        INFTABL,Y
                BPL        GTINFO3              ; BRANCH IF THIS IS DATA IS VALID AS IS.
                AND        #$7F                 ; IS THIS THE 4TH BYTE OF THE EOF PARAMETER?
                BEQ        GTINFO4              ; YES, AND IT'S ALWAYS A ZERO.
                CMP        #D.STOR+1            ; IS THIS THE STORAGE TYPE BYTE?
                BNE        GINFOEND             ; NO, IT'S THE END OF INFO THAT CAN BE RETURNED.
                LDA        DFIL+D.STOR          ; GET STORAGE TYPE
                LSR        A
                LSR        A
                LSR        A
                LSR        A                    ; MAKE IT A VALUE 1-$F BY SHIFTING OUT FILE NAME LENGTH.
                BPL        GTINFO4              ; BRANCH ALWAYS
*
GTINFO3         TAX                             ; USE AS OFFSET INTO 'D.' TABLE.
                LDA        DFIL,X
GTINFO4         STA        (C.FILIST),Y         ; PASS TO USER'S BUFFER
                INY
                CPY        C.FILSTLN            ; HAS REQUEST BEEN FILLED?
                BNE        GTINFO2              ; NO, PASS NEXT
GINFOEND        CLC                             ; INDICATE NO ERRORS
GINFOERR        RTS
*
*
                PAGE
*
SETINFO         JSR        FINDFILE             ; FIND WHAT USER WANTS...
                BCS        SINFOERR             ; RETURN ANY FAILURE.
                LDA        C.FILSTLN            ; TEST FOR NUL CHANGE
                BEQ        SINFEND              ; BRANCH IF NOTHING TO CHANGE.
                LDY        #0                   ; INIT POINTER TO USER SUPPLIED LIST.
                LDA        (C.FILIST),Y         ; FETCH FILE ATTRIBUTES
                AND        #$1C                 ; FORBIDDEN BITS? <SRS 82.162>
                BEQ        SETINF1              ; NO
                LDA        #ACCSERR             ; YES
                SEC
                RTS                             ; RETURN AN ERROR
SETINF1         LDA        BACKMASK             ; GET CURRENT BACKMASK <SRS 82.162>
```

```
* BACKUP KNOWS HOW TO RESET THIS BIT. <SRS 82.162>
                STA        BKBITFLG             ; BIT (USED BY DREVISE)
SETINF1X        LDX        INFTABL,Y            ; GET INDEX INTO CORESPONDING 'D.' TABLE
                BMI        SETINF2              ; BRANCH IF WE'VE REACHED STORAGE TYPE PARAMETER
                LDA        (C.FILIST),Y
                STA        DFIL,X
                INY                             ; HAS USER'S REQUEST BEEN SATISFIED?
                CPY        C.FILSTLN
                BNE        SETINF1X             ; NO, MOVE NEXT BYTE.
SINFEND         JMP        DREVISE              ; GO UPDATE DIRECTORY WITH CURRENT TIME.
*
SETINF2         LDY        C.FILSTLN            ; TEST TO SEE IF USER WANTS HIS TIME STAMP ADDED
                CPY        #$F                  ; (LIST MUST BE AT LEAST $F BYTES LONG)
                BCC        SINFEND              ; NO PUT CURRENT TIME INSTEAD.
                LDY        #$B                  ; MOVE IN THE NEXT GROUP OF BYTES
SETINF3         LDX        INFTABL,Y
                BMI        SINFEND1
                LDA        (C.FILIST),Y
                STA        DFIL,X
                INY
                CPY        C.FILSTLN            ; SATISFACTION YET?
                BNE        SETINF3              ; NOPE, KEEP EM PUMPIN'
SINFEND1        JMP        DREVISE1
*
BKBITFLG        DS         1                    ; FOR TURNING OFF BACKUP BIT
*
*
INFTABL         DFB        D.ATTR,D.FILID,D.AUXID,D.AUXID+1
                DFB        D.STOR+1+$80,D.EOF,D.EOF+1,D.EOF+2 ; (D.STOR=0 THUS D.STOR+1 WAS NECESSARY)
                DFB        $80,D.USAGE,D.USAGE+1,D.MODDT ; (THE $80 IS FOR THE FOURTH BYTE OF EOF)
                DFB        D.MODDT+1,D.MODTM,D.MODTM+1,$FF ; TABLE ALWAYS ENDS IN $FF
                PAGE
*
RENAME          JSR        LOOKFILE             ; LOOK FOR SOURCE (ORIGINAL) FILE.
                BCC        RNAME0               ; BRANCH IF FOUND.
                CMP        #BADPATH             ; TRYING TO RENAME A VOLUME?
                BNE        RNAMERR              ; NO, RETURN OTHER ERROR.
                JSR        RENPATH              ; SYNTAX NEW NAME.
                BCS        RNAMERR
                LDA        WRKPATH              ; FIND OUT IF ONLY ROOTNAME FOR NEW NAME
                CMP        PATHNML
                BNE        RNBADPTH             ; NOT SINGLE NAME, RETURN ERROR!
                LDY        #VCBSTAT             ; TEST FOR OPEN FILES BEFORE CHANGING
                LDA        (VCBPTR),Y
                BPL        RNAMEVOL             ; BRANCH IF VOLUME NOT BUSY
                LDA        #FILBUSY
SINFOERR        EQU        *
                RTS                             ; (CARRY IS SET)
RNAMEVOL        LDY        #0                   ; GET NEWNAME'S LENGTH.
                LDA        (WRKPATH),Y
                TAY
                ORA        #$F0                 ; (ROOT FILE STORAGE TYPE)
                JSR        MVROTNAM             ; UPDATE ROOT DIRECTORY.
                BCS        RNAMERR
                LDY        #0
                LDA        (WRKPATH),Y          ; UPDATE VCB ALSO.
                TAY
RNMEVOL         LDA        (WRKPATH),Y
                STA        (VCBPTR),Y
                DEY
                BPL        RNMEVOL
                CLC
                RTS
*
RNAME0          JSR        RENPATH              ; SET UP AND SYNTAX NEW NAME.
                BCS        RNAMERR
                LDY        #0                   ; VERIFY THAT BOTH NAMES HAVE SAME ROOT.
                LDA        (PATHNML),Y
                TAY
TSTSMROT        LDA        (PATHNML),Y          ; COMPARE NEWNAME'S ROOT NAME WITH
                CMP        (VCBPTR),Y           ; OLD NAME'S VOLUME NAME.
                BNE        RNBADPTH             ; RETURN 'BADPATH' IF NOT SAME VOLUME.
                DEY
                BPL        TSTSMROT             ; (TEST SAME 'ROT')
                JSR        LOOKFILE             ; TEST FOR DUPLICATE FILE NAME.
                BCS        TSTFNF1              ; BRANCH IF ERROR TO TEST FOR FILE NOT FOUND.
                LDA        #DUPERR              ; TELL USER THAT NEW NAME ALREADY EXISTS.
RNAMERR         SEC
                RTS
                PAGE
TSTFNF1         CMP        #FNFERR              ; WAS IT A VALID FILE NOT FOUND?
                BNE        RNAMERR              ; NO, RETURN OTHER ERROR CODE.
                LDX        #2                   ; NOW MOVE NEW NAME'S OWNERSHIP (DIRECTORY HEADER) I.D.
SVENEWID        LDA        D.DEV,X              ; THIS CONSISTS OF THE UNIT NUMBER,
                STA        NPATHDEV,X           ; AND THE ADDRESS OF THE DIRECTORY THE FILE
                DEX                             ; WASN'T FOUND IN. LOGIC BY NEGATION...
                BPL        SVENEWID
                JSR        SETPATH              ; NOW SYNTAX THE PATHNAME OF THE FILE TO BE CHANGED.
                BCS        RNAMERR
                JSR        FINDFILE             ; GET ALL THE INFO ON THIS ONE.
                BCS        RNAMERR
                JSR        TSTOPEN              ; DON'T ALLOW RENAME TO OCCUR IF FILE IS IN USE.
                LDA        #FILBUSY             ; ANTICIPATE ERROR
                BCS        RNAMERR
                LDA        DFIL+D.ATTR          ; TEST BIT THAT SAYS IT'S OK TO RENAME
                AND        #RENAMEN
                BNE        RNAME1               ; BRANCH IF IT'S ALRIGHT TO RENAME.
                LDA        #ACCSERR             ; OTHERWISE REPORT ILLEGAL ACCESS.
                SEC
                RTS
```

```
*
RNAME1          LDX        #2                    ; NOW TEST TO SEE IF NEW PATHNAME FITS IN THE
SAMOWNR         LDA        D.DEV,X               ; SAME DIRECTORY FILE.
                CMP        NPATHDEV,X
                BEQ        RNAME2
RNBADPTH        LDA        #BADPATH              ; TELL USER THAT PATHNAMES INCOMPATABLE.
                SEC
                RTS
*
RNAME2          DEX                              ; TEST ALL THREE BYTES.
                BPL        SAMOWNR
                JSR        RENPATH               ; WELL... SINCE BOTH NAMES WOULD GO INTO THE
                BCS        RNAMERR               ; DIRECTORY, RE-SYNTAX THE NEW NAME TO GET LOCAL NAME ADDRESS.
                TYA                              ; (Y CONTAINS THE LOCAL NAME LENGTH+1)
                BEQ        RNBADPTH              ; REPORT ERROR IF LENGTH INFO NOT IMMEDIATELY AVAILABLE.
                DEY                              ; (REMOVE THE +1)
RNAME3          LDA        (WRKPATH),Y           ; MOVE LOCAL NAME TO DIR ENTRY WORKSPACE.
                STA        DFIL+D.STOR,Y
                DEY
                BNE        RNAME3
                LDA        DFIL+D.STOR           ; PRESERVE FILE STORAGE TYPE.
                AND        #$F0                  ; STRIP OFF OLD NAME LENGTH.
                TAX
                ORA        (WRKPATH),Y           ; ADD IN NEW NAME'S LENGTH
                STA        DFIL+D.STOR
                CPX        #DIRTYP*16            ; THAT FILE MUST BE CHANGED ALSO.
                BNE        RNAMDONE              ; BRANCH IF NOT DIRECTORY TYPE.
                PAGE
                LDA        DFIL+D.FRST           ; READ IN FIRST (HEADER) BLOCK OF SUB DIRECTORY
                STA        BLOKNML
                LDA        DFIL+D.FRST+1
                STA        BLOKNMH
                JSR        RDGBUF
                BCS        RNAMERR               ; REPORT ERRORS
                LDY        #0                    ; CHANGE THE HEADER'S NAME TO MATCH THE OWNER'S NEW NAME.
                LDA        (WRKPATH),Y           ; GET LOCAL NAME LENGTH AGAIN
                TAY
                ORA        #HEDTYP*16            ; ASSUME IT'S A HEADER.
                JSR        MVROTNAM
                BCS        RNAMERR
RNAMDONE        JMP        DREVISE1              ; END BY UPDATING ALL PATH DIRECTORIES
*
*
MVROTNAM        STA        GBUF+4
MVHEDNAM        LDA        (WRKPATH),Y
                STA        GBUF+4,Y
                DEY
                BNE        MVHEDNAM
                JMP        WRTGBUF               ; WRITE CHANGED HEADER BLOCK.
*
```

```
*
RENPATH         LDA        C.NWPATH             ; GET ADDRESS TO NEW PATHNAME.
                STA        TPATH
                LDA        C.NWPATH+1           ; SET UP FOR SYNTAXING ROUTINE (SYNPATH).
                STA        TPATH+1
                LDA        SSNWPATH             ; (MOVE BYTE FOR SISTER PAGE, TOO.)
                STA        SISTPATH
                JMP        SYNPATH              ; GO SYNTAX IT. (RETURNS LAST LOCAL NAME LENGTH IN Y).
*
*
DEALBLK         LDY        #0                   ; BEGIN AT THE BEGINNING.
DALBLK1         STY        SAPTR                ; SAVE CURRENT INDEX.
                LDA        GBUF,Y               ; GET ADDRESS (LOW) OF BLOCK TO BE DEALLOCATED.
                CMP        GBUF+$100,Y          ; TEST FOR NUL BLOCK.
                BNE        DALBLK2              ; BRANCH IF NOT NUL.
                CMP        #0
                BEQ        DALBLK3              ; SKIP IT IF NUL.
DALBLK2         LDX        GBUF+$100,Y          ; GET THE REST OF THE BLOCK ADDRESS.
                JSR        DEALLOC              ; FREE IT UP ON VOLUME BIT MAP.
                BCS        DALBLKERR            ; RETURN ANY ERROR.
                LDY        SAPTR                ; GET INDEX TO SAPLING LEVEL INDEX BLOCK AGAIN.
DALBLK3         INY                             ; POINT AT NEXT BLOCK ADDRESS.
                BNE        DALBLK1              ; BRANCH IF MORE TO DEALLOCATE (OR TEST).
                CLC                             ; INDICATE NO ERROR.
DALBLKERR       RTS
*
*
                PAGE
*
DESTROY         JSR        FINDFILE             ; LOOK FOR FILE TO BE WIPED OUT.
                BCS        DESTERR              ; PASS BACK ANY ERROR.
                JSR        TSTOPEN              ; IS THIS FILE OPEN?
                LDA        TOTENT
                BEQ        DSTROY1              ; BRANCH IF FILE NOT OPEN.
                LDA        #FILBUSY
                SEC                             ; INFORM USER THAT FILE CAN'T BE DESTORYED AT THIS TIME.
                RTS
*
DSTROY1         LDA        #0                   ; FORCE PROPER FREE COUNT IN VOLUME.
                STA        REQL                 ; (NO DISK ACCESS OCCURS IF ALREADY PROPER)
                STA        REQH
                JSR        TSFRBLK
                BCC        DSTROY2
                CMP        #OVRERR              ; WAS IT JUST A FULL DISK?
                SEC
                BNE        DESTERR              ; NOPE, REPORT ERROR.
*
DSTROY2         LDA        DFIL+D.ATTR          ; MAKE SURE IT'S OK TO DESTROY THIS FILE.
                AND        #DSTROYEN
                BNE        DSTROY3              ; BRANCH IF OK.
                LDA        #ACCSERR             ; TELL USER IT'S NOT KOSHER.
                JSR        SYSERR               ; (RETURNS TO CALLER OF DESTORY)
*
DSTROY3         JSR        TWRPROT1             ; BEFORE GOING THRU DEALLOCATION,
                BCS        DESTERR              ; TEST FOR WRITE PROTECTED HARDWARE.
                LDA        DFIL+D.STOR          ; FIND OUT WHICH STORAGE TYPE.
                AND        #$F0                 ; STRIP OFF NAME LENGTH.
                CMP        #TRETYP+1*$10        ; IS IT A SEED, SAPLING, OR TREE?
                BCC        DSTREE               ; BRANCH IF IT IS.
                JMP        DSTDIR               ; OTHERWISE TEST FOR DIRECTORY DESTROY.
*
DSTREE          JSR        GTTINDX              ; GET A BIT MAP BUFFER AND TEMPORARY INDEX BUFFER.
                BCS        DESTERR
                LDA        DFIL+D.STOR          ; GET STORAGE TYPE AGAIN
                AND        #$F0
                CMP        #TRETYP*$10          ; IS THIS A TREE (FULL 2-LEVEL)?
                BNE        DSTSAP               ; NO, TEST FOR SAPLING.
                JSR        RDFRST               ; READ IN ROOT INDEX FOR THIS FILE.
                BCC        DSTRE2               ; BRANCH IF ALL IS WELL.
DESTERR         RTS                             ; OTHERWISE RETURN ERROR.
*
DSTSAP          CMP        #SAPTYP*$10          ; IS IT A SAPLING
                BNE        DSTLAST              ; NO, JUST DEALLOCATE FIRST (AND ONLY) BLOCK.
                JSR        ZTMPIDX              ; CLEAR OUT TEMPORARY INDEX BUFFER.
                LDA        DFIL+D.FRST          ; MAKE THIS SAP LOOK LIKE A TREE...
                LDY        #0                   ; THIS IS DONE BY PLACING THE FIRST BLOCK ADDRESS
                STA        (TINDX),Y            ; IN THE TEMP (TOP) INDEX BUFFER AS
                INC        TINDX+1
                LDA        DFIL+D.FRST+1        ; A SUB INDEX WOULD APPEAR.
                STA        (TINDX),Y
                DEC        TINDX+1
DSTRE2          LDY        #0                   ; BEGIN SCAN OF TOP LEVEL INDEX AT ZERO.
DSTNXT          STY        TREPTR               ; SAVE POINTER TO TREE LEVEL.
                LDA        (TINDX),Y            ; GET BLOCK ADDRESS OF A SUB INDEX BLOCK
                INC        TINDX+1              ; (TEST FOR NUL BLOCK)
                CMP        (TINDX),Y
                BNE        DSTRE3               ; BRANCH IF WE'VE GOT AN BLOCK TO DEALLOCATE.
                CMP        #0                   ; IS ENTIRE ADDRESS ZERO?
                BEQ        DSTRE4               ; YES, DO NEXT. (CARRY SET)
DSTRE3          CLC                             ; INDICATE THERE IS A BLOCK OF INDEXES TO FREE UP.
                STA        BLOKNML
                LDA        (TINDX),Y            ; GET HI ADDRESS TOO.
                STA        BLOKNMH
DSTRE4          DEC        TINDX+1              ; (RESTORE PROPER ADDRESS FOR BUFFER)
                BCS        DSTNXT1              ; BRANCH IF NO SUB INDEX.
                JSR        RDGBUF               ; USE GENERAL BUFFER FOR SUB INDEX BUFFER.
                BCS        DESTERR
                JSR        DEALBLK              ; GO FREE UP BLOCKS IN SUB INDEX
                BCS        DESTERR
                LDY        TREPTR               ; AND FREE UP SUB INDEX BLOCK TOO.
                INC        TINDX+1
                LDA        (TINDX),Y
                TAX
                DEC        TINDX+1
                LDA        (TINDX),Y
                JSR        DEALLOC
                BCS        DESTERR
                LDY        TREPTR
DSTNXT1         INY                           ; HAVE ALL SUB INDEXES BEEN LOCATED?
                BNE        DSTNXT             ; NO, DO NEXT...
DSTLAST         LDA        DFIL+D.FRST        ; DEALLOCATE FIRST BLCOK OF FILE.
                LDX        DFIL+D.FRST+1
                JSR        DEALLOC
                BCS        DESTERR
                LDA        #0                 ; UPDATE DIRECTORY TO FREE ENTRY SPACE.
                STA        DFIL+D.STOR
                CMP        H.FCNT             ; FILE ENTRY WRAP?
                BNE        DST1               ; BRANCH IF NO CARRY ADJUSTMENT
                DEC        H.FCNT+1           ; TAKE CARRY FROM HIGH BYTE OF FILE ENTRIES
DST1            DEC        H.FCNT             ; MARK HEADER WITH ONE LESS FILE
                LDX        BMTAB              ; UPDATE (LAST) BITMAP.
                JSR        BMAPUP
                BCS        DESTERR
                LDY        #VCBTFRE
                LDA        DFIL+D.USAGE
                ADC        (VCBPTR),Y
                STA        (VCBPTR),Y         ; UPDATE CURRENT FREE BLOCK COUNT.
                INY
                LDA        DFIL+D.USAGE+1
                ADC        (VCBPTR),Y
                STA        (VCBPTR),Y
                LDA        #0                 ; FORCE RESCAN FROM FIRST BITMAP
                LDY        #VCBCMAP
                STA        (VCBPTR),Y
                JMP        DREVISE            ; UPDATE DIRECTORY LAST...
*
                PAGE
*
DSTDIR          CMP        #DIRTYP*16         ; IS THIS A DIRECTORY FILE?
                BEQ        DSDIR1             ; YES, PROCEED.
                LDA        #CPTERR            ; FILE IS NOT COMPATABLE.
                JSR        SYSERR             ; GIVE UP.
*
DSDIR1          JSR        FNDBMAP            ; MAKE SURE A BUFFER IS AVAILABLE FOR THE BITMAP.
                BCS        DSDIRERR
                LDA        DFIL+D.FRST        ; READ IN FIRST BLOCK OF DIRECTORY INTO GBUF.
                STA        BLOKNML
                LDA        DFIL+D.FRST+1
                STA        BLOKNMH
                JSR        RDGBUF
                BCS        DSDIRERR
                LDA        GBUF+HCENT+4         ; FIND OUT IF ANY FILES EXIST ON THIS DIRECTORY.
                BNE        DSDIRACC             ; BRANCH IF ANY EXIST.
                LDA        GBUF+HCENT+5
                BEQ        DSDIR2
DSDIRACC        LDA        #ACCSERR
                JSR        SYSERR
*
DSDIR2          LDA        GBUF+2               ; GET FORWARD LINK.
                CMP        GBUF+3               ; TEST FOR NO LINK.
                BNE        DSDIR3
                CMP        #0
                BEQ        DSTLAST              ; IF NO LINK, THEN FINISHED.
DSDIR3          LDX        GBUF+3
                JSR        DEALLOC              ; FREE THIS BLOCK.
                BCS        DSDIRERR
                LDA        GBUF+2
                STA        BLOKNML
                LDA        GBUF+3
                STA        BLOKNMH              ; READ IN LINKED BLOCK.
                JSR        RDGBUF
                BCC        DSDIR2               ; LOOP UNTIL ALL ARE FREED.
DSDIRERR        RTS
*
*
                PAGE
WORKSPC         EQU        *
V.STATUS        DS         1                    ; VOLUME STATUS, INCLUDES 'ACTIVE' IN BIT 7
H.CREDT         DS         2                    ; DIRECTORY CREATION DATE
                DS         2                    ; DIRECTORY CREATION TIME
                DS         1                    ; VERSION UNDER WHICH THIS DIRECTORY WAS CREATED
                DS         1                    ; EARLIEST VERSION THAT IT'S COMPATABLE WITH
H.ATTR          DS         1                    ; ATTRIBUTES (PROTECT BIT, ETC.)
H.ENTLN         DS         1                    ; LENGTH OF EACH ENTRY IN THIS DIRECTORY.
H.MAXENT        DS         1                    ; MAXIMUM NUMBER OF ENTRIES PER BLOCK
H.FCNT          DS         2                    ; CURRENT NUMBER OF FILES IN THIS DIRECTORY
                DS         2                    ; ADDRESS OF FIRST ALLOCATION BIT MAP
                DS         2                    ; TOTAL NUMBER OF BLOCKS ON THIS UNIT
                DS         5                    ; (FOR FUTURE EXPANSION)
*
D.DEV           DS         1                    ; DEVICE NUMBER OF THIS DIRECTORY ENTRY
D.HEAD          DS         2                    ; ADDRESS OF <SUB> DIRECTORY HEADER
D.ENTBLK        DS         2                    ; ADDRESS OF BLOCK WHICH CONTAINS THIS ENTRY
D.ENTNUM        DS         1                    ; ENTRY NUMBER WITHIN BLOCK.
DFIL            EQU        *
D.STOR          EQU        *-DFIL               ; STORAGE TYPE * 16 + FILE NAME LENGTH
                DS         1
; *-DFIL ; FILE NAME
                DS         15
D.FILID         EQU        *-DFIL               ; USER'S IDENTIFICATION BYTE
```

```
                DS         1
D.FRST          EQU        *-DFIL               ; FIRST BLOCK OF FILE
                DS         2
D.USAGE         EQU        *-DFIL               ; NUMBER OF BLOCKS CURRENTLY ALLOCATED TO THIS FILE
                DS         2
D.EOF           EQU        *-DFIL               ; CURRENT END OF FILE MARKER
                DS         3
D.CREDT         EQU        *-DFIL               ; DATE OF FILE'S CREATION
                DS         2
; *-DFIL ; TIME OF FILE'S CREATION
                DS         2
;  EQU *-DFIL ; SOS VERSION THAT CREATED THIS FILE
                DS         1
D.COMP          EQU        *-DFIL               ; BACKWARD VERSION COMPATABILTY
                DS         1
D.ATTR          EQU        *-DFIL               ; 'PROTECT', READ/WRITE 'ENABLE' ETC.
                DS         1
D.AUXID         EQU        *-DFIL               ; USER AUXILLARY IDENTIFACATION
                DS         2
D.MODDT         EQU        *-DFIL               ; FILE'S LAST MODIFICATION DATE
                DS         2
D.MODTM         EQU        *-DFIL               ; FILE'S LAST MODIFICATION TIME
                DS         2
D.DHDR          EQU        *-DFIL               ; HEADER BLOCK ADDRESS OF FILE'S DIRECTORY
                DS         2
*
CMDADR          DS         2
SCRTCH          DS         13                   ; SCRATCH AREA FOR ALLOCATION ADDRESS CONVERSION
OLDEOF          DS         3                    ; TEMP USED IN W/R
OLDMARK         DS         3                    ; USED BY 'RDPOSN' AND 'WRITE'
SCRHIGH         EQU        <SCRTCH              ; AND DEVICE NUMBERS FROM BOB'S CODE.
*
                CHN        SWAPOUT/IN,4,2

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DESTROY
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.4of5.FOUR:SOS.POSN.OPEN.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: POSN.OPEN
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
GETMARK         LDY        #FCBMARK             ; MOVE CURRENT POSITION MARKER TO
GMARK1          LDA        (FCBPTR),Y           ; USER'S 4 BYTE BUFFER POINTED TO BY
                PHA                             ; C.MRKPTR IN SOS ZPAGE
                INY
                CPY        #FCBMARK+3           ; USE STACK AS TEMPORARY STORAGE FOR THREE BYTE
                BNE        GMARK1               ; POSITION VALUE.
                LDA        #0                   ; THE FOURTH (HIGHEST ORDER) BYTE IS ALWAYS ZERO.
                LDY        #3
                PHA
MOVMRK          PLA
                STA        (C.MRKPTR),Y         ; MOVE TO USER'S SPACE
                DEY                             ; IS THERE ANOTHER TO PULL FROM STACK?
                BPL        MOVMRK               ; YES, GET NEXT LOWER BYTE FROM STACK.
                CLC                             ; INDICATE NO ERROR.
                RTS
*
SETMARK         JSR        ADJMARK              ; MAKE ADJUSTMENTS TO REQUESTED MARK ACCORDING TO BASE.
                BCC        SMARK1               ; BRANCH IF ADJUSTMENT WAS VALID.
                RTS
SMARK1          LDX        #2                   ; NOW COMPARE END OF FILE WITH NEW
                LDY        #FCBEOF+2            ; POSITION TO BE SURE IT'S WITHIN
CMPEOF          LDA        TPOSLL,X             ; THE BOUNDS OF CURRENTLY DEFINED
                CMP        (FCBPTR),Y           ; LIMITS.
                BCC        CKSAMBLK             ; BRANCH IF MARK<EOF
                BNE        ERRMEOF              ; RETURN ERROR IF MARK>= EOF
                DEY
                DEX
                BPL        CMPEOF
                BMI        CKSAMBLK             ; BRANCH ALWAYS
ERRMEOF         LDA        #POSNERR             ; TELL USER MARK IS OUT OF RANGE.
                RTS                             ; (CARRY IS SET TO INDICATE ERROR)
*
ADJMARK         LDA        C.MARK+3             ; MAKE SURE FOURTH BYTE OF DISPLACE IS ZIP
                BNE        ERRPOSN              ; BRANCH TO ERR IF NOT
                LDX        #$FD                 ; ANTICIPATE OTHER THAN BASE OF ZERO
                LDY        #FCBMARK             ; FURTHER ASSUME IT'S A BASE OFFSET FROM CURRENT POSITION
                LDA        C.BASE               ; NOW FIND OUT WHAT IT REALLY IS.
                LSR        A                    ; (CARRY SET=SUBTRACT, NON ZERO REMAINDER= OFFSET FROM EOF)
                BCS        SUBMARK
                BEQ        ADJMRK               ; BRANCH IF MARK IS FROM BEGINNING OF FILE
ADDPOSN         LDA        (FCBPTR),Y           ; ADD USER QUANTITY TO CURRENT
                ADC        C.MARK+3,X           ; POSITION TO FORM NEW POSITION.
                STA        >TPOSLL-$FD,X        ; (NOTE: ZERO PAGE REFERENCE WRAPS AROUND IN Z-PAGE)
                INY
                INX
                BNE        ADDPOSN              ; ADD ALL THREE BYTES
                BCS        ERRPOSN              ; BRANCH IF OVERFLOW
                BEQ        ADJMRK1              ; BRANCH ALWAYS
*
                PAGE
SUBMARK         BNE        SUBPOSN              ; BRANCH IF IT'S AN OFFSET FROM CURRENT POSITION
                LDY        #FCBEOF              ; OTHERWISE ASSUME OFFSET FROM END OF FILE.
SUBPOSN         LDA        (FCBPTR),Y           ; SUBTRACT USER QUANTITY TO FORM
                SBC        C.MARK+3,X           ; NEW POSITION. IF FINAL
                STA        >TPOSLL-$FD,X        ; RESULT IS L.T. ZERO, THEN REPORT
                INY                             ; POSITION ERROR...
                INX
                BNE        SUBPOSN
                BCS        ADJMRK1              ; BRANCH IF LEGAL POSITION CALCULATED.
ERRPOSN         LDA        #POSNERR
                SEC                             ; INDICATE ERROR
                RTS
*
ADJMRK          LDX        #2                   ; FIRST SET UP POSITION TEMPS USED
ADJMRK0         LDA        C.MARK,X             ; BY BOTH POSITION ROUTINES
                STA        TPOSLL,X
                DEX
                BPL        ADJMRK0
ADJMRK1         CLC                             ; NO ERRORS
                RTS
*
*
RDPOSN          EQU        *
CKSAMBLK        EQU        *
                LDY        #FCBMARK+1           ; FIRST TEST TO SEE IF NEW POSITION IS
                LDA        (FCBPTR),Y           ; WITHIN THE SAME (CURRENT) DATA BLOCK.
                AND        #$FE
                STA        SCRTCH
                INY                             ; BUMP TO ACCESS HIGHEST ORDER ADDRESS BYTE
                LDA        TPOSLH               ; GET MIDDLE BYTE OF NEW POSITION
                SEC
                SBC        SCRTCH
                STA        SCRTCH
                BCC        TYPMARK              ; BRANCH IF POSSIBLY L.T. CURRENT POSITION
                CMP        #2                   ; MUST BE WITHIN 512 BYTES OF BEGINNING OF CURRENT
                BCS        TYPMARK
                LDA        TPOSHI               ; NOW MAKE SURE WERE TALKIN ABOUT
                CMP        (FCBPTR),Y           ; THE SAME 64K CHUNK!
                BNE        TYPMARK              ; BRANCH IF WE AREN'T.
                JMP        SVMARK               ; IF WE IS, ADJUST FCB AND POSPTR AND RETURN.
*
TYPMARK         LDY        #FCBSTYP             ; NOW FIND OUT WHICH TYPE
                LDA        (FCBPTR),Y           ; OF FILE WE'RE POSITIONING ON.
                BEQ        FERRTYP              ; THERE IS NO SUCH TYPE AS ZERO, BRANCH NEVER!
                CMP        #4                   ; IS IT A TREE CLASS FILE?
                BCC        CHKDSKSW             ; YES, GO POSITION
                JMP        DIRMARK              ; NO, TEST FOR DIRECTORY TYPE.
*
CHKDSKSW        EQU        *                    ; MAKE SURE S/HE HASN'T MOVED THE VOLUME
                LDY        #FCBDEVN
                LDA        (FCBPTR),Y
                STA        DEVNUM               ; MAKE SURE DEVICE NUMBER PARM IS CURRENT
                JSR        TWRPROT1             ; PASSES DEVNUM (CHECK DISK SWITCH)
                LDA        DSWGLOB              ; DISK SWITCH GLOBAL
                BEQ        TREPOS               ; BRANCH IF NONE DETECTED
CHKDSKS1        JSR        VERFYVOL             ; MATCHES VCBPTR VS. DEVNUM
                BCC        TREPOS               ; BRANCH IF DISK HASN'T SWITCHED
                JSR        USRREQ               ; POLITELY ASK USER TO MOUNT
                BCC        CHKDSKS1             ; SAID HE DID, CHECK AGAIN
                LDA        #VNFERR              ; REFUSES TO MOUNT
                RTS
*
FERRTYP         LDY        #FCBREFN             ; CLEAR ILLEGALLY TYPED FCB ENTRY
                STA        (FCBPTR),Y
                LDA        #BADREFNUM           ; TELL EM THERE IS NO SUCH FILE
                SEC
                RTS
*
                PAGE
TREPOS          LDY        #FCBSTYP             ; USE STORAGE TYPE AS NUMBER
                LDA        (FCBPTR),Y           ; OF LEVELS (SINCE 1=SEED, 2=SAPLING, AND 3=TREE)
                STA        LEVELS
                LDY        #FCBSTAT             ; SINCE IT'S A DIFFERENT DATA
                LDA        (FCBPTR),Y           ; BLOCK, MUST NOT FORGET PREVIOUS DATA.
                AND        #DATMOD              ; THEREFORE, SEE IF PREVIOUS DATA WAS MODIFIED
                BEQ        POSNEW1              ; THEN DISK MUST BE UPDATED.
                JSR        WFCBDAT              ; GO WRITE CURRENT DATA BLOCK.
                BCS        POSERR               ; RETURN ANY ERROR ENCOUNTERED.
*
POSNEW1         LDY        #FCBMARK+2           ; TEST TO SEE IF CURRENT
                LDA        (FCBPTR),Y           ; INDEX BLOCK IS GOING TO BE USABLE...
                AND        #$FE                 ; OR IN OTHER WORDS-
                STA        SCRTCH               ; IS NEW POSITION WITHIN 128K OF THE BEGINNING
                LDA        TPOSHI               ; OF CURRENT SAPLING LEVEL CHUNK.
                SEC
                SBC        SCRTCH
                BCC        POSNEW2              ; BRANCH IF A NEW INDEX BLOCK IS ALSO NEEDED
                CMP        #2                   ; NEW POSITION IS > THAN BEGINING OF OLD. IS IT WITHIN 128K?
                BCS        POSNEW2              ; BRANCH IF NOT.
                LDX        LEVELS               ; IS THE FILE WE'RE DEALING WITH A SEED?
                DEX
                BNE        DATLEVEL             ; NO, USE CURRENT INDEXES.
TSTINY          LDA        TPOSLH               ; IS NEW POSITION UNDER 512?
                LSR        A
                ORA        TPOSHI
                BNE        NOIDXDAT             ; NO, MARK BOTH DATA AND INDEX BLOCK AS UN-ALLOCATED.
                LDY        #FCBFRST
                LDA        (FCBPTR),Y           ; FIRST BLOCK IS ONLY BLOCK AND IT'S DATA!
                STA        BLOKNML
                INY
                LDA        (FCBPTR),Y           ; (HIGH BLOCK ADDRESS)
                JMP        RNEWPOS              ; GO READ IN BLOCK AND SET APPROPRIATE STATUSES.
*
                PAGE
POSNEW2         LDY        #FCBSTAT             ; GOTA CHECK TO SEE IF PREVIOUS
                LDA        (FCBPTR),Y           ; INDEX BLOCK WAS MODIFIED.
                AND        #IDXMOD
                BEQ        POSNIDX              ; READ IN OVER IT IF CURRENT IS UP TO DATE.
                JSR        WFCBIDX              ; GO UPDATE INDEX ON DISK (BLOCK ADDR IN FCB)
                BCS        POSERR
POSNIDX         LDX        LEVELS               ; BEFORE READING IN TOP INDEX, CHECK TO BE SURE
                CPX        #3                   ; THAT THERE IS A TOP INDEX...
                BEQ        POSINDEX             ; BRANCH IF FILE IS FULL BLOWN TREE.
                LDA        TPOSHI               ; IS NEW POSITION WITHIN RANGE OF A
                LSR        A                    ; SAPLING FILE (L.T. 128K)?
                PHP                             ; ANTICIPATE NO GOOD.
                LDA        #TOPALC+IDXALC+DATALC ; (TO INDICATE NO LEVEL IS ALLOCATED FOR NEW POSITION.)
                PLP                             ; Z FLAG TELLS ALL...
                BNE        NODATA               ; GO MARK 'EM ALL DUMMY.
                JSR        CLRSTATS             ; GO CLEAR STATUS BITS 0,1,2 (INDEX/DATA ALLOC STATUS).
                DEX                             ; (UNAFFECTED SINCE LOADED ABOVE) CHECK FOR SEED
                BEQ        TSTINY               ; IF SEED, CHECK FOR POSITION L.T. 512...
                JSR        RFCBFST              ; GO GET ONLY INDEX BLOCK
                BCS        POSERR               ; BRANCH IF ERROR
                LDY        #FCBIDXB             ; SAVE NEWLY LOADED INDEX BLOCK'S ADDRESS
                LDA        BLOKNML
                STA        (FCBPTR),Y
                INY
                LDA        BLOKNMH
                STA        (FCBPTR),Y
                BCC        DATLEVEL             ; BRANCH ALWAYS...
POSERR          SEC
                RTS
*
POSINDEX        JSR        CLRSTATS             ; CLEAR ALL ALLOCATION REQUIREMENTS FOR PREVIOUS POSITION
                JSR        RFCBFST              ; GET HIGHEST LEVEL INDEX BLOCK.
                BCS        POSERR
                LDA        TPOSHI               ; THEN TEST FOR A SAP LEVEL INDEX BLOCK
                LSR        A
                TAY
                LDA        (TINDX),Y
                INC        TINDX+1
                CMP        (TINDX),Y            ; (BOTH HI AND LO WILL BE ZERO IF NO INDEX EXISTS)
                BNE        SAPLEVEL
                CMP        #0                   ; ARE BOTH BYTES ZERO?
                BNE        SAPLEVEL
                DEC        TINDX+1              ; DON'T LEAVE WRONG POINTERS LAYING AROUND!
NOIDXDAT        LDA        #IDXALC+DATALC       ; SHOW NEITHER INDEX OR DATA BLOCK ALLOCATED.
                JMP        NODATA
*
                PAGE
SAPLEVEL        STA        BLOKNML              ; READ IN NEXT LOWER INDEX BLOCK
                LDA        (TINDX),Y            ; (HI ADDRESS)
                STA        BLOKNMH
                DEC        TINDX+1
                JSR        RFCBIDX              ; READ IN SAPLING LEVEL
                BCS        POSERR
DATLEVEL        LDA        TPOSHI               ; NOW GET BLOCK ADDRESS OF DATA BLOCK
                LSR        A
                LDA        TPOSLH               ; ( IF THERE IS ONE )
                ROR        A
                TAY
                LDA        (TINDX),Y            ; DATA BLOCK ADDRESS LOW
                INC        TINDX+1
                CMP        (TINDX),Y
                BNE        POSNEW3
                CMP        #0
                BNE        POSNEW3
                LDA        #DATALC              ; SHOW DATA BLOCK AS NEVER BEEN ALLOCATED
                DEC        TINDX+1
*
NODATA          LDY        #FCBSTAT
                ORA        (FCBPTR),Y           ; SET STATUS TO SHOW WHATS MISSIN'
                STA        (FCBPTR),Y
                LSR        A                    ; THROW AWAY BIT THAT SAYS DATA BLOCK UN-ALLOCATED
                LSR        A                    ; CUZ WE KNOW THAT. CARRY NOW INDICATES IF INDEX BLOCK
                JSR        ZIPDATA              ; ALSO IS INVALID AND NEEDS TO BE ZEROED (CARRY UNDISTURBED)
                BCC        SVMARK               ; BRANCH IF INDEX BLOCK DOESN'T NEED ZIPPIN.
ZIPIDX          STA        (TINDX),Y
                INY
                BNE        ZIPIDX
                INC        TINDX+1
ZPIDX1          STA        (TINDX),Y
                INY
                BNE        ZPIDX1
                DEC        TINDX+1              ; RESTORE PROPER ADDRESS
                JMP        SVMARK
*
ZIPDATA         LDA        #0                   ; ALSO IS INVALID AND NEEDS TO BE ZEROED.
```

```
                TAY
ZIPDAT0         STA        (DATPTR),Y           ; ZERO OUT DATA AREA
                INY
                BNE        ZIPDAT0
                INC        DATPTR+1
ZPDAT1          STA        (DATPTR),Y
                INY
                BNE        ZPDAT1
                DEC        DATPTR+1
                RTS
*
                PAGE
*
POSNEW3         STA        BLOKNML              ; GET DATA BLOCK OF NEW POSITION
                LDA        (TINDX),Y            ; (HI ADDRESS)
                DEC        TINDX+1
RNEWPOS         STA        BLOKNMH
                JSR        RFCBDAT
                BCS        PRITZ                ; RETURN ANY ERROR
                JSR        CLRSTATS             ; SHOW WHOLE CHAIN IS ALLOCATED
SVMARK          LDY        #FCBMARK+2           ; UPDATE POSITION IN FILE CONTROL BLOCK
                LDX        #2
SVMRK1          LDA        (FCBPTR),Y           ; REMEMBER OLDMARK IN CASE
                STA        OLDMARK-FCBMARK,Y    ; CALLING ROUTINE FAILS LATER
                LDA        TPOSLL,X
                STA        (FCBPTR),Y
                DEY
                DEX                             ; MOVE 3 BYTE POSITION MARKER
                BPL        SVMRK1
*
                CLC                             ; LAST, BUT NOT LEAST, SET UP
                LDA        DATPTR               ; INDIRECT ADDRESS TO BUFFER PAGE POINTED
                STA        POSPTR               ; TO BY THE CURRENT POSITION MARKER.
                LDA        TPOSLH
                AND        #1
                ADC        DATPTR+1
                STA        POSPTR+1
                LDA        SISDATP
                STA        SISPOSP              ; SISTER PAGE BYTE ALSO.
                RTS                             ; CARRY SHOULD ALWAYS BE CLEAR
PRITZ           SEC                             ; RANDOM ERROR
                RTS                             ; RETURN
*
*
CLRSTATS        LDY        #FCBSTAT             ; CLEAR ALLOCATION STATES FOR DATA BLOCK
                LDA        (FCBPTR),Y           ; AND BOTH LEVELS OF INDEXES.
                AND        #$FF-TOPALC-IDXALC-DATALC
                STA        (FCBPTR),Y           ; THIS SAYS THAT EITHER THEY EXIST CURRENTLY
                RTS                             ; OR THAT THEY'RE UNNECESSARY FOR CURRENT POSITION.
*
```

```
                PAGE
*
DIRMARK         CMP        #DIRTYP              ; IS IT A DIRECTORY?
                BEQ        DIRPOS               ; YES...
                LDA        #CPTERR              ; NO, THERE IS A COMPATABLITY PROBLEM-
                JSR        SYSERR               ; THE DAMN THING SHOULD OF NEVER BEEN OPENED!
*
DIRPOS          LDA        SCRTCH               ; RECOVER RESULTS OF PREVIOUS SUBTRACTION.
                LSR        A                    ; USE DIFFERENCE AS COUNTER AS TO HOW MANY
                STA        CNTENT               ; BLOCKS MUST BE READ TO GET TO NEW POSITION.
                LDY        #FCBMARK+1           ; TEST FOR POSITION DIRECTION.
                LDA        (FCBPTR),Y
                CMP        TPOSLH               ; CARRY INDICATES DIRECTION...
                BCC        DIRFWRD              ; IF SET, POSITION FORWARD.
DIRVRSE         LDY        #0                   ; OTHERWISE, READ DIRECTORY FILE IN REVERSE ORDER.
                JSR        DIRPOS1              ; READ PREVIOUS BLOCK.
                BCS        DRPOSERR             ; BRANCH IF ANYTHING GOES WRONG.
                INC        CNTENT               ; COUNT UP TO 128
                BPL        DIRVRSE              ; LOOP IF THERE IS MORE BLOCKS TO PASS OVER.
                BMI        SVMARK               ; BRANCH ALWAYS.
*
DIRFWRD         LDY        #2                   ; POSITION IS FORWARD FROM CURRENT POSITION.
                JSR        DIRPOS1              ; READ NEXT DIRECTORY BLOCK.
                BCS        DRPOSERR
                DEC        CNTENT
                BNE        DIRFWRD              ; LOOP IF POSITION NOT FOUND IN THIS BLOCK.
                BEQ        SVMARK               ; BRANCH ALWAYS.
*
DIRPOS1         LDA        (DATPTR),Y           ; GET LINK ADDRESS OF PREVIOUS OR
                STA        BLOKNML              ; NEXT DIRECTORY BLOCK.
                INY                             ; BUT FIRST BE SURE THERE IS A LINK.
                CMP        (DATPTR),Y
                BNE        DIRPOS2              ; BRANCH IF CERTAIN LINK EXISTS
                CMP        #0                   ; ARE BOTHE LINK BYTES 0?
                BNE        DIRPOS2              ; NOPE, JUST HAPPEN TO BE THE SAME VALUE.
                LDA        #EOFERR              ; SOMETHING IS WRONG WITH THIS DIRECTORY FILE!
DRPOSERR        SEC                             ; INDICATE ERROR
                RTS
*
DIRPOS2         LDA        (DATPTR),Y           ; (HIGH ORDER BLOCK ADDRESS)
                STA        BLOKNMH
* DROP INTO 'RFCBDAT' (READ FILE'S DATA BLOCK)
*
* NOTE: FOR DIRECTORY POSITIONING NO OPTIMIZATION HAS BEEN
* DONE SINCE DIRECTORY FILES WILL ALMOST ALWAYS BE LESS
* THAN 6 BLOCKS. IF MORE SPEED IS REQUIRED OR DIRECTORY
* TYPE FILES ARE TO BE USED FOR OTHER PURPOSES REQUIRING
* MORE BLOCKS, THEN THE RECOMMENDED METHOD IS TO CALL
* 'RFCBDAT' FOR THE FIRST BLOCK AND GO DIRECTLY TO
* DEVICE (VIA JMP (IOUNITL)) HANDLER FOR SUBSEQUENT
* ACCESSES.
* ALSO NOTE THAT NO CHECKING IS DONE FOR READ/WRITE
* ENABLE SINCE A DIRECTORY FILE CAN ONLY BE OPENED
* FOR READ ACCESS.
*
                PAGE
*
RFCBDAT         LDA        #RDCMD               ; SET READ COMMAND.
                STA        DHPCMD
                LDX        #DATPTR              ; USE X TO POINT AT ADDRESS OF DATA BUFFER
                JSR        FILEIO1              ; GO DO FILE INPUT.
                LDY        #FCBDATB             ; SAVE BLOCK NUMBER JUST READ IN FCB.
                BCC        FCBLOKNM             ; BRANCH IF NO ERRORS HAPPENED.
                RTS                             ; RETURN ERROR
*
RFCBIDX         LDA        #RDCMD               ; PREPARE TO READ IN INDEX BLOCK.
                STA        DHPCMD
                LDX        #TINDX               ; POINT AT ADDRESS OF CURRENT INDEX BUFFER
                JSR        FILEIO1              ; GO READ INDEX BLOCK.
                BCS        RDFCBERR             ; REPORT ERROR
                LDY        #FCBIDXB             ; SAVE BLOCK ADDRESS OF THIS INDEX IN FCB.
FCBLOKNM        LDA        BLOKNML
                STA        (FCBPTR),Y
                INY
                LDA        BLOKNMH
                STA        (FCBPTR),Y
                CLC
RDFCBERR        RTS
*
RFCBFST         LDX        #TINDX               ; POINT AT ADDRESS OF INDEX BUFFER
                LDY        #FCBFRST             ; AND BLOCK ADDRESS OF FIRST FILE BLOCK IN FCB
                LDA        #RDCMD               ; AND LASTLY, MAKE IT A READ!
* DROP INTO DOFILEIO
*
DOFILEIO        STA        DHPCMD               ; SAVE COMMAND.
                LDA        (FCBPTR),Y           ; GET DISK BLOCK ADDRESS FROM FCB.
                STA        BLOKNML
                INY                             ; BLOCK ZERO NOT LEGAL.
                CMP        (FCBPTR),Y
                BNE        FILEIO
                CMP        #0                   ; ARE BOTH BYTES ZERO?
                BNE        FILEIO               ; NO, CONTINUE WITH REQUEST.
                LDA        #ALCERR              ; OTHERWISE REPORT ALLOCATION ERROR.
                JSR        SYSDEATH             ; NEVER RETURNS...
*
                PAGE
FILEIO          LDA        (FCBPTR),Y           ; GET HIGH ADDRESS OF DISK BLOCK
                STA        BLOKNMH
FILEIO1         LDA        0,X                  ; GET MEMORY ADDRESS OF BUFFER FROM
                STA        DBUFPL               ; S.O.S. ZERO PAGE POINTED TO BY
                JSR        WRAPADJ              ;GO ADJUST FOR BANK CROSSING <SRS 82.162>
                LDA        1,X
                STA        DBUFPH               ; SET HI BYTE
                LDA        SISTER+1,X           ; AND BANK PAIR BYTE. <SRS 82.162>
                STA        SISBPH
                LDY        #FCBDEVN
                LDA        (FCBPTR),Y           ; OF COURSE HAVING THE DEVICE NUMBER
                STA        DEVNUM               ; WOULD MAKE THE WHOLE OPERATION MORE MEANINGFUL...
FILEIO2         LDA        #2                   ; ALSO, SET UP BYTE COUNT TO 512 AND
                STA        RQCNTH               ; SET 'BYTES READ' POINTER TO
                STA        IOACCESS             ; (INTERUPT! SET TO INDICATE REG CALL MADE TO DEV HANDLER. RETURN INTERUPT!)
                LDA        #>TRASH              ; A PLACE TO THROW BYTES READ AWAY
                STA        BRDPTR
                LDA        #<TRASH              ; LOCALLY DEFINED
                STA        BRDPTR+1
                LDA        #0                   ; SO THAT IT DOESN'T MESS UP ANY OTHER DATA.
                STA        RQCNTL
                STA        SSBRDPH              ; ('BYTES READ' IS THROWN AWAY)
RPEATIO1        LDA        DEVNUM               ; TRANSFER THE DEVICE NUMBER FOR DISPATCHER TO CONVERT TO UNIT NUMBER.
                STA        UNITNUM
RPEATIO0        LDY        #$9                  ; PREPARE TO SAVE DEVICE PARMS
SAVPRMS         LDA        DEVICE,Y             ; MOVE FROM Z PAGE
                STA        RPTBLOK,Y            ; TO MY OWN SPACE
                DEY                             ; FROM $C9 THROUGH $C0
                BPL        SAVPRMS
DMGRGO          EQU        *                    ; CALL EXTERNAL DEVICE MANAGER
                LDA        #0
                STA        SERR                 ; CLEAR GLOBAL ERROR VALUE
                JSR        DMGR                 ; CALL THE DRIVER
                BCC        RRITZ                ; RTS IF NO ERRORS
                CMP        #XDISKSW             ; DISKSWITCH ITERATES
                BEQ        RPEATIO2             ; BRANCH IF DISK SWITCH AND REPEAT I/O REQUEST
                SEC                             ; REPORT ERROR
RRITZ           RTS
RPEATIO2        LDY        #$9                  ; LENGTH OF PARM BLOCK
GETPRMS         LDA        RPTBLOK,Y
                STA        DEVICE,Y             ; RESTORE POSSIBLY DISTURBED PARM BLOCK
                DEY
                BPL        GETPRMS
                JMP        DMGRGO               ; AND TRY THE I/O AGAIN
*
*
TRASH           DS         2                    ; ONLY USED TO PUT BYTES READ TO SLEEP
RPTBLOK         DS         10                   ; DMGR PARM SAVE BLOCK
*
*
WFCBFST         LDY        #FCBDEVN             ; FETCH THE
                LDA        (FCBPTR),Y           ;  DEVICE NUMBER
                TAX                             ;    AND UPDATE
                JSR        UPBMAP               ;      ITS BITMAP
                LDX        #TINDX               ; POINT AT ADDRESS OF INDEX BLOCK
                LDY        #FCBFRST             ; AND THE DISK ADDRESS OF FILE'S FIRST BLOCK IN FCB
                LDA        #WRTCMD              ; LASTLY, MAKE IT A WRITE REQUEST.
                JMP        DOFILEIO             ; AND GO DO IT!
*
WFCBDAT         LDX        #DATPTR
                LDY        #FCBDATB             ; POINT AT MEMORY ADDRESS WITH X AND DISK ADDRESS WITH Y.
                LDA        #WRTCMD              ; WRITE DATA BLOCK.
                JSR        DOFILEIO
                BCS        FILIOERR             ; REPORT ANY ERRORS
                LDA        #$FF-DATMOD          ; MARK DATA STATUS AS CURRENT.
                JMP        FCBUPDAT
*
WFCBIDX         LDY        #FCBDEVN             ; MAKE SURE
                LDA        (FCBPTR),Y           ;  THE BITMAP
                TAX                             ;    FOR THIS DEVICE ("X")
                JSR        UPBMAP               ;      IS UPDATED
                LDX        #TINDX               ; POINT AT ADDRESS OF INDEX BUFFER
                LDY        #FCBIDXB             ; AND BLOCK ADDRESS OF THAT INDEX BLOCK.
                LDA        #WRTCMD
                JSR        DOFILEIO             ; GO WRITE OUT INDEX BLOCK.
                BCS        FILIOERR             ; REPORT ANY ERRORS
                LDA        #$FF-IDXMOD          ; MARK INDEX STATUS AS CURRENT.
FCBUPDAT        LDY        #FCBSTAT             ; CHANGE STATUS BYTE TO
                AND        (FCBPTR),Y           ; REFLECT SUCCESSFUL DISK FILE UPDATE.
                STA        (FCBPTR),Y           ; (CARRY IS UNAFFECTED)
FILIOERR        RTS
*
*
                PAGE
OPEN            JSR        FINDFILE             ; FIRST OF ALL LOOK UP THE FILE...
                BCC        OPEN0
                CMP        #BADPATH             ; IS AN ATTEMPT TO OPEN A ROOT DIRECTORY?
                BNE        ERROPN               ; NO, PASS BACK ERROR
*
OPEN0           JSR        TSTOPEN              ; FIND OUT IF ANY OTHER FILES ARE WRITING
                BCC        OPEN1                ; TO THIS SAME FILE. (BRANCH IF NOT)
ERRBUSY         LDA        #FILBUSY             ; REPORT SHARED ACCESS NOT ALLOWED.
ERROPN          SEC
                RTS                             ; RETURN ERROR.
*
OPEN1           LDA        DATPTR               ; GET ADDRESS OF FIRST FREE FCB FOUND
                STA        FCBPTR               ; DURING TEST OPEN SEQUENCE AND USE
                LDA        DATPTR+1             ; IT AS FILE CONTROL AREA. IF HIGH BYTE OF
                STA        FCBPTR+1             ; POINTER IS ZERO, THEN NO FCB
                BNE        ASGNFCB              ; IS AVAILABLE FOR USE.
                LDA        #FCBFULL             ; REPORT FCB FULL ERROR.
                SEC
                RTS
*
```

```
ASGNFCB         LDY        #$1F                 ; ASSIGN FCB, BUT FIRST
                LDA        #0                   ; CLEAN OUT ANY OLD RUBBISH LEFT AROUND...
CLRFCB          STA        (FCBPTR),Y
                DEY
                BPL        CLRFCB
                LDY        #FCBENTN             ; NOW BEGIN CLAIM BY MOVING IN FILE
FCBOWNR         LDA        D.DEV-1,Y            ; OWNERSHIP INFORMATION.
                STA        (FCBPTR),Y           ; NOTE: THIS CODE DEPENDS UPON THE DEFINED
                DEY                             ; ORDER OF BOTH THE FCB AND DIRECTORY ENTRY
                BNE        FCBOWNR              ; BUFFER (D.). BEWARE OF CHANGES!!! *************
                LDA        DFIL+D.STOR          ; GET STORAGE TYPE.
                LSR        A                    ; STRIP OFF FILE NAME LENGTH.
                LSR        A
                LSR        A                    ; (BY DIVIDING BY 16)
                LSR        A
                TAX                             ; SAVE IN X FOR LATER TYPE COMPARISON
                LDY        #FCBSTYP
                STA        (FCBPTR),Y           ; SAVE STORAGE TYPE.
                LDA        C.OPLSTLN            ; IS THERE AN OPEN LIST?
                BEQ        DEFOPEN              ; NO, USE DEFAULT REQUST ACCESS...
                LDY        #0                   ; YES, FIND OUT WHAT ACCESS IS REQUESTED.
                LDA        (C.OPLIST),Y         ; IF REQ-ACCESS IS ZERO, THEN
                BEQ        DEFOPEN              ; USE DEFAULTS...
                AND        DFIL+D.ATTR          ; CHECK REQUEST AGAINST ATTRIBUTES.
                CMP        (C.OPLIST),Y         ; WERE ALL ACCESS REQUESTS SATISFIED?
                BEQ        SVATTRB              ; YES, SAVE ATTRIBUTES.
                LDA        #ACCSERR             ; REPORT ACCESS REQUEST CAN'T BE MET.
                SEC
                RTS
                PAGE
DEFOPEN         LDA        DFIL+D.ATTR          ; GET FILES ATTRIBUTES AND
                AND        #READEN+WRITEN       ; USE IT AS A DEFAULT ACCESS REQUEST.
SVATTRB         LDY        #FCBATTR
                CPX        #DIRTYP              ; IF DIRECTORY, DON'T ALLOW WRITE ENABLE
                BNE        SVATTR1
                AND        #READEN
SVATTR1         STA        (FCBPTR),Y
                AND        #WRITEN              ; CHECK FOR WRITE ENABLED REQUESTED.
                BEQ        OPEN2                ; BRANCH IF READ ONLY OPEN.
                LDA        TOTENT               ; OTHERWISE, BE SURE NO ONE ELSE IS READING SAME
                BNE        ERRBUSY              ; FILE (SET UP BY TSTOPEN).
OPEN2           LDA        DFIL+D.COMP          ; OH, BY THE WAY... IS THIS FILE
                BEQ        OPEN3                ; COMPATABLE WITH VERSION 0000? ***************
ERRCMPAT        LDA        #CPTERR              ; REPORT FILE IS INCOMPATABLE!
                SEC
                RTS
*
OPEN3           CPX        #TRETYP+1            ; IS IT A TREE TYPE FILE?
                BCC        OPEN4                ; TEST FOR FURTHER COMPATABLITY. IT MUST
                CPX        #DIRTYP              ; BE EITHER A TREE OR A DIRECTORY.
                BNE        ERRCMPAT             ; REPORT INCOMPATABLE.
OPEN4           LDY        #FCBFRST             ; MOVE ADDRESS OF FIRST BLOCK OF FILE
                LDA        DFIL+D.FRST          ; INTO FCB. NO CHECKING IS DONE FOR VALIDITY.
                STA        (FCBPTR),Y
                STA        BLOKNML
                INY
                LDA        DFIL+D.FRST+1
                STA        (FCBPTR),Y           ; NOTE: THE FCB HAS NOT BEEN OFFICIALLY
                STA        BLOKNMH              ; CLAIMED YET. TO DO THIS, THE FIRST BYTE
                LDY        #FCBEOF              ; MUST CONTAIN A VALID REFERENCE NUMBER.
EOFCBMV         LDA        DFIL+D.EOF-FCBEOF,Y  ; MOVE CURRENT END OF FILE
                STA        (FCBPTR),Y           ; TO FCB.
                INY
                CPY        #FCBEOF+3
                BNE        EOFCBMV
                LDA        DFIL+D.USAGE
                STA        (FCBPTR),Y           ; AND CURRENT BLOCK COUNT OF FILE.
                INY
                LDA        DFIL+D.USAGE+1
                STA        (FCBPTR),Y
                LDA        C.OPLSTLN            ; NOW THAT WE'VE COME THIS FAR, FIND
                BEQ        DEFBUFR              ; OUT WHICH TYPE OF BUFFER AND ALLOCATE IT!
                CMP        #1                   ; WAS IT ONLY TO SET ATTRIBUTES?
                BEQ        DEFBUFR
                CMP        #4                   ; IS A FULL ADDRESS INCLUDED?
                BEQ        UBUFSPEC
                LDA        #BADLSTCNT
                SEC
                RTS
*
                PAGE
UBUFSPEC        LDY        #1                   ; (INDEX TO 'PAGECNT' OF OPEN LIST)
                LDA        (C.OPLIST),Y         ; IS USER SPECIFING THE BUFFER?
                BEQ        DEFBUFR              ; NO, USE DEFAULT BUFFER (DYNAMIC)
                CPX        #TRETYP+1            ; IF TREE TYPE FILE, THEN AT LEAS 4 PAGES ARE NEEDED.
                BCC        ONEKTST              ; BRANCH IF TREE TYPE.
                CMP        #2                   ; DID USER GIVE AT LEAST 2 PAGES FOR DIRECTORY TYPE?
                BCS        FIXDBUF              ; YES, LOG IT WITH BUFFER MANAGER
ERRBTS          LDA        #BTSERR              ; REPORT NOT ENOUGH BUFFER SPACE.
                SEC
                RTS
*
ONEKTST         CMP        #4                   ; IS THERE AT LEAST ONE KILOBYTE BUFFER FOR TREES?
                BCC        ERRBTS               ; NO, THEN TO HELL WITH IT!.
FIXDBUF         JSR        REQFXBUF             ; CALL BOB AND ASK FOR HIM TO FIX IT...
                BCC        FCBUFFER             ; GO SAVE BUFFER NUMBER.
ERROPN1         RTS                             ; RETURN ANY ERROR ENCOUNTERED.
*
DEFBUFR         LDA        #4                   ; ASSUME TREE FILE (4 PAGES REQUIRED)
                CPX        #TRETYP+1
                BCC        BUFREQST             ; BRANCH IF IT IS A TREE.
                LDA        #2                   ; OTHERWIZE, WE JUST NEED TWO PAGES.
BUFREQST        JSR        REQBUF               ; CALL BOB TO ALLOCATE A DYNAMIC BUFFER.
                BCS        ERROPN1              ; REPORT ANY ERRORS.
FCBUFFER        LDY        #FCBBUFN             ; SAVE BUFFER NUMBER AND THEN
                STA        (FCBPTR),Y           ; FIND OUT WHERE IT IS.
                JSR        GTBUFFRS             ; HAVE BOB RETURN ADDRESS IN DATA & INDEX POINTERS.
                BCS        ERROPEN2             ; IF ERROR, FREE BUFFER BEFOR RETURNING.
                LDY        #FCBREFN             ; NOW CLAIM FCB FOR THIS FILE.
                LDA        CNTENT               ; THIS WAS SET UP BY 'TSTOPEN'............
                STA        (FCBPTR),Y
                LDY        #FCBLEVL             ; MARK LEVEL
                LDA        LEVEL                ; AT WHICH
                STA        (FCBPTR),Y           ; FILE WAS OPENED
                LDY        #FCBSTYP             ; GET STORAGE TYPE AGAIN.
                LDA        (FCBPTR),Y           ; FILE MUST BE POSITIONED TO BEGINNING.
                CMP        #TRETYP+1            ; IS IT A TREE FILE?
                BCS        OPNDIR               ; NO, ASSUME IT'S A DIRECTORY.
                LDA        #$FF                 ; FOOL THE POSITION ROUTINE INTO GIVING
                LDY        #FCBMARK             ; A VALID POSITION WITH PRELOADED DATA, ETC.
OPNPOS          STA        (FCBPTR),Y
                INY
                CPY        #FCBMARK+3
                BNE        OPNPOS
                LDY        #2                   ; SET DESIRED POSITION TO ZERO.
                LDA        #0
OPNPOS1         STA        TPOSLL,Y
                DEY
                BPL        OPNPOS1
                JSR        RDPOSN               ; LET TREE POSITION ROUTINE DO THE REST.
                BCC        OPENDONE             ; BRANCH IF SUCCESSFUL.
*
                PAGE
ERROPEN2        PHA                             ; SAVE ERROR CODE.
                LDY        #FCBBUFN             ; SINCE ERROR WAS ENCOUNTERED BEFORE FILE
                LDA        (FCBPTR),Y           ; WAS SUCCESSFULLY OPENED, THEN
                JSR        RELBUF               ; IT'S NECESSARY TO FREE THE BUFFER AND
                LDY        #FCBREFN             ; FILE CONTROL BLOCK.
                LDA        #0
                STA        (FCBPTR),Y
                PLA
                SEC
                RTS
*
OPNDIR          JSR        RFCBDAT              ; READ IN FIRST BLOCK OF DIRECTORY FILE.
                BCS        ERROPEN2             ; RETURN ANY ERROR AFTER FREEING BUFFER & FCB
OPENDONE        LDY        #VCBOPNC             ; INCREMENT OPEN COUNT FOR THIS
                LDA        (VCBPTR),Y           ; VOLUME. ALSO MARK STATUS.
                CLC
                ADC        #1
                STA        (VCBPTR),Y
                LDY        #VCBSTAT             ; HI BIT INDICATES VOLUME BUSY
                LDA        (VCBPTR),Y
                ORA        #$80
                STA        (VCBPTR),Y           ; DOESN'T MATTER HOW MANY, JUST BE SURE IT'S SET.
                LDY        #FCBREFN             ; PASS USER HIS REFERENCE NUMBER
                LDA        (FCBPTR),Y
                LDY        #0
                STA        (C.OUTREF),Y
                CLC
                RTS
*
                PAGE
*
TSTOPEN         LDA        FCBADDRH             ; TEST FOR SHARED ACCESS FILES WITH WRITE ENABLED.
                STA        FCBPTR+1
                LDA        FCBANKNM
                STA        SISFCBP
                LDA        #0
                STA        DATPTR+1             ; MARK AS NO FREE FOUND.
                STA        CNTENT
                STA        TOTENT               ; ALSO, INIT COUNT OF MATCHING FILES
TSTOPN1         STA        FCBPTR               ; SAVE NEW LOW ORDER ADDRESS
                LDX        DATPTR+1             ; FIND OUT IF A FREE SPOT HAS BEEN FOUND YET.
                BNE        TSTOPN2              ; YES, DON'T INCREMENT REFNUM (CNTENT).
                INC        CNTENT               ; BUMP REFNUM
TSTOPN2         LDY        #FCBREFN             ; TEST FOR IN USE FCB
                LDA        (FCBPTR),Y           ; (NON ZERO)
                BNE        CHKACTV              ; THIS FCB IS IN USE, COPARE OWNERSHIP.
                TXA                             ; TEST AGAIN FOR FREE FCB
                BNE        TSNXFCB              ; BRANCH IF A FREE SPOT HAS ALREADY BEEN FOUND.
                LDA        FCBPTR               ; TRANSFER CURRENT POINTER SO IT MAY BE
                STA        DATPTR               ; USED AS A FREE FCB BY OPEN.
                LDA        FCBPTR+1             ; HIGH BYTE ALWAYS NON ZERO.
                STA        DATPTR+1
                JMP        TSNXFCB
*
CHKACTV         EQU        *                    ; IF MATCHING FILE IS SWAPPED, IT DOESNT COUNT
                LDY        #FCBSWAP
                LDA        (FCBPTR),Y
                BNE        TSNXFCB              ; BRANCH IF SWAPPED
                LDY        #FCBENTN             ; NOTE: THIS CODE DEPENDS ON THE
WHOWNS          LDA        (FCBPTR),Y           ; DEFINED ORDER OF FCB AND DIRECTORY
                CMP        D.DEV-1,Y            ; *****************************
                BNE        TSNXFCB              ; BRANCH IF THIS ONE HAS A DIFFERENT OWNER.
                DEY
                BNE        WHOWNS
                INC        TOTENT               ; REPORT THIS ONE AS A CO-OWNER.
                LDY        #FCBATTR             ; NOW FIND OUT IF THIS ONE WANTS TO WRITE.
                LDA        (FCBPTR),Y
                AND        #WRITEN              ; IF WRITE IS NOT ENABLED THEN CONTINUE.
                BEQ        TSNXFCB
                SEC                             ; OTHERWISE, JUST SET THE CARRY TO SHOW
                RTS                             ; THAT THE FILE CAN'T BE SHARED.
*
TSNXFCB         LDA        FCBPTR               ; CALCULATE NEXT FCB AREA (+$20)
                CLC
                ADC        #$20
                BCC        TSTOPN1              ; LOOP IF NO PAGE CROSS.
                LDX        FCBPTR+1
                INC        FCBPTR+1
                CPX        FCBADDRH             ; HAVE WE LOOKED AT BOTH PAGES?
                BEQ        TSTOPN1              ; NOPE, LOOK AT PAGE TWO.
                CLC                             ; INDICATE NO FILES THAT SHARE HAVE WRITE ENABLED,
                RTS
*
                CHN        READ/WRITE,4,2

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: POSN.OPEN
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.4of5.FOUR:SOS.READ.WRITE.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: READ.WRITE
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                PAGE
READ            CLC                                ; FIRST DETERMINE IF REQESTED
                LDY        #FCBATTR                ; READ IS LEGAL
                LDA        (FCBPTR),Y
                AND        #READEN                 ; IS READ ENABLED?
                BNE        READ1                   ; YES, CONTINUE...
                LDA        #ACCSERR                ; REPORT ILLEGAL ACCESS.
                SEC
                RTS
*
READ1           LDY        #FCBMARK                ; GET CURRENT MARK INTO 'TPOS' AND
                LDA        (FCBPTR),Y              ; DETERMINE IF RESULTING POSITION
                STA        TPOSLL                  ; EXCEEDS CURRENT END OF FILE.
                ADC        C.BYTES
                STA        SCRTCH
                INY
                LDA        (FCBPTR),Y
                STA        TPOSLH
                ADC        C.BYTES+1               ; (THIS WAS DONE STRAIT-LINE SINCE
                STA        SCRTCH+1                ; WE'RE ADDING A TWO BYTE TO A THREE
                INY                                ; BYTE QUANTITY)
                LDA        (FCBPTR),Y
                STA        TPOSHI
                ADC        #0                      ; ADD IN REMAINING CARRY.
                STA        SCRTCH+2
                LDY        #FCBEOF+2               ; NOW TEST EOF AGAINST POSITION GENERATED
EOFTEST         LDA        SCRTCH-FCBEOF,Y
                CMP        (FCBPTR),Y              ; IS NEW POSITION > EOF?
                BCC        READ2                   ; NO, PROCEED.
                BNE        ADJSTCNT                ; YES, ADJUST 'C.BYTES' REQUEST
                DEY
                CPY        #FCBEOF-1               ; HAVE WE COMPARED ALL TREE BYTES?
                BNE        EOFTEST                 ; NO, TEST NEXT LOWEST.
ADJSTCNT        EQU        *                       ; ADJUST REQUEST TO READ UP TO (BUT
                LDY        #FCBEOF                 ; NOT INCLUDING) END OF FILE.
                LDA        (FCBPTR),Y              ; RESULT= (EOF-1)-POSITION
                SBC        TPOSLL
                STA        C.BYTES
                INY
                LDA        (FCBPTR),Y
                SBC        TPOSLH
                STA        C.BYTES+1
                ORA        C.BYTES              ; IF BOTH BYTES ARE ZERO, REPORT EOF ERROR.
                BNE        READ2
                LDA        #EOFERR
                JSR        SYSERR
READ2           LDA        C.BYTES
                STA        RWREQL
                BNE        READ3                ; BRANCH IF READ REQUEST DEFINITELY NON-ZERO.
                CMP        C.BYTES+1
                BNE        READ3                ; BRANCH IF READ REQUEST<>ZERO
                STA        RWREQH
GORDDNE         JMP        READONE              ; DO NOTHING.
                PAGE
*
READ3           LDA        C.BYTES+1
                STA        RWREQH
                LDA        C.OUTBUF             ; MOVE POINTER TO USERS BUFFER TO BFM
                STA        USRBUF               ; Z-PAGE AREA.
                LDX        #C.OUTBUF            ; <SRS 82.162>
                JSR        WRAPADJ              ; ADJUST FOR BANK CROSSING. <SRS 82.162>
                STA        USRBUF+1
                STY        SISUSRBF             ; SAVE VALID USER BUFFER ADDRESS (THAT WILL NOT CROSS BANKS)
                LDY        #FCBSTYP             ; NOW FIND OUT IF IT'S A TREE READ OR OTHER.
                LDA        (FCBPTR),Y
                CMP        #TRETYP+1
                BCC        TREAD                ; BRANCH IF A TREE FILE.
                JMP        DREAD                ; OTHEWISE ASSUME IT'S A DIRECTORY.
*
TREAD           JSR        RDPOSN               ; GET DATA POINTER SET UP.
                BCC        TREAD0               ; REPORT ANY ERRORS
                JMP        ERRFIX1
TREAD0          JSR        PREPRW               ; TEST FOR NEWLINE, SETS UP FOR PARTIAL READ.
                JSR        READPART             ; MOVE CURRENT DATA BUFFER CONTENTS TO USER AREA
                BVS        GORDDNE              ; BRANCH IF REQUEST IS SATISFIED.
                BCS        TREAD                ; CARRY SET INDICATES NEWLINE IS SET.
                LDA        RWREQH               ; FIND OUT HOW MANY BLOCKS ARE TO BE READ
                LSR        A                    ; IF LESS THAN TWO, THEN DO IT THE SLOW WAY.
                BEQ        TREAD
                STA        BULKCNT              ; SAVE BULK BLOCK COUNT.
                LDY        #FCBSTAT             ; MAKE SURE CURRENT DATA AREA
                LDA        (FCBPTR),Y           ; DOESN'T NEED TO BE WRITTEN BEFORE
                AND        #DATMOD              ; RESETTING POINTER TO READ DIRECTLY INTO
                BNE        TREAD                ; USER'S AREA. BRANCH IF DATA NEED TO BE WRITTEN
                STA        IOACCESS             ; TO FORCE FIRST CALL THRU ALL DEVICE HANDLER CHECKING.
                LDA        USRBUF               ; MAKE THE DATA BUFFER THE USER'S SPACE.
                STA        DATPTR
                LDA        USRBUF+1
                STA        DATPTR+1
                LDA        SISUSRBF
                STA        SISDATP
*
                PAGE
RDFAST          JSR        RDPOSN               ; GET NEXT BLOCK DIRECTLY INTO USER SPACE.
                BCS        ERRFIX               ; BRANCH ON ANY ERROR.
RDFAST0         INC        DATPTR+1             ; BUMP ALL POINTERS BY 512 (ONE BLOCK)
                INC        DATPTR+1
                DEC        RWREQH
                DEC        RWREQH
                INC        TPOSLH
                INC        TPOSLH
                BNE        RDFAST1              ; BRANCH IF POSITION DOES NOT GET TO A 64K BOUNDARY.
                INC        TPOSHI               ; OTHERWISE, MUST CHECK FOR A 128K BOUNDARY
                LDA        TPOSHI               ; SET CARRY IF MOD 128K HAS BEEN REACHED
                EOR        #1
                LSR        A
RDFAST1         DEC        BULKCNT              ; HAVE WE READ ALL WE CAN FAST?
                BNE        RDFAST2              ; BRANCH IF MORE TO READ.
                JSR        FXDATPTR             ; GO FIX UP DATA POINTER TO SOS BUFFER.
                LDA        RWREQL               ; TEST FOR END OF READ.
                ORA        RWREQH               ; ARE BOTH ZERO?
                BEQ        READONE
                BNE        TREAD                ; NO, READ LAST PARTIAL BLOCK.
*
RDFAST2         BCS        RDFAST
                LDA        TPOSHI               ; GET INDEX TO NEXT BLOCK ADDRESS
                LSR        A
                LDA        TPOSLH
                ROR        A
                TAY                             ; INDEX TO ADDRESS IS INT(POS/512)
                LDA        (TINDX),Y            ; GET LOW ADDRESS
                STA        BLOKNML
                INC        TINDX+1
                CMP        (TINDX),Y            ; ARE BOTH HI AND LOW ADDRESS THE SAME?
                BNE        REALRD               ; NO, IT'S A REAL BLOCK ADDRESS.
                CMP        #0                   ; ARE BOTH BYTES ZERO?
                BNE        REALRD               ; NOPE -- MUST BE REAL DATA
                STA        IOACCESS             ; DON'T DO REPEATIO JUST AFTER SPARSE
                BEQ        NOSTUF               ; BRANCH ALWAYS (CARRY SET)
REALRD          LDA        (TINDX),Y            ; GET HIGH ADDRESS BYTE
                CLC
NOSTUF          DEC        TINDX+1
                BCS        RDFAST               ; BRANCH IF NO BLOCK TO READ
                STA        BLOKNMH
                LDA        IOACCESS             ; HAS FIRST CALL GONE TO DEVICE YET?
                BEQ        RDFAST               ; NOPE, GO THRU NORMAL ROUTE...
                LDA        DATPTR+1             ; RESET HI BUFFER ADDRESS FOR DEVICE HANDLER
                STA        DBUFPH
                JSR        REPEATIO
                BCC        RDFAST0              ; BRANCH IF NO ERRORS.
                PAGE
ERRFIX          PHA                                ; SAVE ERROR CODE
                JSR        FXDATPTR                ; GO RESTORE DATA POINTERS, ETC...
                PLA
ERRFIX1         PHA                                ; SAVE ERROR CODE
                JSR        READONE                 ; PASS BACK NUMBER OF BYTES ACTUALLY READ.
                PLA
                SEC                                ; REPORT ERROR
                RTS
*
READONE         LDY        #0                      ; RETURN TOTAL NUMBER OF BYTES ACTUALLY READ
                SEC                                ; THIS IS DERIVED FROM C.BYTES-RWREQ
                LDA        C.BYTES
                SBC        RWREQL
                STA        (C.OUTCNT),Y
                INY
                LDA        C.BYTES+1
                SBC        RWREQH
                STA        (C.OUTCNT),Y
                JMP        RDPOSN                  ; LEAVE WITH VALID POSITION IN FCB.
*
PREPRW          SEC                                ; ADJUST POINTER TO USER'S BUFFER TO
                LDA        USRBUF                  ; MAKE THE TRANSFER
                SBC        TPOSLL
                STA        USRBUF
                BCS        PREPRW1                 ; BRANCH IF NO ADJUSTMENT TO HI ADDR. NEEDED.
                DEC        USRBUF+1                ; NOTE: SARA ALLOWS INDIRECT FROM $101 UP
PREPRW1         LDY        #FCBATTR                ; AS LONG AS ACTUAL RESULTING ADDRESS IS >=$200
                LDA        (FCBPTR),Y              ; TEST FOR NEW LINE ENABLED
                AND        #NLINEN                 ; SET CARRY IF IT IS.
                CLC
                BEQ        NONEWLIN                ; BRANCH IF NEWLINE IS NOT ENABLED
                SEC
                LDY        #FCBNEWL
                LDA        (FCBPTR),Y              ; MOVE NEWLINE CHARACTER TO MORE
                STA        NLCHAR                  ; ACCESSABLE SPOT.
NONEWLIN        LDY        TPOSLL                  ; GET INDEX TO FIRST DATA
                LDA        DATPTR                  ; RESET LOW ORDER OF POSPTR TO BEGINNING OF PAGE.
                STA        POSPTR
                LDX        RWREQL                  ; AND LASTLY GET LOW ORDER COUNT OF REQUESTED BYTES.
                RTS                                ; RETURN STATUSES...
*
READPART        TXA
                BNE        RDPART0                 ; BRANCH IF REQUEST IS NOT A EVEN PAGES
                LDA        RWREQH                  ; A CALL OF ZERO BYTES SHOULD NEVER GET HERE!
                BEQ        SETRDNE                 ; BRANCH IF NOTHIN' TO DO.
                DEC        RWREQH
RDPART0         DEX
RDPART          LDA        (POSPTR),Y              ; MOVE DATA TO USER'S BUFFER
                STA        (USRBUF),Y              ; ONE BYTE AT A TIME.
                TXA                                ; NOTE: THIS ROUTINE IS CODED TO BE
                BEQ        ENDRQCHK                ; FASTEST WHEN NEWLINE IS DISABLED.
RDPART1         BCS        TSTNEWL                 ; BRANCH IF NEW LINE NEEDS TO BE TESTED.
RDPART2         DEX
                INY                                ; PAGE CROSSED?
                BNE        RDPART                  ; NO. MOVE NEXT BYTE.
                LDA        POSPTR+1                ; TEST FOR END OF BUFFER
                INC        USRBUF+1                ; BUT FIRST ADJUST USER BUFFER POINTER
                INC        TPOSLH                  ; AND POSITION.
                BNE        RDPART3
                INC        TPOSHI
RDPART3         INC        POSPTR+1                ; AND SOS BUFFER HIGH ADDRESS.
                EOR        DATPTR+1                ; (CARRY HAS BEEN CLEVERLY UNDISTURBED.)
                BEQ        RDPART                  ; BRANCH IF MORE TO READ IN BUFFER.
                CLV                                ; INDICATE NOT FINISHED.
                BVC        RDPRTDNE                ; BRANCH ALWAYS.
*
ENDRQCHK        LDA        RWREQH
                BEQ        RDRQDNE                 ; BRANCH IF REQEST SATISFIED.
                INY                                ; DONE WITH THIS BLOCK OF DATA?
                BNE        ENDRCHK1                ; NO, ADJUST HIGH BYTE OF REQUEST.
                LDA        POSPTR+1                ; MAYBE- CHECK FOR END OF BLOCK BUFFER.
                EOR        DATPTR+1                ; (DON'T DISTURB CARRY)
                BNE        ENDRCHK2                ; BRANCH IF HI COUNT CAN BE DEALT WITH NEXT TIME.
ENDRCHK1        DEC        RWREQH
ENDRCHK2        DEY                                ; RESTORE PROPER VALUE TO 'Y'
                JMP        RDPART1
*
TSTNEWL         LDA        (POSPTR),Y              ; GET LAST BYTE TRANSFERED AGAIN.
                EOR        NLCHAR                  ; HAVE WE MATCHED NEWLINE CHARACTER?
                BNE        RDPART2                 ; NO, READ NEXT.
RDRQDNE         INY                                ; ADJUST POSITION.
                BNE        SETRDNE
                INC        USRBUF+1                ; BUMP POINTERS.
                INC        TPOSLH
                BNE        SETRDNE
                INC        TPOSHI
SETRDNE         BIT        SETVFLG                 ; (SET V FLAG)
RDPRTDNE        STY        TPOSLL                  ; SAVE LOW POSITION
                BVS        RDONE1
                INX                                ; LEAVE REQUEST AS +1 FOR NEXT CALL
RDONE1          STX        RWREQL                  ; AND REMAINDER OF REQUEST COUNT.
                PHP                                ; SAVE STATUSES
                CLC                                ; ADJUST USER'S LOW BUFFER ADDRESS
                TYA
                ADC        USRBUF
                STA        USRBUF
                BCC        RDPART4
                INC        USRBUF+1                ; ADJUST HI ADDRESS AS NEEDED.
RDPART4         PLP                                ; RESTORE RETURN STATUSES
```

```
SETVFLG         RTS                           ; (THIS BYTE <$60> IS USED TO SET V FLAG)
*
FXDATPTR        LDA       DATPTR              ; PUT CURRENT USER BUFFER
                STA       USRBUF              ; ADDRESS BACK TO NORMAL
                LDA       DATPTR+1
                STA       USRBUF+1            ; BANK PAIR BYTE SHOULD BE MOVED ALSO.
                LDA       SISDATP
                STA       SISUSRBF
                LDY       #FCBBUFN            ; RESTORE BUFFER ADDRESS
                LDA       (FCBPTR),Y
                LDX       #DATPTR
                JMP       GETBUFADR           ; END VIA CALL TO BOB'S CODE.
*
                PAGE
*
* READ DIRECTORY FILE...
*
DREAD           JSR       RDPOSN
                BCS       ERRDRD              ; PASS BACK ANY ERRORS
                JSR       PREPRW              ; PREPARE FOR TRANSFER.
                JSR       READPART            ; MOVE DATA TO USER'S BUFFER
                BVC       DREAD               ; REPEAT UNTIL REQUEST IS SATISFIED.
                JSR       READONE             ; UPDATE FCB AS TO NEW POSITION.
                BCC       DREDONE             ; BRANCH IF ALL IS WELL.
                CMP       #EOFERR             ; WAS LAST READ TO END OF FILE?
                SEC                           ; ANTICIPATE SOME OTHER PROBLEM
                BNE       DREDERR             ; BRANCH IF NOT EOF ERROR.
                JSR       SVMARK
                JSR       ZIPDATA             ; CLEAR OUT DATA BLOCK.
                LDY       #FCBDATB+1          ; PROVIDE DUMMY BACK POINTER FOR FUTURE RE-POSITION
                LDA       (FCBPTR),Y          ; GET HI BYTE OF LAST BLOCK.
                PHA
                DEY
                LDA       (FCBPTR),Y          ; AND LOW BYTE.
                PHA
                LDA       #0                  ; NOW MARK CURRENT BLOCK AS IMPOSIBLE.
                STA       (FCBPTR),Y
                INY
                STA       (FCBPTR),Y
                TAY                           ; NOW MOVE LAST BLOCK ADDRESS TO DATA BUFFER AS BACK POINTER.
                PLA
                STA       (DATPTR),Y
                PLA
                INY
                STA       (DATPTR),Y
DREDONE         CLC                           ; INDICATE NO ERROR
DREDERR         RTS
*
ERRDRD          JMP       ERRFIX1             ; REPORT HOW MUCH WE COULD TRANSFER BEFORE ERROR.
*
```

```
                PAGE
WRITE           CLC                              ; FIRST DETERMINE IF REQESTED
                LDY        #FCBATTR              ; WRITE IS LEGAL
                LDA        (FCBPTR),Y
                AND        #WRITEN               ; IS WRITE ENABLED?
                BNE        WRITE1                ; YES, CONTINUE...
ERRACCS         LDA        #ACCSERR              ; REPORT ILLEGAL ACCESS.
                SEC
WPERROR         RTS
*
WRITE1          JSR        TSTWPROT              ; OTHERWISE, MAKE SURE DEVICE IS NOT WRITE PROTECTED.
                BCS        WPERROR               ; REPORT WRITE PROTECTED AND ABORT OPERATION.
*
                LDY        #FCBMARK              ; GET CURRENT MARK INTO 'TPOS' AND
                LDA        (FCBPTR),Y            ; DETERMINE IF RESULTING POSITION
                STA        TPOSLL                ; EXCEEDS CURRENT END OF FILE.
                ADC        C.BYTES
                STA        SCRTCH
                INY
                LDA        (FCBPTR),Y
                STA        TPOSLH
                ADC        C.BYTES+1             ; (THIS WAS DONE STRAIGHT-LINE SINCE
                STA        SCRTCH+1              ; WE'RE ADDING A TWO BYTE TO A THREE
                INY                              ; BYTE QUANTITY)
                LDA        (FCBPTR),Y
                STA        TPOSHI
                ADC        #0                    ; ADD IN REMAINING CARRY.
                STA        SCRTCH+2
                LDY        #FCBEOF+2             ; NOW TEST EOF AGAINST POSITION GENERATED
WEOFTST         LDA        SCRTCH-FCBEOF,Y
                CMP        (FCBPTR),Y            ; IS NEW POSITION > EOF?
                BCC        WRITE2                ; NO, PROCEED.
                BNE        WADJEOF               ; YES, ADJUST END OF FILE
                DEY
                CPY        #FCBEOF-1             ; HAVE WE COMPARED ALL TREE BYTES?
                BNE        WEOFTST               ; NO, TEST NEXT LOWEST.
WADJEOF         CLC                              ; ADJUST REQUEST TO WRITE UP TO (BUT
                LDY        #FCBEOF               ; NOT INCLUDING) END OF FILE.
WRTADJEOF       LDA        (FCBPTR),Y            ; SAVE OLD EOF IN CASE OF LATER ERROR
                STA        OLDEOF-FCBEOF,Y
                LDA        SCRTCH-FCBEOF,Y       ; RESULT=EOF
*
                STA        (FCBPTR),Y
                INY
                CPY        #FCBEOF+3
                BNE        WRTADJEOF
WRITE2          LDA        C.BYTES
                STA        RWREQL
                BNE        WRITE3                ; BRANCH IF WRITE REQUEST DEFINITELY NON-ZERO.
                CMP        C.BYTES+1
                BNE        WRITE3               ; BRANCH IF WRITE REQUEST<>ZERO
                STA        RWREQH
                JMP        WRITDONE             ; DO NOTHING.
*
                PAGE
WRITE3          LDA        C.BYTES+1
                STA        RWREQH
                LDA        C.OUTBUF             ; MOVE POINTER TO USERS BUFFER TO BFM
                STA        USRBUF               ; Z-PAGE AREA.
                LDA        C.OUTBUF+1
                STA        USRBUF+1             ; (SO IT MAY BE ADJUSTED WITHOUT LOOSING
                LDA        SISOUTBF             ; ORIGINAL ADDRESS.)
                STA        SISUSRBF
                LDY        #FCBSTYP             ; NOW FIND OUT IF IT'S A TREE WRITE OR OTHER.
                LDA        (FCBPTR),Y
                CMP        #TRETYP+1
                BCC        TWRITE               ; BRANCH IF A TREE FILE.
                JMP        ERRACCS              ; OTHEWISE RETURN AN ACCESS ERROR!
TWRITE          JSR        RDPOSN               ; READ BLOCK WE'RE
                BCS        WRITERROR
                LDY        #FCBSTAT
                LDA        (FCBPTR),Y
                AND        #DATALC+IDXALC+TOPALC
                BEQ        TREWRT1
                LDY        #0                   ; FIND OUT IF ENOUGH DISK SPACE IS AVAILABLE FOR
TWRTALC         INY                             ; INDEXES AND DATA BLOCK
                LSR        A
                BNE        TWRTALC
                STY        REQL
                STA        REQH
                JSR        TSFRBLK
                BCS        WRITERROR            ; PASS BACK ANY ERRORS.
                LDY        #FCBSTAT
                LDA        (FCBPTR),Y           ; NOW GET MORE SPECIFIC.
                AND        #TOPALC              ; ARE WE LACKING A TREE TOP?
                BEQ        TSTSAPWR             ; NO, TEST FOR LACK OF SAPLING LEVEL INDEX.
                JSR        TOPDOWN              ; GO ALLOCATE TREE TOP AND ADJUST FILE TYPE.
                BCC        DBLOKALC             ; CONTINUE WITH ALLOCATION OF DATA BLOCK.
WRITERROR       PHA                             ; SAVE ERROR
                LDY        #FCBEOF
WRITERR01       LDA        OLDEOF-FCBEOF,Y
                STA        (FCBPTR),Y           ; RESTORE OLD EOF UPON ERR
                INY
                CPY        #FCBEOF+3
                BNE        WRITERR01
                LDY        #FCBMARK
WRITERR02       LDA        OLDMARK-FCBMARK,Y
                STA        (FCBPTR),Y           ; AND RESTORE OLD MARK!
                INY
                CPY        #FCBMARK+3
                BNE        WRITERR02
                PLA
                SEC
                RTS                             ; ERROR RETURN
*
TWRITEGO        BVC        TWRITE               ; A PIGGY-BACK BACKWARD BRANCH
*
                PAGE
TSTSAPWR        LDA        (FCBPTR),Y           ; GET STATUS BYTE AGAIN.
                AND        #IDXALC              ; DO WE NEED A SAPLING LEVEL INDEX BLOCK?
                BEQ        DBLOKALC             ; NO, ASSUME IT'S JUST A DATA BLOCK NEEDED.
                JSR        SAPDOWN              ; GO ALLOCATE AN INDEX BLOCK AND UPDATE TREE TOP.
                BCS        WRITERROR            ; RETURN ANY ERRORS.
DBLOKALC        JSR        ALCWBLK              ; GO ALLOCATE FOR DATA BLOCK.
                BCS        WRITERROR
                LDA        TPOSHI               ; CALCULATE POSITION WITHIN INDEX BLOCK.
                LSR        A
                LDA        TPOSLH
                ROR        A
                TAY                             ; NOW PUT BLOCK ADDRESS INTO INDEX BLOCK
                INC        TINDX+1              ; HIGH BYTE FIRST.
                LDA        SCRTCH+1
                TAX
                STA        (TINDX),Y
                DEC        TINDX+1              ; (RESTORE POINTER TO LOWER PAGE OF INDEX BLOCK)
                LDA        SCRTCH               ; GET LOW BLOCK ADDRESS
                STA        (TINDX),Y            ; NOW STORE LOW ADDRESS.
                LDY        #FCBDATB             ; ALSO UPDATE FILE CONTROL BLOCK TO INDICATE
                STA        (FCBPTR),Y           ; THAT THIS BLOCK IS ALLOCATED.
                INY
                TXA                             ; GET HIGH ADDRESS AGAIN.
                STA        (FCBPTR),Y
                LDY        #FCBSTAT
                LDA        (FCBPTR),Y
                ORA        #IDXMOD
                AND        #$FF-DATALC-IDXALC-TOPALC ; CLEAR ALLOCATION REQUIREMENT BITS.
                STA        (FCBPTR),Y
TREWRT1         LDX        #USRBUF              ; LOCATE POINTER TO ADJUST <SRS 82.162>
                JSR        WRAPADJ              ; ADJUST FOR BANK CROSSING <SRS 82.162>
                JSR        PREPRW               ; WRITE ON
                JSR        WRTPART
                BVC        TWRITEGO
WRITDONE        JMP        RDPOSN               ; UPDATE FCB WITH NEW POSITION.
*
                PAGE
WRTPART         TXA
                BNE        WRPART               ; BRANCH IF REQUEST IS NOT A EVEN PAGES
                LDA        RWREQH               ; A CALL OF ZERO BYTES SHOULD NEVER GET HERE!
                BEQ        SETWRDNE             ; DO NOTHING!
*
```

```
                DEC        RWREQH
WRPART          DEX
                LDA        (USRBUF),Y          ; MOVE DATA FROM USER'S BUFFER
                STA        (POSPTR),Y          ; ONE BYTE AT A TIME.
                TXA
                BEQ        ENDWQCHK
WRPART2         INY                            ; PAGE CROSSED?
                BNE        WRPART              ; NO. MOVE NEXT BYTE.
                LDA        POSPTR+1            ; TEST FOR END OF BUFFER
                INC        USRBUF+1            ; BUT FIRST ADJUST USER BUFFER POINTER
                INC        TPOSLH              ; AND POSITION.
                BNE        WRPART3
                INC        TPOSHI
WRPART3         INC        POSPTR+1            ; AND SOS BUFFER HIGH ADDRESS.
                EOR        DATPTR+1            ; (CARRY HAS BEEN CLEVERLY UNDISTURBED.)
                BEQ        WRPART              ; BRANCH IF MORE TO WRITE TO BUFFER.
                CLV                            ; INDICATE NOT FINISHED.
                BVC        WRPRTDNE            ; BRANCH ALWAYS.
*
ENDWQCHK        LDA        RWREQH
                BEQ        WRTRQDNE            ; BRANCH IF REQEST SATISFIED.
                INY                            ; ARE WE DONE WITH THIS BLOCK OF DATA?
                BNE        ENDWCHK1            ; BRANCH IF NOT.
                LDA        POSPTR+1
                EOR        DATPTR+1            ; WHILE THIS IS REDUNDANT, IT'S NECESSARY FOR
                BNE        ENDWCHK2            ; PROPER ADJUSTMENT OF REQUEST COUNT.
ENDWCHK1        DEC        RWREQH              ; (NOT FINISHED- OK TO ADJUST HI BYTE.)
ENDWCHK2        DEY                            ; RESET MODIFIED Y
                JMP        WRPART2
*
WRTRQDNE        INY                            ; AND POSITION.
                BNE        SETWRDNE
                INC        USRBUF+1            ; BUMP POINTERS.
                INC        TPOSLH
                BNE        SETWRDNE
                INC        TPOSHI
SETWRDNE        BIT        SETVFLG             ; (SET V FLAG)
WRPRTDNE        STY        TPOSLL              ; SAVE LOW POSITION
                STX        RWREQL              ; AND REMAINDER OF REQUEST COUNT.
                PHP                            ; SAVE STATUSES
                LDY        #FCBSTAT
                LDA        (FCBPTR),Y
                ORA        #DATMOD+USEMOD
                STA        (FCBPTR),Y
                CLC                            ; ADJUST USER'S LOW BUFFER ADDRESS
                LDA        TPOSLL
                ADC        USRBUF
                STA        USRBUF
                BCC        WRPART4
                INC        USRBUF+1            ; ADJUST HI ADDRESS AS NEEDED.
```

```
WRPART4         JSR        FCBUSED              ; SET DIRECTORY FLUSH BIT
                PLP                             ; RESTORE RETURN STATUSES
                RTS
                PAGE
TOPDOWN         JSR        SWAPDOWN             ; FIRST MAKE CURRENT 1ST BLOCK AN ENTRY IN NEW TOP.
                BCS        TPDWNERR             ; RETURN ANY ERRORS
                LDY        #FCBSTYP             ; FIND OUT IF STORAGE TYPE HAS BEEN CHANGED TO 'TREE'.
                LDA        (FCBPTR),Y           ; (IF NOT, ASSUME IT WAS ORIGINALLY A SEED AND
                CMP        #TRETYP              ; BOTH LEVELS NEED TO BE BUILT.
                BEQ        TOPDWN1              ; OTHERWISE, ONLY AN INDEX NEED BE ALLOCATED)
                JSR        SWAPDOWN             ; MAKE PREVIOUS SWAP A SAP LEVEL INDEX BLOCK.
                BCS        TPDWNERR
TOPDWN1         JSR        ALCWBLK              ; GET ANOTHER BLOCK ADDRESS FOR THE SAP LEVEL INDEX.
                BCS        TPDWNERR
                LDA        TPOSHI               ; CALCULATE POSITION OF NEW INDEX BLOCK
                LSR        A                    ; IN THE TOP OF THE TREE.
                TAY
                LDA        SCRTCH               ; GET ADDRESS OF NEWLY ALOCATED INDEX BLOCK AGAIN
                TAX
                STA        (TINDX),Y
                INC        TINDX+1
                LDA        SCRTCH+1
                STA        (TINDX),Y            ; SAVE HI ADDRESS
                DEC        TINDX+1
                LDY        #FCBIDXB+1           ; MAKE NEWLY ALLOCATED BLOCK THE CURRENT INDEX BLOCK.
                STA        (FCBPTR),Y
                TXA
                DEY
                STA        (FCBPTR),Y
                JSR        WFCBFST              ; SAVE NEW TOP OF TREE.
                BCS        TPDWNERR
                JMP        ZTMPIDX              ; END BY RE-CLEARING CURRENT (NEW) INDEX BLOCK.
*
SAPDOWN         LDY        #FCBSTYP             ; FIND OUT IF WE'RE DEALING WITH A TREE
                LDA        (FCBPTR),Y           ; OR A SIMPLE SEED.
                CMP        #SEEDTYP             ; IF SEED THEN AN ADJUSTMENT TO FILE TYPE IS NECESSARY.
                BEQ        SAPDWN1              ; BRANCH IF SEED.
                JSR        RFCBFST              ; OTHERWISE READ IN TOP OF TREE.
                BCC        TOPDWN1              ; BRANCH IF NO ERROR.
TPDWNERR        RTS                             ; RETURN ERRORS
*
                PAGE
SAPDWN1         EQU        *                    ; MAKE CURRENT SEED INTO A SAPLING
*
SWAPDOWN        JSR        ALCWBLK              ; ALLOCATE A BLOCK BEFORE SWAP
                BCS        SWAPERR              ; RETURN ERRORS IMMEDIATELY.
                LDY        #FCBFRST             ; GET PREVIOUS FIRST BLOCK
                LDA        (FCBPTR),Y           ; ADDRESS INTO INDEX BLOCK.
                PHA                             ; SAVE TEMPORARLY WHILE SWAPPING IN NEW TOP INDEX
                LDA        SCRTCH               ; GET NEW BLOCK ADDRESS (LOW)
                TAX
                STA        (FCBPTR),Y
                INY
                LDA        (FCBPTR),Y
                PHA
                LDA        SCRTCH+1             ; AND HIGH ADDRESS TOO.
                STA        (FCBPTR),Y
                LDY        #FCBIDXB+1           ; MAKE NEW TOP ALSO THE CURRENT INDEX IN MEMORY.
                STA        (FCBPTR),Y
                TXA                             ; GET LOW ADDRESS AGAIN
                DEY
                STA        (FCBPTR),Y
                LDY        #0                   ; MAKE PREVIOUS THE FIRST ENTRY IN SUB INDEX
                INC        TINDX+1
                PLA
                STA        (TINDX),Y
                DEC        TINDX+1
                PLA
                STA        (TINDX),Y
                JSR        WFCBFST              ; SAVE NEW FILE TOP.
                BCS        SWAPERR
                LDY        #FCBSTYP             ; NOW ADJUST STORAGE TYPE
                LDA        #1                   ; BY ADDING 1 (THUS SEED BECOMES SAPLING BECOMES TREE)
                ADC        (FCBPTR),Y
                STA        (FCBPTR),Y
                LDY        #FCBSTAT
                LDA        (FCBPTR),Y           ; MARK STORAGE TYPE MODIFIED.
                ORA        #STPMOD
                STA        (FCBPTR),Y
                CLC                             ; RETURN 'NO ERROR' STATUS.
SWAPERR         RTS
*
                PAGE
ALCWBLK         JSR        ALC1BLK
                BCS        ALUSERR
                LDY        #FCBUSE
                LDA        (FCBPTR),Y           ; BUMP CURRENT USAGE COUNT BY 1.
                CLC
                ADC        #1
                STA        (FCBPTR),Y
                BCC        INCUSG1
                INY
                LDA        (FCBPTR),Y
                ADC        #0
                STA        (FCBPTR),Y
INCUSG1         LDY        #FCBSTAT             ; MARK USAGE AS MODIFIED.
                LDA        (FCBPTR),Y
                ORA        #USEMOD
                STA        (FCBPTR),Y
                CLC                             ; INDICATE NO ERROR
```

```
ALUSERR         RTS                                 ; ALL DONE
*
TSTWPROT        LDY        #FCBSTAT                 ; CHECK FOR A 'NEVER BEEN MODIFIED' CONDITION
                LDA        (FCBPTR),Y               ; GET STATUS BYTE
                AND        #USEMOD+DATMOD+IDXMOD+EOFMOD
                CLC                                 ; ANTICIPATE WRITE OK
                BNE        ALUSERR                  ; ORDINARY RTS
                LDY        #FCBDEVN                 ; GET FILE'S DEVICE NUMBER
                LDA        (FCBPTR),Y
                STA        DEVNUM                   ; GET CURRENT STATUS OF BLOCK DEVICE
TWRPROT1        LDA        #STATCMD
                STA        DHPCMD
                LDA        #STATSUB                 ; STORE SUB COMMAND OF STATUS CALL
                STA        DSTATREQ
                LDA        #>TWRCODE
                STA        DSTATBFL                 ; FETCH RETURN CODE IN SCRATCH AREA
                LDA        #<TWRCODE
                STA        DSTATBFH
                LDA        #0                       ; MAKE SURE REGULAR RAM IS SELECTED (NO BANKS)
                STA        SISDSTAT
                STA        SERR                     ; CLEAR GLOBAL ERROR FLAG
                LDA        DEVNUM                   ; SET UP LAST PARM
                STA        UNITNUM                  ; FOR DEVICE CALL
                JSR        DMGR                     ; MAKE THE EXTERNAL CALL
                BCS        WPROTRET                 ; RETURN ANY SPECIFIC ERRORS
                LDA        TWRCODE                  ; GET STATUS BYTE
                LSR        A                        ; SHIFT WRITE PROTECT STATE INTO CARRY
                LSR        A
                LDA        #XNOWRITE                ; ANTICIPATE WRITE PROTECTED.
                RTS                                 ; CARRY IS INDETERMINATE
WPROTRET        EQU        *
                CMP        #XDISKSW                 ; IF EXPLICITLY DISK SWITCH
                BNE        WPROT1                   ; BRANCH IF XNODRIVE OR XNOWRITE
                STA        DSWGLOB                  ; IF DISKSW, FLAG UNTIL ENTIRE OPERATION IS COMPLETE
                CLC
                RTS                                 ; DISKSWITCH DOESNT SET CARRY
WPROT1          SEC
                RTS
DSWGLOB         DS         1                        ; DISK SWITCH GLOBAL
TWRCODE         DS         1                        ; A RARE EMBEDDED TEMP STORE
*
                PAGE
*
* MEMORY 'WRAP-AROUND' ADJUST ROUTINE.  THIS ROUTINE ADJUSTS
* ADDRESSES THAT CROSS BANK PAIR BOUNDARIES.  ON ENTRY, X CONTAINS
* THE OFFSET OF THE ZERO PAGE EXTENDED POINTER TO BE ADJUSTED.
* ON EXIT, THE POINTER WILL HAVE BEEN ADJUSTED, IF NECESSARY,
* AND THE ASSOCIATED X-BYTE WILL ALSO HAVE BEEN ADJUSTED.
* ONLY ADDRESSES IN THE RANGE $8200-$8E00 WILL BE ADJUSTED.
*
```

```
* UPON EXIT, A CONTAINS HIGH BYTE OF ADDRESS & Y CONTAINS UPDATED X-BYTE.
* THIS ROUTINE LEAVES X UNCHANGED.
*
WRAPADJ         LDA        1,X                   ; GET HIGH ADDRESS BYTE <SRS 82.162>
                LDY        SISTER+1,X            ; CHECK X-BYTE <SRS 82.162>
                BPL        WRAPDNE               ; NOT AN EXTENDED ADDRESS. <SRS 82.162>
                CMP        #$82                  ; DOES IT NEED UPDATING? <SRS 82.162>
                BCC        WRAPDNE               ; NO <SRS 82.162>
                CPY        #$8F                  ; SPECIAL BANK? <SRS 82.162>
                BCS        WRAPDNE               ; NO <SRS 82.162>
                AND        #$7F                  ; ADJUST THE ADDRESS <SRS 82.162>
                STA        1,X                   ; UPDATE <SRS 82.162>
                INC        SISTER+1,X            ; INCREMENT X-BYTE <SRS 82.162>
                INY                              ; UPDATE Y ALSO <SRS 82.162>
*
WRAPDNE         RTS                              ; RETURN VALID HIGH ADDRESS AND BANK BYTE.

                CHN        CLOSE/EOF,4,2

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: READ.WRITE
*************************************************************************



```

```
==============================================================================
DOCUMENT :SOS1.3.4of5.FOUR:SOS.SWAPOUT.IN.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SWAPOUT.IN
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

SWAPOUT         EQU        *
*
* SWAP OUT A VOLUME LOGGED ON A DEVICE
* INPUT ARGUMENT: DEVICE NUMBER "A"
* (STORED AS "DEVNUM")
* OUTPUT ARGUMENT: NONE
* CONDITION CODE: CARRY SET USER DID NOT COMPLY WITH REQUEST
*
* SAVE VCBPTR, FCBPTR, DEVNUM ON STACK
* 1) FIND UNSWAPPED VOLUME IN VCB
* 2) IF DIRTY BIT MAP FOR THIS VOLUME THEN DO
*    IF NOT ONLINE, REQUEST USER TO INSERT
*    IF REQUEST DENIED, UNCONDITIONALLY  CLOSE ALL FILES ON THIS VOLUME AND RTS
*    IF ONLINE, UPDATE AND RELEASE BIT MAP
*    DOEND
* 3) SWAP IT (MARK VCBSWAP FIELD $80, MARK ALL FILES ON THIS VOLUME WITH SWAP MARK $8X WHERE X=VCB ENTRY)
*    "VCB ENTRY" DEFINED AS: HIGH ORDER NIBBLE OF LOW ORDER BYTE OF ENTRIES VCB ADDRESS
*    RESTORE VCBPTR, FCBPTR
* RTS
*
                TAX                             ; SAVE DEVICE NUMBER
                JSR        SAVECBS
                STX        DEVNUM               ; PERMANENTLY
SWAPOUTX        JSR        DEVVCB               ; FIND MATCHING UNSWAPPED ACTIVE VCB ENTRY (BY DEVNUM)
                BCS        SORTS                ; NO FIND--RETURN WITHOUT ERROR
                LDY        #VCBSTAT
                LDA        (VCBPTR),Y           ; GET STATUS OF FILES ON THIS VOLUME
                BPL        UNLOG                ; IF NO OPEN FILES, JUST THROW VOLUME AWAY
                LDA        DEVNUM               ; DIRTY BM EXIST ON THIS VOLUME?
                LDX        #0
                CMP        BMADEV,X             ; IN BIT MAP "A"?
                BEQ        FDIRBM               ; BRANCH IF YES
                LDX        #6                   ; BIT MAP HEADER TABLE SIZE
                CMP        BMADEV,X             ; IN BIT MAP "B"?
                BEQ        FDIRBM               ; BRANCH IF YES
                JMP        MARKSWAP             ; NO NEED TO WRITE BIT MAP
FDIRBM          LDA        BMASTAT,X            ; IS BIT MAP DIRTY?
                BPL        MARKSWAP             ; BRANCH IF NOT
GETVOL          JSR        VERFYVOL             ; IS THE CORRECT VOLUME ON LINE NOW?
                BCC        VONLINE              ; BRANCH IF YES
                JSR        USRREQ               ; OTHERWISE, REQUEST USER INSERTION
                BCC        GETVOL               ; AND VERIFY IT AGAIN
                JSR        CLOSEU               ; USER SAID "NO": UNCONDITIONALLY CLOSE VOLUME
                JSR        RESTCBS
                SEC
                RTS                             ; ERROR RETURN TO CALLER
VONLINE         LDX        DEVNUM               ; UPDATE THE
                JSR        UPBMAP               ; DIRTY BIT MAP
MARKSWAP        LDA        VCBPTR               ; CALCULATE
                LSR        A                    ; SWAP BYTE
                LSR        A                    ; AND
                LSR        A                    ; MARK ALL FILES
                LSR        A                    ; BELONGING TO THIS VOLUME
                SEC                             ; AS SWAPPED OUT
                ORA        #$80
                PHA                             ; SAVE SWAP BYTE
                JSR        FCBSCAN
                PLA                             ; MARK VCBSWAP
                LDY        #VCBSWAP             ; BYTE
                STA        (VCBPTR),Y
SORTS           JSR        RESTCBS              ; RESTORE FCBPTR, VCBPTR, DEVNUM
                CLC
                RTS                             ; SUCCESSFUL SWAP OUT
UNLOG           LDA        #0
                STA        VCB,X                ; UNLOG VOLUME
                BEQ        SORTS                ; SWAP THE EASY WAY! (BRANCH ALWAYS)
*
*
*
SWAPIN          EQU        *
*
* UNSWAP A VOLUME AND ALL ITS FILES
*
* INPUT ARGUMENT: VOLUME NAME (VCBPTR)
* OUTPUT ARGUMENT: NONE
* CONDITION CODE: CARRY SET : USER DID NOT COMPLY WITH REQUEST
*
* SAVE VCBPTR, FCBPTR ON STACK
* 1) FIND SWAPPED VOLUME IN VCB, IF NOT FOUND, THEN RTS.
* 2) IF ANOTHER UNSWAPPED VOLUME ON DEVICE, THEN SWAP IT
* 3) VERIFY UNSWAPPED VOLUME, IF NOT OK THEN REQUEST INSERTION
* 4) UNMARK VCB'S AND FCB'S
* RTS
                JSR        SAVECBS              ; SAVE FCB, VCB POINTERS, DEVNUM
                LDY        #VCBNML              ; MAKE SURE VOLUME
                LDA        (VCBPTR),Y           ; IS AT LEAST OPEN
                BEQ        USRTS                ; BRANCH IF NOT RIGHT BACK TO CALLER
                LDY        #VCBSWAP             ; SEE IF
                LDA        (VCBPTR),Y           ; CURRENTLY SWAPPED
                BEQ        USRTS                ; IF NOT, RETURN IMMEDIATELY TO CALLER
                LDY        #VCBDEV              ; SAVE DEVICE NUMBER
                LDA        (VCBPTR),Y
                STA        DEVNUM
                PHA                             ; SAVE DEVNUM AGAIN (SWAPOUTX TRASHES DEVNUM ON RETURN)
                JSR        SWAPOUTX             ; AND MAKE SURE ANY CURRENT ACTIVE VOLUME IS SWAPPED OUT (NOTICE ENTRY POINT)
                PLA                             ; RECALL CURRENT DEVICE NUMBER
                STA        DEVNUM               ; AND SAVE IT TO ITS PROPER PLACE
SI1             JSR        VERFYVOL             ; VERIFY THE CURRENT VOLUME MOUNTED
                BCC        UNMARK               ; IF THE RIGHT ONE, GO MARK IT AS UNSWAPPED
                JSR        USRREQ               ; ELSE REQUEST USER TO INSERT
                BCC        SI1                  ; USER SAID 'OK'
                JSR        CLOSEU               ; OTHERWISE UNCONDITIONALLY CLOSE
                JSR        RESTCBS
                SEC
                RTS
UNMARK          LDY        #VCBSWAP             ; FETCH
                LDA        (VCBPTR),Y           ; VOLUME
                PHA                             ; SWAP BYTE
                LDA        #0                   ; BUT CLEAR
                STA        (VCBPTR),Y           ; VOLUME SWAP
                PLA
                CLC                             ; "UNSWAPPED"
                JSR        FCBSCAN
                LDA        DEVNUM               ; MAKE SURE BIT MAPS
                JSR        CLEARBMS             ; ARE MARKED AS INVALID ON THIS DEVICE
USRTS           JSR        RESTCBS              ; RESTORE VCB, FCB PTRS
                CLC                             ; NO ERRORS
                RTS
*
SAVEPTRS        DS         5                    ; A RARE EMBEDDED TEMP SAVE AREA, USED ONLY BY ...
*
*
SAVECBS         EQU        *                    ; SAVE FCBPTR, VCBPTR IN A TEMP SAVE AREA
                LDA        VCBPTR
                STA        SAVEPTRS
                LDA        VCBPTR+1
                STA        SAVEPTRS+1
                LDA        FCBPTR
                STA        SAVEPTRS+2
                LDA        FCBPTR+1
                STA        SAVEPTRS+3
                LDA        DEVNUM
                STA        SAVEPTRS+4
                RTS
*
RESTCBS         EQU        *                    ; RESTORE FCBPTR, VCBPTR
* NOTICE THERE EXISTS A SEQUENCE OF CALLS (SWAPIN, WHICH MAY CALL SWAPOUT) THAT JSR'S TO SAVECBS ONCE BUT JSR'S RESTCBS TWICE.
                LDA        SAVEPTRS
                STA        VCBPTR
                LDA        SAVEPTRS+1
                STA        VCBPTR+1
                LDA        SAVEPTRS+2
                STA        FCBPTR
                LDA        SAVEPTRS+3
                STA        FCBPTR+1
                LDA        SAVEPTRS+4
                STA        DEVNUM
                RTS
*
*
* MARK ALL FILES BELONGING TO A VOLUME
* AS SWAPPED-IN OR SWAPPED-OUT.
*
* INPUT ARGS: DEVNUM -- DEVICE NUMBER OF MOUNTED VOLUME
*             A REGISTER - SWAP BYTE
*             CARRY -- CARRY FLAG SET MEANS SWAP OUT; ELSE SWAP IN
*
* OUTPUT ARGS: NONE
* GLOBALS AFFECTED: FCB, FCBPTR
* REGISTER STATUS: SCRAMBLED
*
FCBSCAN         EQU        *                    ; MARK FILES BELONGING TO VOLUME AS SWAPPED OR UNSWAPPED
*
                TAX                             ; SAVE SWAP BYTE
                LDY        FCBADDRH             ; POINT TO
                STY        FCBPTR+1             ; BEGINNING TO FCB
                LDY        #0
                STY        FCBPTR
                BCS        FCBOUT               ; SWAP OUT A VOLUMES FILES
FCBIN           EQU        *                    ; SWAPIN A VOLUMES FILES
                JSR        FCBFETCH             ; GET NEXT ACTIVE FCB CANDIDATE
                BCS        FCBRTS               ; NO MORE FILES TO PROCESS
                LDY        #FCBSWAP
                TXA
                CMP        (FCBPTR),Y           ; SWAP BYTES MATCH?
                BNE        FCBIN1               ; BRANCH IF NOT
                LDA        #0
                STA        (FCBPTR),Y           ; MARK FILE AS SWAPPED IN
FCBIN1          JSR        NEXTFCB              ; ADVANCE FCB POINTER
                BCS        FCBRTS               ; NO MORE TO LOOK AT
                JMP        FCBIN                ; AND LOOK AT NEXT FILE
*
FCBOUT          EQU        *                    ; SWAPPED OUT A VOLUMES FILES
                JSR        FCBFETCH             ; GET NEXT ACTIVE FILE IN FCB
                BCS        FCBRTS               ; NO MORE FILES -- RETURN TO USER
                LDY        #FCBSWAP             ; COMPARE
                LDA        (FCBPTR),Y
                BNE        FCBOUT1              ; ALREADY SWAPPED OUT
                TXA
                STA        (FCBPTR),Y           ; MARK AS SWAPPED
FCBOUT1         JSR        NEXTFCB              ; ADVANCE FCB POINTER
                BCS        FCBRTS
                JMP        FCBOUT               ; SWAP OUT NEXT FILE
*
FCBRTS          RTS
FCBFETCH        EQU        *                    ; GET NEXT ACTIVE FILE FROM FCB
* X REGISTER MUST NOT BE DISTURBED
* USES FCBPTR
                LDY        #FCBDEVN             ; MAKE
                LDA        (FCBPTR),Y           ; SURE DEVICE
                CMP        DEVNUM               ; MATCHES
                BNE        NEXTFCB
                LDY        #FCBREFN             ; MAKE SURE FILE IS ACTIVE
                LDA        (FCBPTR),Y
                BEQ        NEXTFCB              ; BRANCH IF NOT
                CLC
                RTS                             ; RETURN WITH CARRY CLEAR SHOWING AN ACTIVE FILE
NEXTFCB         LDA        FCBPTR
                CLC
                ADC        #$20                 ; FCB ENTRY SIZE
                STA        FCBPTR
                BCC        FCBFETCH             ; BRANCH IF NO PAGE CROSS
                LDA        FCBPTR+1
                INC        FCBPTR+1             ; SECOND PAGE
                CMP        FCBADDRH
                BEQ        FCBFETCH             ; LOOK AT PAGE TWO
NEXTEND         SEC
                RTS                             ; SHOW NO MORE FILES TO LOOK AT
USRREQ          EQU        *                    ; OPERATOR CONSOLE MESSAGE INTERFACE
* PRODUCES A MESSAGE REQUESTING
* THE SYSTEM OPERATOR TO MOUNT THE VOLUME
* SPECIFIED BY "VCBPTR" ON DEVICE SPECIFIED
* BY DEVNUM.  THIS MODULE INSISTS
* UPON THE CORRECT OPERATOR ACTION
* UPON THREE FAILURES TO COMPLY,
* THE MODULE WILL SIGNIFY FAILURE WITH
* CARRY SET.  IF THE CORRECT ACTION IS TAKEN,
* CARRY WILL BE RETURNED CLEAR
*
* INPUT ARGS: VOLUME NAME (VCBPTR)
*             DEVICE NUMBER (DEVNUM)
*
* OUTPUT ARGS: CC = OPERATOR COMPLIED WITH REQUESTED ACTION
*              CS = OPERATOR COULDN'T/DIDN'T COMPLY
*
* GLOBALS AFFECTED: NONE
*
* STATUS OF REGISTERS: UNCERTAIN
*
```

```
VNML            EQU        ZPGTEMP              ; VOLUME NAME LENGTH
                LDY        #VCBNML              ; IF ILLEGAL VCB
                LDA        (VCBPTR),Y           ; GET OUT QUICK
                BEQ        NEXTEND              ; BRANCH TO SEC RTS
                LDX        #$E                  ; LENGTH OF NAMED AREA-1
                LDA        #$0                  ; NULLS
UR1             STA        MDEV,X               ; BOTH CLEAR
                STA        MVOL,X               ; IN ONE LOOP
                DEX
                BPL        UR1
*
* DO A D-INFO TO FETCH THE DEVICE NAME
*
                LDA        #5                   ; DO ALL
                STA        $C0                  ; NECESSARY
                LDA        DEVNUM               ; HOUSKEEPING
                STA        $C1                  ; TO SET UP
                LDA        #>MDEV-1             ; A DEVICE MANAGER CALL
                STA        $C2
                LDA        #<MDEV-1
                STA        $C3
                LDA        #$8F                 ; EXTEND BYTE
                STA        $14C3
                LDA        #0
                STA        $14C2
                STA        $C4
                STA        $C5
                STA        $C6                  ; ZERO SUPERFLUOUS PARMS
                STA        URDERR               ; RESET FAILURE COUNT
                JSR        RPEATIO0             ; GET INFO FROM BOBS CODE
                LDA        #$20                 ; "SPACE" RESTORED
                STA        MDEV-1               ; RESTORED
                LDY        #VCBNML
                LDA        (VCBPTR),Y           ; LENGTH OF VOLUME NAME
                STA        VNML                 ; SAVED FOR WORK
                LDA        #0
                TAX
                LDY        #VCBNAM              ; POINT TO BEGINNING OF VOLUME NAME
UR2             LDA        (VCBPTR),Y
                STA        MVOL,X
                INX
                INY                             ; VOLUME NAME MOVED
                DEC        VNML                 ; TO MESSAGE BUFFER
                BNE        UR2                  ; CHARACTER BY CHARACTER
URDU            LDX        #>UMB                ; PASS THE AREA'S ADDR
                LDY        #<UMB                ; IN X AND Y REGS,LOW, HIGH)
                JSR        OPMSGRPLY            ; HAVE MESSAGE SYSTEM PRINT IT
                JSR        VERFYVOL             ; DID THE USER COMPLY?
                BCS        URDU1                ; BRANCH IF NOT
                RTS                             ; EXIT--CARRY IS CLEAR
```

```
URDU1           INC        URDERR               ; COLLECT USER ERRORS
                LDA        URDERR
                CMP        #3                   ; ONLY THREE TRIES ALLOWED
                BCC        URDU                 ; RETRY MESSAGE IF LESS THAN THREE TRIES
                RTS                             ; OTHERWISE RETURN WITH CARRY SET
*
*
*
*
*
* CLOSE UNCONDITIONAL
*
* (USER HAS REPLIED 'N' TO A VOLUME MOUNT REQUEST
* CLOSE ALL FILES ON VOLUME/UNLOG VOLUME
*
* INPUT ARGUMENT: (VCBPTR)
* OUTPUT ARGUMENT: NONE
*
CLOSEU          EQU        *
VSWA            EQU        ZPGTEMP              ; THE 'SWAP BYTE' STORED HERE
                LDY        #VCBDEV              ; FETCH
                LDA        (VCBPTR),Y           ; THE DEVICE NUMBER
                STA        DEVNUM               ; OF THIS VOLUME & SAVE IT
                LDY        #VCBSWAP             ; FETCH THE
                LDA        (VCBPTR),Y           ; SWAP BYTE
                STA        VSWA                 ; SAVE FOR REFERENCE, TOO
                LDA        #0
                LDY        #VCBNML              ; UNLOG THE VOLUME
                STA        (VCBPTR),Y           ; BY SETTING LEN OF VOL NAME TO ZERO
                LDY        #VCBSWAP
                STA        (VCBPTR),Y           ; TURN OFF SWAP FLAG
                LDY        FCBADDRH             ; SET UP FCB SCAN FROM BEGINNING OF FCB
                STY        FCBPTR+1
                LDY        #0
                STY        FCBPTR
VFCBLOP         LDY        #FCBDEVN             ; FETCH
                LDA        (FCBPTR),Y           ; THE DEVICE
                CMP        DEVNUM               ; NUMBER AND SEE IF A MATCH
                BNE        VFCBNXT              ; BRANCH IF NO MATCH
                LDY        #FCBREFN             ; SEE EVEN IF FILE OPEN
                LDA        (FCBPTR),Y
                BEQ        VFCBNXT              ; BRANCH IF NOT
                LDY        #FCBSWAP             ; CHECK TO SEE IF ATTACHED
                LDA        (FCBPTR),Y           ; TO SAME VOLUME
                CMP        VSWA
                BNE        VFCBNXT              ; BRANCH IF NOT
                LDY        #FCBBUFN             ; RELEASE
                LDA        (FCBPTR),Y           ; ANY
                JSR        RELBUF               ; BUFFERS ASSOCIATED
                LDY        #FCBSWAP             ; AND CLEAR
                LDA        #0                   ; THE SWAP BYTE
                STA        (FCBPTR),Y
                LDY        #FCBREFN             ; AND FINALLY
                STA        (FCBPTR),Y           ; SAY 'CLOSED'
VFCBNXT         LDA        FCBPTR
                CLC
                ADC        #$20                 ; FCB ENTRY SIZE
                STA        FCBPTR
                BCC        VFCBLOP
                LDA        FCBPTR+1
                INC        FCBPTR+1             ; LOOK AT SECOND PAGE
                CMP        FCBADDRH
                BEQ        VFCBLOP              ; CHECK PAGE TWO OF FCB
                RTS                             ; RETURN TO USER W/O ERROR
*
FCBUSED         EQU        *                    ; MARK AS FCB AS DIRTY SO
*                                                 THE DIRECTORY WILL BE FLUSHED ON 'FLUSH'
                STY        ZPGTEMP
                PHA                             ; SAVE REGS
                LDY        #FCBDIRTY
                LDA        (FCBPTR),Y           ; FETCH CURRENT FCBDIRTY BYTE
                ORA        #FCBMOD              ; MARK FCB AS DIRTY
                STA        (FCBPTR),Y           ; SAVE IT BACK
                PLA
                LDY        ZPGTEMP              ; AND RESTORE REGS
                RTS
*
URDERR          DS         1                    ; ERROR COUNT FOR USRREQ
*
*
UMB             EQU        *
                DFB        $49,$6E,$73,$65,$72,$74,$20
                DFB        $76,$6F,$6C,$75,$6D,$65
                DFB        $3A,$20              ; "INSERT VOLUME: "
MVOL            DS         15
                DFB        $0D                  ; CR LINE TERMINATOR
                DFB        $20,$20,$20,$20,$69,$6E,$20
                DFB        $64,$65,$76,$69,$63,$65
                DFB        $3A,$20              ; "    IN DEVICE: "
MDEV            DS         15
                DFB        $0D                  ; CR LINE TERMINATOR
                DFB        $74,$68,$65,$6E,$20,$70,$72
                DFB        $65,$73,$73,$20,$74,$68,$65,$20
                DFB        $41,$4C,$50,$48,$41,$20,$4C
                DFB        $4F,$43,$4B,$20,$6B,$65,$79
                DFB        $20,$74,$77,$69,$63,$65
* "THEN PRESS THE ALPHA LOCK KEY TWICE"
* FOLLOWED WITH $FF MESSAGE TERMINATOR (HIGH BIT SIGNIFICANT)
                DFB        $FF                  ; MESSAGE TERMINATOR (HIGH BIT)
*
```

```
ZZLEN           EQU        *-ZZORG
ZZEND           EQU        *
                IFNE       ZZLEN-LENBFM
                FAIL       2,"SOSORG             FILE IS INCORRECT FORMBFM"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SWAPOUT.IN
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.C.BI2.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: C.BI2
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

:TABS 17,23,40
::PR#1,L58        132N
SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: C.BI2
*************************************************************************

```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.C.S.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: C.S
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

:TABS 17,23,40
::PR#1,L58        132N
SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
END

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: C.S
*************************************************************************

```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.C3.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: C3
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

:TABS 17,23,40
::PR#1,L58        132N
SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
END

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: C3
*************************************************************************

```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.COMP.OPR.IPL.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.OPR.IPL
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

:TABS 17,23,40
::PR#1,L58        132N
SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
SL4:DR1:A,6,1
END

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.OPR.IPL
*************************************************************************

```

```
================================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.COMP.SOS.NOLIST.TEXT
================================================================================

***************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.SOS.NOLIST
***************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

:TABS 17,23,40
SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
SL4:DR1:ASM INIT.SRC,INIT.OBJ,6,1
SL4:DR1:ASM SYSGLOB.SRC,SYSGLOB.OBJ,6,1
SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
SL4:DR1:ASM UMGR.SRC,UMGR.OBJ,6,1
SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
END

***************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.SOS.NOLIST
***************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.COMPILE.BFM.TEXT
==============================================================================

**************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.BFM
**************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

:T 17,23,40
::PR#1,L58        132N
::SL4:DR1:ASM PRINT,BFM.OBJ,6,1
::END

**************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.BFM
**************************************************************************

```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.COMPILE.SOS.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.SOS
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

:TABS 17,23,40
::PR#1,L58       132N
SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
SL4:DR1:ASM INIT.SRC,INIT.OBJ,6,1
SL4:DR1:ASM SYSGLOB.SRC,SYSGLOB.OBJ,6,1
SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
SL4:DR2:ASM UMGR.SRC,UMGR.OBJ,6,1
SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
::END

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.SOS
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.FEB01.1982.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: FEB01.1982
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
SL4:DR1:ASM INIT.SRC,INIT.OBJ,6,1
SL4:DR1:ASM SYSGLOB.SRC,SYSGLOB.OBJ,6,1
SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
SL4:DR1:ASM UMGR.SRC,UMGR.OBJ,6,1
SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
END

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: FEB01.1982
*************************************************************************

```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.LC.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: LC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                IFNE       ZZLEN-LEN????
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR ??????"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: LC
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.LCHK.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: LCHK
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                INCLUDE    SOSORG,6,1,254
                ORG        ???????
---------------------------------
                IFNE       ZZLEN-LEN????
                FAIL       2,"SOSORG            FILE IS INCORRECT FOR ??????"
                FIN

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: LCHK
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.PUBLICRELEASE.TEXT
==============================================================================

***************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: PUBLICRELEASE
***************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

:T 15,19,32
::PR#1,L58        132N
::SL4:DR1:ASM PRINT,BFM.OBJ,S6,D1
::END

***************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: PUBLICRELEASE
***************************************************************************

```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.SOS.BLOAD.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.BLOAD
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

MON I
CALL-151
1600:0
1601<1600.93FEM
3D0G
MON I
BLOAD SOSLDR.ABS,A$2000
BLOAD INIT.ABS,A$2AF8
BLOAD SYSGLOB.ABS,A$2CF8
BLOAD BFM.INIT2.ABS,A$2E00
BLOAD BFM.ABS,A$3200
BLOAD OPRMSG.ABS,A$5466
BLOAD IPL.ABS,A$55C0
BLOAD UMGR.ABS,A$5A8B
BLOAD DISK3.ABS,A$5E99
BLOAD SYSERR.ABS,A$6404
BLOAD DEVMGR.ABS,A$64D9
BLOAD SCMGR.ABS,A$665E
BLOAD FMGR.ABS,A$68F4
BLOAD CFMGR.ABS,A$6955
BLOAD BUFMGR.ABS,A$6B52
BLOAD MEMMGR.ABS,A$6E6E
NOMON I

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.BLOAD
*************************************************************************
```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.SOS.LINK.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.LINK
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

SOSLDR.OBJ
INIT.OBJ
SYSGLOB.OBJ
BFM.INIT2.OBJ
BFM.OBJ
OPRMSG.OBJ
IPL.OBJ
UMGR.OBJ
DISK3.OBJ
SYSERR.OBJ
SCMGR.OBJ
FMGR.OBJ
CFMGR.OBJ
DEVMGR.OBJ
BUFMGR.OBJ
MEMMGR.OBJ
END

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.LINK
*************************************************************************

```

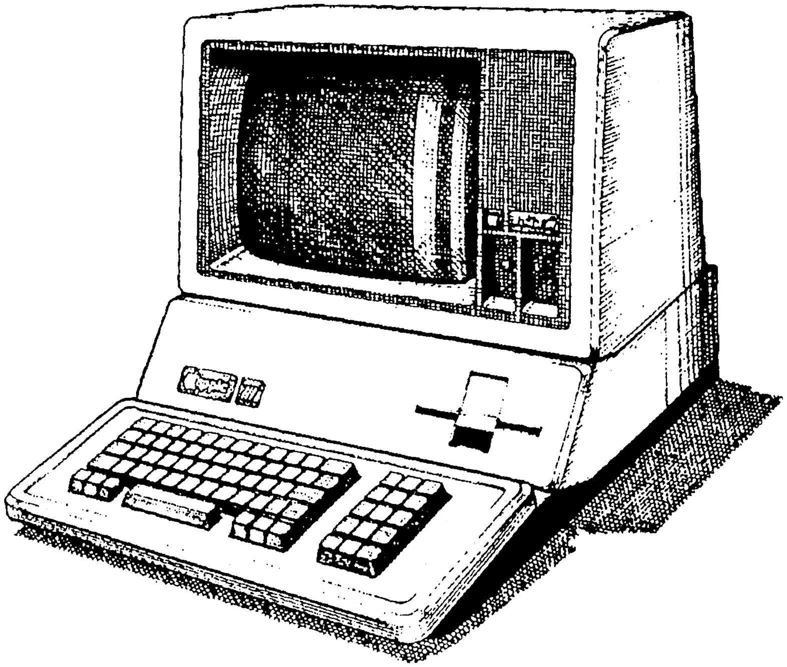
```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.SOS.RENAME.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.RENAME
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

MON I
RENAME SOSLDR.OBJ.ABS,SOSLDR.ABS
RENAME INIT.OBJ.ABS,INIT.ABS
RENAME SYSGLOB.OBJ.ABS,SYSGLOB.ABS
RENAME OPRMSG.OBJ.ABS,OPRMSG.ABS
RENAME BFM.OBJ.ABS,BFM.ABS
RENAME BFM.INIT2.OBJ.ABS,BFM.INIT2.ABS
RENAME IPL.OBJ.ABS,IPL.ABS
RENAME UMGR.OBJ.ABS,UMGR.ABS
RENAME DISK3.OBJ.ABS,DISK3.ABS
RENAME SYSERR.OBJ.ABS,SYSERR.ABS
RENAME SCMGR.OBJ.ABS,SCMGR.ABS
RENAME FMGR.OBJ.ABS,FMGR.ABS
RENAME CFMGR.OBJ.ABS,CFMGR.ABS
RENAME DEVMGR.OBJ.ABS,DEVMGR.ABS
RENAME BUFMGR.OBJ.ABS,BUFMGR.ABS
RENAME MEMMGR.OBJ.ABS,MEMMGR.ABS
NOMON I

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.RENAME
*************************************************************************

```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.SOSORG.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: SOSORG
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

                REP        100
*  SOS KERNEL MODULE ORIGINS
ORGLODR         EQU        $1E00                 ; ORIGIN OF SOS LOADER
ORGINIT         EQU        $28F8                 ; ORIGIN OF INIT
ORGGLOB         EQU        $18FC                 ; ORIGIN OF SYSGLOB
ORGBFMI         EQU        $B800                 ; ORIGIN OF BFM.INIT2 & BITMAPS
ORGBFM          EQU        $BC00                 ; ORIGIN OF BFM
ORGPATCH        EQU        $DE66                 ; ORIGIN OF PATCH AREA
ORGOMSG         EQU        $DE66                 ; ORIGIN OF OPRMSG
ORGIPL          EQU        $DFC0                 ; ORIGIN OF IPL
ORGUMGR         EQU        $E48B                 ; ORIGIN OF UMGR
ORGDISK3        EQU        $E899                 ; ORIGIN OF DISK3
ORGSERR         EQU        $EE04                 ; ORIGIN OF SYSERR
ORGDMGR         EQU        $EED9                 ; ORIGIN OF DEVMGR
ORGSCMGR        EQU        $F05E                 ; ORIGIN OF SCMGR
ORGFMGR         EQU        $F2F4                 ; ORIGIN OF FMGR
ORGCFM          EQU        $F355                 ; ORIGIN OF CFMGR
ORGBUFMG        EQU        $F552                 ; ORIGIN OF BUFMGR
ORGMEMMG        EQU        $F86E                 ; ORIGIN OF MEMMGR
ORGEND          EQU        $FFBF                 ; END MARKER
                REP        100
*  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
LENLODR         EQU        ORGINIT-ORGLODR       ; LENGTH OF SOS LOADER
LENINIT         EQU        $01B2                 ; LENGTH OF INIT
LENBFMI         EQU        ORGBFM-ORGBFMI        ; LENGTH OF BFM.INIT2 & BITMAPS
LENBFM          EQU        ORGPATCH-ORGBFM       ; LENGTH OF BFM
LENPATCH        EQU        ORGOMSG-ORGPATCH      ; LENGTH OF PATCH AREA
LENOMSG         EQU        ORGIPL-ORGOMSG        ; LENGTH OF OPRMSG
LENIPL          EQU        ORGUMGR-ORGIPL        ; LENGTH OF IPL
LENUMGR         EQU        ORGDISK3-ORGUMGR      ; LENGTH OF UMGR
LENDISK3        EQU        ORGSERR-ORGDISK3      ; LENGTH OF DISK3
LENSERR         EQU        ORGDMGR-ORGSERR       ; LENGTH OF SYSERR
LENDMGR         EQU        ORGSCMGR-ORGDMGR      ; LENGTH OF DEVMGR
LENSCMGR        EQU        ORGFMGR-ORGSCMGR      ; LENGTH OF SCMGR
LENFMGR         EQU        ORGCFM-ORGFMGR        ; LENGTH OF FMGR
LENCFM          EQU        ORGBUFMG-ORGCFM       ; ORIGIN OF CFMGR
LENBUFMG        EQU        ORGMEMMG-ORGBUFMG     ; LENGTH OF BUFMGR
LENMEMMG        EQU        ORGEND-ORGMEMMG       ; LENGTH OF MEMMGR
                REP        100
*      SOS BLOAD ADDRESSES
```

```
BLALODR         EQU        $2000                ; BLOAD ADDRESS OF SOS LOADER
BLAINIT         EQU        BLALODR+LENLODR      ; BLOAD ADDRESS OF INIT
BLAGLOB         EQU        $2CF8                ; BLOAD ADDRESS OF SYSGLOB
BLABFMI         EQU        $2E00                ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
BLABFM          EQU        $3200                ; BLOAD ADDRESS OF BFM
BLAPATCH        EQU        BLABFM+LENBFM        ; BLOAD ADDRESS OF PATCH AREA
BLAOMSG         EQU        BLAPATCH+LENPATCH    ; BLOAD ADDRESS OF OPRMSG
BLAIPL          EQU        BLAOMSG+LENOMSG      ; BLOAD ADDRESS OF IPL
BLAUMGR         EQU        BLAIPL+LENIPL        ; BLOAD ADDRESS OF UMGR
BLADISK3        EQU        BLAUMGR+LENUMGR      ; BLOAD ADDRESS OF DISK3
BLASERR         EQU        BLADISK3+LENDISK3    ; BLOAD ADDRESS OF SYSERR
BLADMGR         EQU        BLASERR+LENSERR      ; BLOAD ADDRESS OF DEVMGR
BLASCMGR        EQU        BLADMGR+LENDMGR      ; BLOAD ADDRESS OF SCMGR
BLAFMGR         EQU        BLASCMGR+LENSCMGR    ; BLOAD ADDRESS OF FMGR
BLACFM          EQU        BLAFMGR+LENFMGR      ; BLOAD ADDRESS OF CFMGR
BLABUFMG        EQU        BLACFM+LENCFM        ; BLOAD ADDRESS OF BUFMGR
BLAMEMMG        EQU        BLABUFMG+LENBUFMG    ; BLOAD ADDRESS OF MEMMGR
                REP        100

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSORG
*************************************************************************


```

```
==============================================================================
DOCUMENT :SOS1.3.5of5.FIVE:SOS.TCOMP.SOS.TEXT
==============================================================================

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: TCOMP.SOS
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT

:TABS 17,23,40
SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
SL4:DR1:ASM UMGR.SRC,UMGR.OBJ,6,1
SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1

*************************************************************************
* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: TCOMP.SOS
*************************************************************************
```

```
End of File -- Lines: 17761 Characters: 568225

SUMMARY:

  Total number of files : 1
  Total file lines      : 17761
  Total file characters : 568225
```



The End