Apple /// Computer
SOS 1.3 Source Code Listing

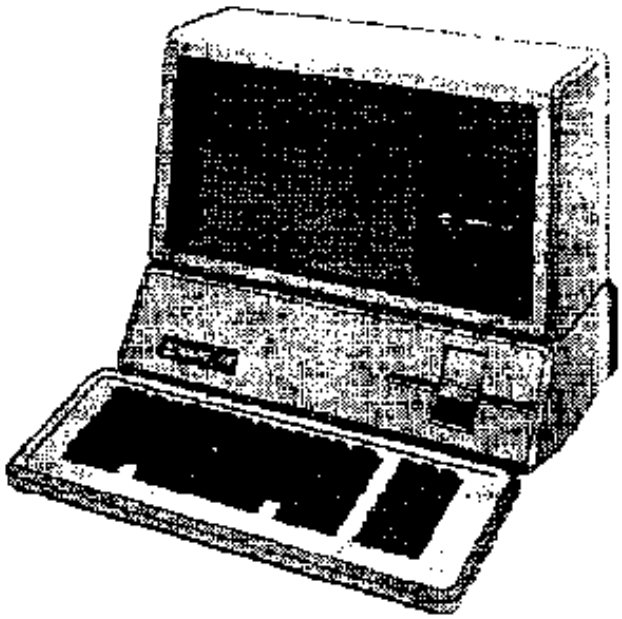
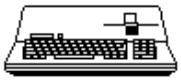


Table of Contents

Read Me
Source Code Catalog
Source File Line and Character Counts
Source Code File Listings

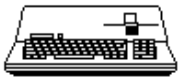


Read Me

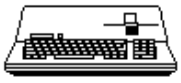
```
================================================================================

                    READ ME FILE FOR SOS SOURCE CODE DISK

                           Publicus  --  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 ...
```

Source Code Catalog

```
][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][[
           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
```

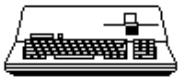


```
000063    COMPILE.SOS            2  31-Dec-89 20:01  Asciifile   97    3
000064    SOS.BLOAD              1  31-Dec-89 20:02  Asciifile  450    1
000065    SOS.LINK               1  31-Dec-89 20:03  Asciifile  170    1
000066    SOS.RENAME             2  31-Dec-89 20:03  Asciifile   11    3
000067    FEB01.1982             2  31-Dec-89 20:04  Asciifile   64    3
000068    PUBLICRELEASE          1  31-Dec-89 20:05  Asciifile   71    1
000069    COMP.SOS.NOLIST        2  31-Dec-89 20:05  Asciifile   79    3
000070    TCOMP.SOS              1  31-Dec-89 20:06  Asciifile  388    1
000071    SOSORG                 5  31-Dec-89 20:07  Asciifile  428    6
000072    C.S                    1  31-Dec-89 20:08  Asciifile  116    1
000073    C.BI2                  1  31-Dec-89 20:09  Asciifile   76    1
000074    C3                     1  31-Dec-89 20:09  Asciifile  155    1
000075    COMP.OPR.IPL           1  31-Dec-89 20:10  Asciifile  124    1
000076  16 files listed, 244 blocks available
```



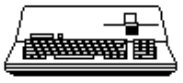
Source File Line and Character Counts

File Name	Lines	Chars
SOS.ALLOC.TEXT	397	17638
SOS.BFM.INIT2.SRC.TEXT	292	7309
SOS.BUFMGR.SRC.TEXT	789	19743
SOS.C.BI2.TEXT	12	535
SOS.C.S.TEXT	14	569
SOS.C3.TEXT	15	605
SOS.CFMGR.SRC.TEXT	496	13356
SOS.CLOSE.EOF.TEXT	410	17303
SOS.COMP.OPR.IPL.TEXT	15	594
SOS.COMP.SOS.NOLIST.TEXT	27	1055
SOS.COMPILE.BFM.TEXT	13	538
SOS.COMPILE.SOS.TEXT	28	1067
SOS.CREATE.TEXT	547	23241
SOS.DESTROY.TEXT	482	21039
SOS.DEVMGR.SRC.TEXT	298	8289
SOS.DISK3.DATA.SRC.TEXT	84	2847
SOS.DISK3.MAIN.SRC.TEXT	234	8117
SOS.DISK3.SIO.SRC.TEXT	255	8500
SOS.DISK3.SRC.TEXT	220	7736
SOS.DISK3.SUBS.SRC.TEXT	286	9331
SOS.DISK3.USEL.SRC.TEXT	168	5831
SOS.DISK3.WRT.SRC.TEXT	170	5611
SOS.EQUATES.TEXT	366	13973
SOS.FEB01.1982.TEXT	25	1029
SOS.FMGR.SRC.TEXT	119	3003
SOS.FNDFIL.TEXT	809	32436
SOS.INIT.SRC.TEXT	476	12190
SOS.IPL.SRC1.TEXT	664	18089
SOS.IPL.SRC2.TEXT	465	12829
SOS.LC.TEXT	13	562
SOS.LCHK.TEXT	16	659
SOS.MEMMGR.A.SRC.TEXT	486	13115
SOS.MEMMGR.B.SRC.TEXT	606	13264
SOS.MEMMGR.C.SRC.TEXT	500	10657
SOS.OPRMSG.SRC.TEXT	342	8421
SOS.PATH.TEXT	596	25582
SOS.POSN.OPEN.TEXT	718	32672
SOS.PRINT.TEXT	31	1063
SOS.PUBLICRELEASE.TEXT	13	544
SOS.READ.WRITE.TEXT	669	28620
SOS.SCMGR.SRC.TEXT	607	16237
SOS.SOS.BLOAD.TEXT	33	897
SOS.SOS.LINK.TEXT	26	633
SOS.SOS.RENAME.TEXT	27	988
SOS.SOSLDR.A.SRC.TEXT	123	4449
SOS.SOSLDR.B.SRC.TEXT	85	3059
SOS.SOSLDR.C.SRC.TEXT	252	7735
SOS.SOSLDR.D.SRC.TEXT	635	19725
SOS.SOSLDR.E.SRC.TEXT	442	12539
SOS.SOSLDR.F.SRC.TEXT	585	16170
SOS.SOSLDR.SRC.TEXT	131	4618
SOS.SOSORG.TEXT	68	3844
SOS.SWAPOUT.IN.TEXT	404	15484
SOS.SYSERR.SRC.TEXT	206	5470
SOS.SYSGLOB.SRC.TEXT	370	12894
SOS.TCOMP.SOS.TEXT	20	844
SOS.UMGR.SRC.TEXT	818	21588
SOS.VOLUME.TEXT	225	7273
Total	17223	564009

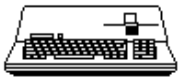


Source Code File Listings

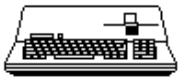
```
================================================================================
FILE: "SOS.ALLOC.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: ALLOC
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  *
000007  DEALLOC         STX       BMCNT               ; SAVE HIGH ORDER ADDRESS OF BLOCK TO BE FREED.
000008                  PHA                           ; SAVE IT
000009                  LDX       VCBPTR              ; WHILE THE BITMAP
000010                  LDA       VCB+VCBTBLK+1,X     ; DISK ADDRESS IS CHECKED
000011                  CMP       BMCNT               ; TO SEE IF IT MAKES SENSE
000012                  PLA                           ; RESTORE
000013                  BCC       DEALERR1            ; BRANCH IF IMPOSSIBLE
000014                  TAX
000015                  AND       #$7                 ; GET THE BIT TO BE OR-ED IN.
000016                  TAY
000017                  LDA       WHICHBIT,Y          ; (SHIFTING TAKES 7 BYTES, BUT IS SLOWER)
000018                  STA       NOFREE              ; SAVE BIT PATTERN
000019                  TXA                           ; GET LOW BLOCK ADDRESS AGAIN.
000020                  LSR       BMCNT
000021                  ROR       A                   ; GET POINTER TO BYTE IN BITMAP THAT REPRESENTS
000022                  LSR       BMCNT               ; THE BLOCK ADDRESS.
000023                  ROR       A
000024                  LSR       BMCNT
000025                  ROR       A
000026                  STA       BMPTR               ; SAVE POINTER.
000027                  LSR       BMCNT               ; NOW TRANSFER BIT WHICH SPECIFIES WHICH PAGE OF BITMAP.
000028                  ROL       HALF
000029                  LDX       BMTAB               ; (THIS POINTS TO THE TABLE FOR THE BITMAP BUFFER USED).
000030                  LDA       BMACMAP,X           ; WHAT IS THE CURRENT MAP
000031                  CMP       BMCNT               ; IS IN CORE BIT MAP THE ONE WE WANT?
000032                  BEQ       DEALL1              ; BRANCH IF IN-CORE IS CORRECT.
000033                  JSR       BMAPUP              ; PUT CURRENT MAP AWAY.
000034                  BCS       DEALERR             ; PASS BACK ANY ERROR.
000035                  LDA       BMCNT               ; GET DESIRED MAP NUMBER.
000036                  LDY       #VCBCMAP
000037                  STA       (VCBPTR),Y          ; AND MAKE IT CURRENT.
000038                  LDX       BMTAB
000039                  LDA       BMADEV,X
000040                  JSR       GTBMAP              ; READ IT INTO THE BUFFER,
000041                  BCS       DEALERR
000042  DEALL1          LDY       BMPTR               ; INDEX TO BYTE.
000043                  LSR       HALF
000044                  BCC       DEALL2              ; BRANCH IF ON PAGE ONE OF BITMAP.
000045                  INC       BMADR+1
000046  DEALL2          LDA       NOFREE              ; THE INDIVIDUAL BIT.
000047                  ORA       (BMADR),Y
000048                  STA       (BMADR),Y
000049                  BCC       DEALL3              ; BRANCH IF ADDRESS IS PROPER
000050                  DEC       BMADR+1
000051  DEALL3          LDX       BMTAB               ; MARK BITMAP AS MODIFIED.
000052                  LDA       #$80
000053                  ORA       BMASTAT,X
000054                  STA       BMASTAT,X
000055                  CLC
000056  DEALERR         RTS
000057  DEALERR1        LDA       #BITMAPADR          ; BIT MAP BLOCK NUMBER IMPOSSIBLE
000058                  SEC                           ; SAY BIT MAP DISK ADDRESS WRONG
000059                  RTS                           ;   (PROBABLY DATA MASQUERADING AS INDEX BLOCK)
000060  *
000061  WHICHBIT        DFB       $80,$40,$20,$10
000062                  DFB       8,4,2,1
000063  *
000064  *
000065                  PAGE
000066  *
000067  ALCIDXS         LDA       #0                  ; ALLOCATION OF THE INDEXES ALWAYS FILLS IN
000068                  STA       SAPTR               ; STARTING AT THE BEGINNING OF THE BLOCK.
000069                  JSR       ALC1BLK             ; THIS GETS FIRST INDEX AND SETS UP A
000070                  BCS       ERRALC1             ; POINTER TO THE FREE BLOCKS (TO AVOID
000071  ALIDX1          LDY       SAPTR               ; SCANNING THE WHOLE BLOCK EVERY TIME).
000072                  STA       (TINDX),Y           ; SAVE INDEX BLOCK ADDRESS (LOW)
000073                  INC       TINDX+1
000074                  LDA       SCRTCH+1            ; GET HIGH BYTE OF ADDRESS
```



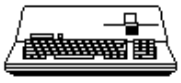
```
000075                  STA     (TINDX),Y       ; (AND SAVE IT)
000076                  DEC     TINDX+1
000077                  DEC     REQL            ; HAS REQUEST BEEN SATIFIED?
000078                  BEQ     ALDXEND         ; (CARRY IS CLEAR)
000079                  INC     SAPTR           ; BUMP INDEX POINTER
000080                  LDY     BMPTR           ; GET INDEX POINTER TO LAST ACCESSED BIT GROUP
000081                  LDA     HALF            ; WHICH HALF OF MAP? (BOTH BMPTR & HALF SET UP BY 'ALC1BLK')
000082                  BNE     SECNDHAF
000083                  JSR     GETBITS1        ; GET NEXT FREE BLOCK ADDRESS.
000084                  BCC     ALIDX1          ; BRANCH IF NO ERROR
000085  ERRALC1         RTS
000086  *
000087  SECNDHAF        JSR     GETBITS2        ; GET NEXT FREE BLOCK ADDRESS FROM SECOND HALF OF BIT MAP
000088                  BCC     ALIDX1          ; BRANCH IF NO ERROR.
000089  ALDXEND         RTS                     ; RETURN STATUS (CARRY SET INDICATES ERROR)
000090  *
000091  *
000092  ALC1BLK         JSR     FNDBMAP         ; GET ADDRESS OF BIT MAP IN 'BMADR'
000093                  BCS     ERRALC1         ; BRANCH IF ERROR ENCOUNTERED
000094  SRCHFRE         LDY     #0              ; START SEARCH AT BEGINNING OF BIT MAP BLOCK
000095                  STY     HALF            ; INDICATE WHICH HALF (PAGE) WE'RE SEARCHING.
000096  GETBITS1        LDA     (BMADR),Y
000097                  BNE     BITFOUND        ; FREE BLOCKS ARE INDICATED BY 'ON' BITS
000098                  INY
000099                  BNE     GETBITS1        ; CHECK ALL OF 'EM IN FIRST PAGE.
000100                  INC     BMADR+1         ; BUMP HIGH ADDRESS OF CURRENT BITMAP
000101                  INC     HALF            ; INDICATE SEARCH HAS PROGRESSED TO PAGE 2
000102                  INC     BASVAL          ; BASE VALUE= BASE ADDRESS/2048
000103  GETBITS2        LDA     (BMADR),Y       ; SEARCH SECOND HALF FOR FREE BLOCK
000104                  BNE     BITFOUND
000105                  INY
000106                  BNE     GETBITS2
000107                  DEC     BMADR+1         ; RESET BIT MAP ADDRESS TO BEGINNING.
000108                  INC     BASVAL          ; ADD 2048 OFFSET FOR NEXT PAGE
000109                  JSR     NXTBMAP         ; GET NEXT BITMAP (IF IT EXISTS) AND UPDATE VCB.
000110                  BCC     SRCHFRE         ; BRANCH IF NO ERROR ENCOUNTERED.
000111                  RTS                     ; RETURN ERROR.
000112                  PAGE
000113  *
000114  BITFOUND        STY     BMPTR           ; SAVE INDX POINTER TO VALID BIT GROUP
000115                  LDA     BASVAL          ; SET UP FOR BLOCK ADDRESS CALCULATION
000116                  STA     SCRTCH+1
000117                  TYA                     ; GET ADDRESS OF BIT PATTERN
000118                  ASL     A               ; MULTIPLY THIS AND BASVAL BY 8
000119                  ROL     SCRTCH+1
000120                  ASL     A
000121                  ROL     SCRTCH+1
000122                  ASL     A
000123                  ROL     SCRTCH+1
000124                  TAX                     ; NOW X= LOW ADDRESS WITHIN 7 OF ACTUAL ADDRESS.
000125                  LDA     (BMADR),Y       ; GET BIT PATTERN AGAIN
000126                  SEC                     ; MARK RIGHT END OF BYTE.
000127  ADCALC          ROL     A               ; FIND LEFT MOST 'ON' BIT
000128                  BCS     BOUNCE          ; BRANCH IF FOUND.
000129                  INX                     ; ADJUST LOW ADDRESS
000130                  BNE     ADCALC          ; BRANCH ALWAYS
000131  BOUNCE          LSR     A               ; RESTORE ALL BUT LEFT MOST BIT TO ORIGINAL POSITION
000132                  BCC     BOUNCE          ; LOOP UNTIL MARK (SET ABOVE) MOVES INTO CARRY
000133                  STA     (BMADR),Y       ; UPDATE BITMAP TO SHOW ALLOCATED BLOCK IN USE.
000134                  STX     SCRTCH          ; SAVE LOW ADDRESS.
000135                  LDX     BMTAB           ; UPDATE BIT MAP BUFFER STATUS
000136                  LDA     #$80            ; INDICATE MAP HAS BEEN MODIFIED
000137                  ORA     BMASTAT,X       ; (X IS EITHER 0 OR 6 FOR
000138                  STA     BMASTAT,X       ; BUFFER 'A' OR 'B' RESPECTIVELY.)
000139                  LDY     #VCBTFRE        ; SUBTRACT 1 FROM TOTAL FREE
000140                  LDA     (VCBPTR),Y      ; BLOCKS IN VCB TO ACCOUNT FOR NEWLY
000141                  SBC     #1              ; ALLOCATED BLOCK (CARRY IS SET FROM 'BOUNCE')
000142                  STA     (VCBPTR),Y
000143                  BCS     RET1BLK         ; BRANCH IF HI FREE COUNT DOESN'T NEED ADJUSTMENT.
000144                  INY
000145                  LDA     (VCBPTR),Y      ; ADJUST HIGH COUNT.
000146                  SBC     #0              ; (CARRY IS CLEAR, SO ACC=ACC-1)
000147                  STA     (VCBPTR),Y
000148  RET1BLK         CLC                     ; INDICATE NO ERROR ENCOUNTERED
000149                  LDA     SCRTCH          ; GET ADDRESS LOW IN ACC.
000150                  LDY     SCRTCH+1        ; AND HIGH ADDRESS IN Y
000151                  RTS                     ; RETURN ADDRESS OF NEWLY ALLOCATED BLOCK.
000152  *
000153                  PAGE
000154  *
000155  GTTINDX         LDY     #VCBDEV         ; GET DEVICE NUMBER SO WE DON'T
```



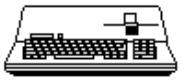
```
000156                  LDX     #0                  ; ANTICPATE USING BUFFER 'A'.
000157                  LDA     (VCBPTR),Y          ; USE THE BUFFER USED BY IT!
000158                  CMP     BMADEV              ; IS IT IN BUFFER 'A'?
000159                  BEQ     FREEBE              ; IF SO, FREE 'B'!
000160                  CMP     BMBDEV              ; IF NOT, IS IT IN 'B'?
000161                  BEQ     FREEA               ; IF SO, FREE UP BUFFER 'A'
000162                  JSR     FNDBMAP             ; OTHERWISE, FORCE ALLOCATION FOR ONE OF THE BUFFERS
000163                  BCC     GTTINDX             ; NOW TRY AGAIN.
000164                  RTS                         ; RETURN ERROR.
000165  *
000166  FREEBE          LDX     #BMTABSZ            ; DE-ALLOCATE BUFFER IF NECESSARY
000167  FREEA           STX     NOFREE              ; SAVE WHICH BUFFER WE'RE LOOKIN AT.
000168                  LDY     BMASTAT,X           ; DO WE NEED TO WRITE BUFFER TO FREE IT?
000169                  BPL     USEBUF              ; NO, THEN USE IT.
000170                  STX     ZPGTEMP             ; SAVE BM BUFFER ID FOR A BIT
000171                  JSR     WRTBMAP             ; WRITE BM TO OWNING UNIT
000172                  BCS     SOMERR1             ; RETURN ANY ERROR (W/O RELEASING BM)
000173                  LDX     ZPGTEMP             ; FETCH THE BM BUFFER ID
000174                  LDA     #0
000175                  STA     BMASTAT,X           ; AND MARK BM BUFFER AS FREE
000176  USEBUF          LDX     NOFREE              ; GET INDEX TO BUFFER INFO
000177                  LDA     #0                  ; MARK STATUS OF BUFFER AS FREE.
000178                  STA     BMADEV,X            ; (DEVICE 0 IS NOT ANY DEVICE)
000179                  STA     TINDX
000180                  STA     BMADR
000181                  LDA     BMAMADR,X           ; GET MEMORY ADDRESS OF FREE BUFFER.
000182                  STA     TINDX+1
000183                  TXA                         ; SET UP PROPER HI ADDRESS OF BIT MAP TOO...
000184                  EOR     #BMTABSZ            ; SELECT ALTERNATE BIT MAP TABLE.
000185                  STA     BMTAB               ; (TO INDICATE WHICH IS BITMAP)
000186                  TAX
000187                  LDA     BMAMADR,X           ; GET HIGH ADDRESS OF BIT MAP.
000188                  STA     BMADR+1
000189                  LDA     BMBUFBNK            ; AND BANK PAIR NUMBER.
000190                  STA     SSTIDXH
000191                  STA     SISBMADR
000192                  CLC                         ; INDICATE NO ERRORS
000193  SOMERR1         RTS
000194  *
000195                  PAGE
000196  NXTBMAP         LDY     #VCBTBLK+1          ; BEFORE BUMPING TO NEXT MAP,
000197                  LDA     (VCBPTR),Y          ; CHECK TO BE SURE THERE IS
000198                  LSR     A                   ; INDEED A NEXT MAP!
000199                  LSR     A
000200                  LSR     A
000201                  LSR     A
000202                  LDY     #VCBCMAP
000203                  CMP     (VCBPTR),Y          ; ARE THERE MORE MAPS?
000204                  BEQ     NOMORBIT            ; BRANCH IF NO MORE TO LOOK AT.
000205                  LDA     (VCBPTR),Y          ; ADD 1 TO CURRENT MAP
000206                  CLC
000207                  ADC     #1
000208                  STA     (VCBPTR),Y
000209                  LDY     #VCBDEV
000210                  LDA     (VCBPTR),Y
000211                  TAX                         ; GO WRITE OUT LAST MAP IF NECESSARY
000212                  JSR     UPBMAP
000213                  JMP     FNDBMAP             ; READ NEXT BIT MAP INTO BUFFER
000214  *
000215  GETA.BUF        LDX     #0
000216                  BEQ     FRESHMAP
000217  *
000218  GETB.BUF        LDX     #BMTABSZ
000219                  BNE     FRESHMAP            ; BRANCH ALWAYS
000220  *
000221  *
000222  FNDBMAP         LDY     #VCBDEV             ; GET DEVICE NUMBER
000223                  LDA     (VCBPTR),Y
000224                  LDX     #0                  ; START WITH MAP 'A'
000225  FNDMAP1         CMP     BMADEV,X
000226                  BNE     TRYMAP2
000227  FRESHMAP        STX     BMTAB               ; SAVE POINTER TO BIT MAP INFO TABLE
000228                  LDY     BMASTAT,X           ; IS THIS ONE ALREADY MODIFIED?
000229                  BMI     BMFOUND             ; YES, RETURN POINTER IN 'BMADR'
000230                  JSR     GTBMAP              ; OTHERWISE READ IN FRESH BIT MAP
000231                  BCC     BMFOUND             ; BRANCH IF SUCCESSFUL.
000232                  RTS                         ; OTHERWISE, RETURN ERROR.
000233  *
000234  TRYMAP2         DEX                         ; WAS LAST FAILURE MAP 'A'
000235                  BPL     FRBMBUF             ; NO, MUST FREE UP ONE OF THE BUFFERS
000236                  LDX     #BMTABSZ            ; TRY BIT MAP BUFFER 'B'.
```



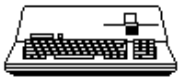
```
000237                  JMP         FNDMAP1
000238                  PAGE
000239  *
000240  BMFOUND         LDX         BMTAB               ; WHICH TABLE?
000241                  LDY         #VCBCMAP
000242                  LDA         (VCBPTR),Y
000243                  ASL         A
000244                  STA         BASVAL
000245                  LDA         BMAMADR,X           ; GET HIGH ADDRESS
000246                  STA         BMADR+1
000247                  LDA         BMBUFBNK            ; GET BANK NUMBER OF BUFFER BIT MAP BUFFERS
000248                  STA         SISBMADR
000249                  LDA         #0                  ; BUFFERS ALWAYS FALL ON A PAGE BOUNDARY
000250                  STA         BMADR
000251                  CLC                             ; INDICATE ALL IS VALID AND GOOD!
000252                  RTS
000253  *
000254  NOMORBIT        LDA         #OVRERR             ; INDICATE REQUEST CAN'T BE FILLED.
000255                  SEC                             ; INDICATE ERROR
000256                  RTS
000257  *
000258  FRBMBUF         SEC
000259                  LDX         BMTAB               ; FIND OUT WHICH WAS LAST USED.
000260                  BEQ         CHKBMB              ; IF 'A' WAS USED CHECK 'B' FIRST
000261                  CLC                             ; INDICATE 'A' IS CHECKED FIRST
000262                  BIT         BMASTAT             ; IS BUFFER 'A' FREE (UNMODIFIED)?
000263                  BPL         GETA.BUF            ; YES, USE IT.
000264  CHKBMB          BIT         BMBSTAT             ; IS BUFFER 'B' FREE?
000265                  BCC         FREBUF1             ; BRANCH IF BOTH ARE USED
000266                  BPL         GETB.BUF            ; YES...
000267                  BIT         BMASTAT             ; (CHECK 'A')
000268                  BPL         GETA.BUF
000269  FREBUF1         LDX         #0
000270                  BCC         FREBUFA             ; BRANCH IF BUFFER 'A' HAS LEAST PRIORITY.
000271                  LDX         #BMTABSZ
000272  FREBUFA         STX         ZPGTEMP             ; SAVE BM BUFF ID FOR A BIT
000273                  JSR         WRTBMAP             ; XREG PASSES BM BUFF ID
000274                  BCS         NOGO                ; ERROR ENCOUNTERED ON WRITING
000275                  LDX         ZPGTEMP             ; FETCH BM BUFF ID
000276                  LDA         #0
000277                  STA         BMASTAT,X           ; AND MARK BM BUFFER AS FREE
000278                  BCC         FNDBMAP             ; LOOK AGAIN FOR FRRE BIT MAP BUFFER SPACE
000279  NOGO            RTS                             ; RETURN ERROR ON WRITING BM
000280  *
000281  UPBMAP          CPX         BMADEV              ; UPDATE BIT MAP OF DEVICE X
000282                  BNE         UPBM1
000283                  CLC                             ; FREE BUFFER 'A' IF NEEDED.
000284                  BIT         BMASTAT
000285                  BMI         FREBUF1             ; (CARRY CLEAR FOR BUFFER 'A')
000286                  RTS
000287                  PAGE
000288  *
000289  UPBM1           CPX         BMBDEV
000290                  BNE         NOUPDAT             ; DON'T UPDATE IF NOT NECESSARY.
000291                  BIT         BMBSTAT
000292                  BMI         FREBUF1             ; (CARRY IS SET)
000293  NOUPDAT         CLC
000294                  RTS                             ; RETURN 'NO ERROR'
000295  *
000296  CLEARBMS        EQU         *                   ; MAKE SURE ALL BIT MAPS ASSOCIATED
000297  * WITH A DEVICE ARE MARKED INVALID
000298  * IF A NEW VOLUME IS LOGGED IN ON IT.
000299  * INPUT ARG: A REG = DEVNUM
000300  * X REG PRESERVED
000301                  LDY         #0
000302                  CMP         BMADEV
000303                  BNE         CLRBM1              ; BRANCH IF BIT MAP A NOT OWNED
000304                  BIT         BMASTAT
000305                  BMI         CLRBM2              ; BRANCH IF BITMAP A BUSY
000306                  STY         BMADEV              ; ELSE, CLEAR IT
000307  CLRBM2          RTS                             ; NEED ONLY CLEAR ONE
000308  CLRBM1          CMP         BMBDEV              ; BIT MAP B?
000309                  BNE         CLRBM2              ; BRANCH IF BIT MAP B NOT OWNED BY DEVNUM
000310                  BIT         BMBSTAT
000311                  BMI         CLRBM2              ; BRANCH IF BITMAP B BUSY
000312                  STY         BMBDEV              ; ELSE CLEAR IT
000313                  RTS                             ; AND RETURN TO CALLER (NO ERRORS)
000314  *
000315  GTBMAP          STA         BMADEV,X            ; SAVE ACC AS CURRENT DEVICE FOR BUFFER
000316                  LDA         BMAMADR,X           ; GET HIGH ORDER ADDRESS OF BUFFER
000317                  STA         BMADR+1             ; SELECTED BY X
```



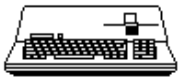
```
000318                  LDA         BMBUFBNK            ; AND GET BANK PAIR NUMBER
000319                  STA         SISBMADR            ; OF BOTH BIT MAP BUFFERS 'A' AND 'B'
000320                  LDY         #VCBCMAP            ; GET LOWEST MAP NUMBER WITH FREE BLOCKS IN IT.
000321                  LDA         (VCBPTR),Y
000322                  STA         BMACMAP,X           ; ASSOCIATE THE OFFSET WITH THE BITMAP CONTROL BLOCK
000323                  CLC
000324                  LDY         #VCBDMAP            ; ADD THIS NUMBER TO THE BASE
000325                  ADC         (VCBPTR),Y          ; ADDRESS OF FIRST BIT MAP
000326                  STA         BMADADR,X           ; SAVE LOW ADDRESS OF BIT MAP TO BE USED.
000327                  INY                             ; NOW GET HIGH DISK ADDRESS OF MAP
000328                  LDA         (VCBPTR),Y          ; ADD TO THIS THE STATE OF THE CARRY
000329                  ADC         #0
000330                  STA         BMADADR+1,X         ; SAVE HIGH DISK ADDRESS TOO.
000331  ; DROP INTO 'RDBMAP'
000332  *
000333                  PAGE
000334  *
000335                  LDA         #RDCMD              ; (X CONTAINS AN INDEX TO DETERMINE WHICH BUFFER)
000336  DOBMAP          STA         DHPCMD              ; SAVE DEVICE COMMAND
000337                  LDA         DEVNUM              ; FIX THE 'BIT MAP TRASH BUG'
000338                  PHA                             ; BY NOT MUNGING DEVNUM
000339                  LDA         BMADEV,X            ; GET DEVICE NUMBER.
000340                  STA         DEVNUM
000341                  LDA         BMADADR,X           ; AND MAP'S DISK ADDRESS
000342                  STA         BLOKNML
000343                  LDA         BMADADR+1,X
000344                  STA         BLOKNMH
000345                  LDA         BMAMADR,X           ; LASTLY GET THE ADDRESS OF THE BUFFER
000346                  LDX         BMBUFBNK            ; AND BANK NUMBER.
000347                  JSR         DOBITMAP            ; (NOTE: LOW ADDRESS IS FIXED TO ZERO AS THIS IS A BUFFER)
000348                  PLA                             ; RESTORE
000349                  STA         DEVNUM              ; THE DEVNUM WE CAME IN WITH!
000350                  RTS
000351  *
000352  WRTBMAP         LDA         #WRTCMD             ; WRITE BIT MAP POINTED TO BY X
000353                  JMP         DOBMAP
000354  *
000355  WRTGBUF         LDA         #WRTCMD             ; SET CALL FOR WRITE.
000356                  BNE         SVGCMD              ; BRANCH ALWAYS.
000357  RDGBUF          LDA         #RDCMD              ; SET CALL FOR READ.
000358  SVGCMD          STA         DHPCMD              ; PASSED TO DEVICE HANDLER.
000359                  LDA         BLOKNML             ; SAVE CURRENT
000360                  STA         TTLINK              ;   GBUF BLOCK
000361                  LDA         BLOKNMH             ; ADDRESS
000362                  STA         TTLINK+1            ; FOR DIRECTORY EXTEND
000363                  LDA         #GBUF/256           ; GET HIGH ADDRESS OF GENERAL BUFFER
000364                  LDX         #0                  ; TO FORCE ACCESS TO NON BANK MEMORY.
000365  DOBITMAP        EQU         *
000366  DOIDX           STA         DBUFPH
000367                  STX         SISBPH              ; SELECT BANK
000368                  LDA         #0                  ; GENERAL PURPOSE BUFFERS ALWAYS
000369                  STA         DBUFPL              ; START ON A PAGE BOUNDARY.
000370                  JMP         FILEIO2             ; END VIA DEVICE DISPATCHER.
000371  *
000372  TTLINK          DS          2                   ; GBUF CURRENT ADDRESS
000373  *
000374  WRTINDX         LDA         #WRTCMD
000375                  LDX         IDXADRL             ; GET BLOCK ADDRESS OF INDEX BLOCK
000376                  LDY         IDXADRH
000377  DOFRST          STA         DHPCMD              ; (ENTRY USED BY RD/WRTDFRST)
000378                  STX         BLOKNML
000379                  STY         BLOKNMH
000380                  LDA         TINDX+1             ; HIGH RAM ADDRESS OF INDEX BLOCK
000381                  LDX         SSTIDXH             ; AND BANK NUMBER.
000382                  JMP         DOIDX               ; AND GO DO REQUESTED OPERATION.
000383  *
000384  WRTDFRST        LDA         #WRTCMD             ; WRITE FILE'S FIRST BLOCK (USED
000385                  BNE         FADDR               ; BY CREATE, SO ADDRESS IN 'D.' STUFF).
000386  RDFRST          LDA         #RDCMD
000387  FADDR           LDX         DFIL+D.FRST         ; (BUFFER ADDRESS IS IN 'TINDX')
000388                  LDY         DFIL+D.FRST+1
000389                  JMP         DOFRST
000390  *
000391  *
000392                  CHN         POSN/OPEN,4,2
000393
000394  ****************************************************************************
000395  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: ALLOC
000396  ****************************************************************************
000397
```



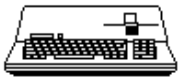
End of File -- Lines: 397 Characters: 17241



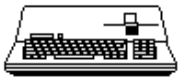
```
================================================================================
FILE: "SOS.BFM.INIT2.SRC.TEXT"
================================================================================

000001  ****************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: BFM.INIT2.SRC
000003  ****************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1  BFM.INIT2"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGBFMI
000010                  MSB        OFF
000011                  REP        60
000012  *          COPYRIGHT (C) APPLE COMPUTER INC.  1980
000013  *                      ALL RIGHTS RESERVED
000014                  REP        60
000015  *
000016  * BLOCK FILE MANAGER INIT2
000017  *
000018  *  SECONDARY INITIALIZATION ROUTINE FOR BLOCK FILE MANAGER
000019  *
000020  * MODIFIED: 03/25/81 TO UTILIZE NEW
000021  *   DISK DRIVER'S SEEKDSK3 ROUTINE.
000022  *  CHANGES MARKED BY 'D3RRA81084'
000023  *
000024  * MODIFIED: 08/19/81 TO WORK WITH NEW
000025  *   SOSLDR MODULE.
000026                  REP        60
000027  *
000028                  ENTRY      BFM.INIT2
000029  *
000030  *EXTRN I.BASE.P ; ENTRY IN SOSLDR
000031                  EXTRN      SYSBANK
000032                  EXTRN      SXPAGE
000033                  EXTRN      CZPAGE
000034                  EXTRN      SEEKDSK3            ;IN DISKDH/D3RRA81084
000035                  EXTRN      NMIDSBL             ;/D3RRA81084
000036  I.BASE.P        EQU        $2
000037                  PAGE
000038  *
000039  * CONSTANTS
000040  *
000041  KERNEL.BASE     EQU        $B800               ; BASE ADDRESS OF SOS KERNEL
000042  ROMID           EQU        $A0                 ;$F1B9 OF NEW ROM/D3RRA81084
000043  SLOT            EQU        $60
000044  BEGTRK          EQU        $9
000045  BEGSECT         EQU        $2
000046  ENDSECT         EQU        $6
000047  *
000048  * ZERO PAGE
000049  *
000050  TRACK           EQU        $99
000051  SECTOR          EQU        $98
000052  VOLUME          EQU        $9A
000053  KEY             EQU        $E0                 ; THRU $E7
000054  PREV.K          EQU        KEY+$8
000055  XIDX            EQU        KEY+$9
000056  I               EQU        KEY+$A              ; & $B
000057  *
000058  * ROM ROUTINES
000059  *
000060  RDADR           EQU        $F1B9               ;REV1
000061  RDADRX          EQU        $F1BD               ;REV0
000062  *
000063  * HARDWARE LOCATIONS
000064  *
000065  E.REG           EQU        $FFDF
000066  B.REG           EQU        $FFEF
000067  MOTORON         EQU        $C089
000068  MOTOROFF        EQU        $C088
000069                  PAGE
000070                  REP        60
000071  *
000072  * BFM.INIT2 ENTRY POINT
000073  *
000074                  REP        60
000075  *
000076  STATE           DFB        $FE                 ; FF=1ST ENTRY, 0=2ND ENTRY, 1=PROT
```



```
000077  *
000078  BFM.INIT2       EQU     *
000079                  INC     STATE
000080                  BMI     BFMI050
000081                  JSR     GETK
000082                  LDA     RETRY
000083                  BEQ     BADNEWS
000084                  BCC     BFMI050
000085                  JSR     NMIDSBL
000086                  JSR     DC
000087                  INC     STATE
000088  BFMI050         CLC
000089                  RTS
000090  BADNEWS         SEC                             ; I/O ERROR
000091                  RTS
000092                  PAGE
000093                  REP     60
000094  *
000095  * DECODE SUBROUTINE
000096  *
000097  * TO ENCODE:
000098  *    E0.E8:        - INIT KEY  & PREV.K
000099  *    B84E:4C 64 B8 - JUMPS AROUND INTERP'S 3 BYTE OVERWRITE
000100  *    1A02.1A03:    - NEW INTERP'S LOAD ADR (LO,HII)
000101  *    B81DG:        - JSR FROM MONITOR
000102  *
000103                  REP     60
000104  DC              EQU     *
000105                  LDA     B.REG                   ; SAVE BANK REGISTER
000106                  PHA
000107                  LDA     SYSBANK                 ;    AND SWITCH TO SYSTEM BANK
000108                  STA     B.REG
000109                  CLC                             ; FETCH LOADER'S INTERPRETER POINTER
000110                  LDA     CZPAGE+I.BASE.P
000111                  ADC     #3
000112                  STA     I
000113                  PHA
000114                  LDA     CZPAGE+I.BASE.P+1
000115                  ADC     #0
000116                  STA     I+1
000117                  PHA
000118                  LDA     #0
000119                  STA     SXPAGE+I+1
000120  *
000121                  LDY     I                       ; ALIGN I PTR TO PAGE BOUNDARY
000122                  LDA     #0
000123                  STA     I
000124                  STA     PREV.K
000125  *
000126                  JSR     DCLOOP                  ; DECODE
000127  *
000128                  PLA                             ; RETRIEVE LOADER'S INTERPRETER POINTER
000129                  STA     I+1
000130                  PLA
000131                  STA     I
000132  *
000133                  LDY     #1                      ; REPOSITION LOADER'S INTERPRETER POINTER (PUT ENCODE JMP HERE)
000134                  LDA     (I),Y
000135                  STA     CZPAGE+I.BASE.P
000136                  INY
000137                  LDA     (I),Y
000138                  STA     CZPAGE+I.BASE.P+1
000139  *
000140                  LDY     #2                      ; WALK ON INTERPRETER'S FIRST INSTRUCTION (3 BYTES)
000141                  LDA     #0
000142  DCA             STA     (I),Y
000143                  DEY
000144                  BPL     DCA
000145                  PLA                             ; RESTORE BANK REGISTER (ENCODE JMP JUMPS TO HERE)
000146                  STA     B.REG
000147                  RTS
000148                  PAGE
000149                  REP     60
000150  *
000151  * DECODE LOOP SUBROUTINE
000152  *
000153                  REP     60
000154  DCLOOP          EQU     *
000155                  LDX     #7                      ; SHIFT LEFT ONE BIT
000156                  CLC
000157                  LDA     KEY
```

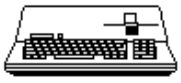


```
000158                  BPL         DC1
000159                  SEC
000160  DC1             ROL         KEY,X
000161                  DEX
000162                  BPL         DC1
000163  *
000164  DC2             TYA
000165                  AND         #7
000166                  EOR         #2
000167                  TAX
000168                  LDA         KEY,X
000169                  PHA
000170                  AND         #7
000171                  TAX
000172                  PLA
000173                  CLC
000174                  ADC         PREV.K
000175                  CLC
000176                  ADC         KEY,X
000177                  STA         PREV.K
000178                  EOR         (I),Y               ; DECODE BYTE
000179                  STA         (I),Y               ; AND PUT IT BACK
000180                  INY
000181                  BNE         DC2
000182                  INC         I+1
000183                  LDA         I+1
000184                  CMP         #<KERNEL.BASE
000185                  BCC         DCLOOP
000186                  RTS
000187                  PAGE
000188                  REP         60
000189  *
000190  * GETKEY SUBROUTINE
000191  *
000192                  REP         60
000193  *
000194  RETRY           DFB         10+1                ;TEN RETRIES
000195  OURTRACK        DS          1                   ;CURRENT TRACK/D3RRA81084
000196  *
000197  GETK            EQU         *
000198                  LDX         #7
000199                  STX         XIDX
000200                  LDX         #SLOT
000201                  LDA         MOTORON,X           ;ENSURE MOTOR STAYS ON
000202                  LDA         E.REG               ; SELECT 1MHZ, ROM
000203                  ORA         #$83
000204                  STA         E.REG
000205  *
000206  * NOTE: THE SEEKDSK3 ROUTINE HAS THESE /D3RRA81084
000207  *  CAVEATS: 1MHZ MODE, MOTOR IS ON, /D3RRA81084
000208  *  DRIVE CURRENTLY SELECTED, ROM+I/O ENABLED! /D3RRA81084
000209  *
000210  GETK010         LDA         #BEGTRK
000211                  STA         OURTRACK            ;WHERE WE SEEK TO /D3RRA81084
000212                  JSR         SEEKDSK3            ;HAVE DISKDH SEEK FOR US /D3RRA81084
000213  GETK020         LDX         #SLOT
000214                  JSR         DOREAD              ;FIND A SECTOR HEADER
000215                  BCS         IOERROR             ;=>RETRY IF BAD
000216                  LDA         SECTOR              ;WHERE ARE WE?
000217                  CMP         #BEGSECT            ;AT THE RIGHT PLACE?
000218                  BNE         GETK020             ;=>NO, GET THERE
000219  *
000220  GETK100         LDX         #1
000221                  JSR         WAIT                ; (X * 1284) + 15 MILISECONDS
000222                  LDX         XIDX
000223                  LDA         VOLUME
000224                  STA         KEY,X
000225                  DEC         XIDX
000226                  BMI         ENUFF
000227                  INC         OURTRACK            ;BUMP FOR NEXT TRACK /D3RRA81084
000228                  LDA         OURTRACK            ;WHERE TO GO /D3RRA81084
000229                  LDX         #SLOT
000230                  JSR         SEEKDSK3            ;DISKDH, PLEASE SEEK ME /D3RRA81084
000231                  LDX         #SLOT
000232                  JSR         DOREAD
000233                  BCC         GETK100
000234                  BCS         IOERROR
000235  *
000236  ENUFF           LDX         #SLOT
000237                  LDA         MOTOROFF,X
000238                  LDA         E.REG               ; SELECT 2MHZ, RAM
```

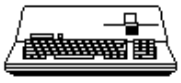


```
000239                  AND         #$7C
000240                  STA         E.REG
000241                  PAGE
000242                  LDA         SECTOR
000243                  CMP         #ENDSECT            ;TRACKS SYNC'ED?
000244                  BNE         NOTPROT
000245                  LDA         KEY
000246                  EOR         KEY+1
000247                  BEQ         NOTPROT             ;IF FIRST 2 VOLS ARE EQUAL
000248                  SEC
000249                  RTS
000250  *
000251  NOTPROT         LDA         #0
000252                  CLC
000253                  RTS
000254  *
000255  *
000256  DOREAD          JSR         WHICHROM
000257                  BCS         OLDREAD
000258                  JMP         RDADR
000259  OLDREAD         JMP         RDADRX
000260  *
000261  *
000262  WHICHROM        LDA         RDADR
000263                  CMP         #ROMID
000264                  CLC
000265                  BEQ         NEWROM
000266                  SEC
000267  NEWROM          RTS
000268  *
000269  *
000270  IOERROR         DEC         RETRY
000271                  BEQ         ERR1
000272                  JMP         GETK                ; TRY, TRY AGAIN
000273  ERR1            JMP         ENUFF               ; I/O ERROR, CLEANUP AND EXIT
000274  *
000275  *
000276  WAIT            LDY         #0
000277  W1              DEY
000278                  BNE         W1
000279                  DEX
000280                  BNE         W1
000281                  RTS
000282
000283  ZZLEN           EQU         $400
000284                  IFNE        ZZLEN-LENBFMI
000285                  FAIL        2,"SOSORG           FILE IS INCORRECT FOR BFM.INIT2"
000286                  FIN
000287
000288  ************************************************************************
000289  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: BFM.INIT2.SRC
000290  ************************************************************************
000291
000292
```

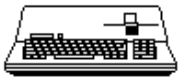
End of File -- Lines: 292 Characters: 7017



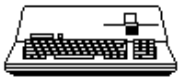
```
================================================================================
FILE: "SOS.BUFMGR.SRC.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: BUFMGR.SRC
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1  BUFFER MANAGER"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009  *ORGBUFMG EQU $F552
000010  *LENBUFMG EQU $31C
000011                  ORG        ORGBUFMG
000012  ZZORG           EQU        *
000013                  MSB        OFF
000014                  REP        60
000015  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000016  *                      ALL RIGHTS RESERVED
000017                  REP        60
000018  *
000019  * BUFFER MANAGER (VERSION = 1.1O   )
000020  *                (DATE    = 8/04/81)
000021  *
000022  * THIS MODULE IS RESPONSIBLE FOR CREATING AND RELEASING BUFFERS
000023  * FOR BOTH THE BLOCK FILE MANAGER AND, LATER, DEVICE HANDLERS
000024  * THE BUFFER MANAGER CREATES BUFFERS BY REQUESTING MEMORY
000025  * SEGMENTS FROM THE MEMORY MANAGER, AND RELEASES THEM VIA SAME.
000026  * THE PRIMARY DATA STRUCTURE IN THIS MODULE IS THE BUFFER TABLE.
000027  *
000028                  REP        60
000029  *
000030                  ENTRY      REQBUF
000031                  ENTRY      REQFXBUF
000032                  ENTRY      GETBUFADR
000033                  ENTRY      CHKBUF
000034                  ENTRY      RELBUF
000035  *
000036                  EXTRN      MMGR
000037                  EXTRN      SXPAGE
000038                  EXTRN      CZPAGE
000039                  EXTRN      CXPAGE
000040  *
000041                  EXTRN      SYSERR
000042                  EXTRN      SERR
000043                  EXTRN      OUTOFMEM
000044                  EXTRN      BUFTBLFULL
000045                  EXTRN      BADSYSBUF
000046  *
000047                  EXTRN      SYSDEATH
000048                  EXTRN      BADBUFNUM
000049                  EXTRN      BADBUFSIZ
000050  *
000051                  ENTRY      BUF.CNT
000052                  ENTRY      PGCT.T
000053                  ENTRY      XBYTE.T
000054                  ENTRY      BUFREF
000055                  PAGE
000056                  REP        60
000057  *
000058  * DATA DECLARATIONS
000059  *
000060                  REP        60
000061  *
000062  Z.REG           EQU        $FFD0
000063  *
000064  * MEMORY MGMT CALL PARM LOCATIONS ON SOS ZPAGE
000065  *
000066  M.TPARMX        EQU        $60                 ; FIRST ADR OF MEM SYS CALL PARMS ON SOS ZPAGE
000067  REQCODE         EQU        M.TPARMX+$0
000068  *
000069  FINDSEG         EQU        $1
000070  SRCHMODE        EQU        M.TPARMX+$1
000071  F.ID            EQU        M.TPARMX+$2
000072  F.PGCT          EQU        M.TPARMX+$3
000073  F.PGCTX         DS         2                   ; TEMP LOC FOR F.PGCT PARM
000074  F.BASE          EQU        M.TPARMX+$5
000075  F.BASEX         DS         2                   ; TEMP LOC FOR F.BASE PARM
000076  F.LIM           EQU        M.TPARMX+$7
```



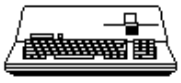
```
000077  F.LIMX          DS          2                   ; TEMP LOC FOR F.LIM PARM
000078  F.NUM           EQU         M.TPARMX+$9
000079  F.NUMX          DS          1                   ; TEMP LOC FOR F.NUM PARM
000080  *
000081  RELSEG          EQU         $5
000082  RLS.NUM         EQU         M.TPARMX+$1
000083  *
000084  * REQBUF DATA DECLARATIONS
000085  *
000086  RQB.PGCT        DS          1                   ; REQUESTED PAGE COUNT
000087  RQB.BNUM        DS          1                   ; BUFFER NUMBER (FM GETFREE CALL)
000088  *
000089  * REQFXBUF DATA DECLARATIONS
000090  *
000091  RQFB.PGCT       DS          1                   ; REQUESTED PAGE COUNT
000092  RQFB.BNUM       DS          1                   ; BUFFER NUMBER (FM GETFREE CALL)
000093  MAXPGCT         EQU         64                  ; MAX BUFSIZE=16K
000094  F.TPARMX        EQU         $A0                 ; FIRST ADR OF FILE SYS CALL PARMS ON SOS ZPAGE
000095  OPEN.LIST       EQU         F.TPARMX+$5         ; LOC OF OPEN.LIST PARM (OPEN SYS CALL)
000096  *
000097  * BUFCOMPACT DATA DECLARATIONS (SOURCE ALSO USED BY CHKBUF)
000098  *
000099  BUFC.BNUM       DS          1                   ; BUF# OF LOWEST BUFFER IN BUF.TBL
000100  SOURCE          EQU         M.TPARMX+$10        ; & $11
000101  DEST            EQU         M.TPARMX+$12        ; & $13
000102                  PAGE
000103                  REP         60
000104  *
000105  * BUFFER TABLE
000106  *
000107  * THE BUFFER TABLE CONSISTS OF "CNT"-1 ENTRIES (1 TO "CNT"-1).
000108  * EACH ENTRY IS "SIZ" BYTES IN LENGTH.  THE "PGCT" FIELD
000109  * CONTAINS 3 SUBFIELDS.  BIT 7 IS THE "FREE" FLAG (0=ACTIVE,1=FREE)
000110  * BIT 6 IS THE "FIXED" FLAG (0=FLOATING BUFFER,1=FIXED BUFFER)
000111  * BITS 5 THRU 0 CONTAIN THE PAGE COUNT OF AN "ACTIVE" ENTRY
000112  * (0=>1 PAGE,63=>64 PAGES DECIMAL).  THE "XBYTE" FIELD CONTAINS
000113  * THE PROPER XBYTE OF AN "ACTIVE" ENTRY.  THE "ADRH" FIELD
000114  * CONTAINS THE HIGH BYTE OF THE BUFFER ADDRESS.  IF THE
000115  * BUFFER ENTRY IS "FLOATING", THEN THE "SEG" FIELD CONTAINS THE
000116  * SEGMENT NUMBER AND THE LOW BYTE OF THE BUFFER ADDRESS IS
000117  * ASSUMMED TO BE ZERO.
000118  *
000119  * THUS, THE FOLLOWING RESTRICTIONS APPLY TO BUFFERS:
000120  *
000121  * (1) MAXIMUM BUFFER LENGTH IS 64 PAGES (16K)
000122  * (2) "FLOATING" BUFFERS ALWAYS BEGIN ON A PAGE BOUNDARY
000123  *     "FIXED" BUFFERS DO NOT.
000124  * (3) BUFFERS ARE ALWAYS AN INTEGRAL NUMBER OF PAGES IN LENGTH
000125  * (4) BUFFERS ALWAYS RESIDE IN THE 32K BANK MEMORY REGION,
000126  *     A LIMITATION OF FIND.SEG (MEMORY MANAGER)
000127  * (5) MAXIMUM NUMBER OF BUFFERS = 16; ENTRY 0 IS NOT USED.
000128  *
000129                  REP         60
000130  *
000131  * BUFFER TABLE
000132  *
000133  BUF.SIZ         EQU         5
000134  BUF.CNT         EQU         17
000135  BUF.TBL         DS          BUF.SIZ*BUF.CNT
000136  PGCT.T          EQU         BUF.TBL
000137  XBYTE.T         EQU         PGCT.T+BUF.CNT
000138  ADRH.T          EQU         XBYTE.T+BUF.CNT
000139  SEG.T           EQU         ADRH.T+BUF.CNT
000140  ADRL.T          EQU         SEG.T
000141  CHK.T           EQU         ADRL.T+BUF.CNT
000142  ISFIXED         EQU         $40
000143  ISFREE          EQU         $80
000144  *
000145  * BUFFER REFERENCE TABLE
000146  *
000147  * FIRST BYTE IS COUNT, FOLLOWED BY "COUNT" BUFFER #S.
000148  * THIS TABLE IS A LIST OF ALL BUFFERS REFERENCED DURING ONE
000149  * SOS SYSTEM CALL.  BUFFER #S ARE ADDED TO THIS LIST BY
000150  * GETBUFADR AND REMOVED BY CHKSUM.
000151  *
000152  BUFREF.CNT      EQU         17
000153  BUFREF          DS          BUFREF.CNT
000154  ZPAGEX          DS          1
000155                  PAGE
000156                  REP         60
000157  *
```



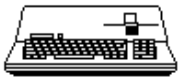
```
000158  * REQBUF
000159  *
000160  * INPUT:  PAGE.CNT (A)
000161  * OUTPUT: BUFNUM   (A)
000162  * ERROR:  "BUFFER TABLE FULL" - SYSERR
000163  *         "OUT OF MEMORY"     - SYSERR
000164  *         "BAD BUFFER SIZE"   - SYSDEATH
000165  *
000166  * THIS ROUTINE FINDS A FREE ENTRY IN THE BUFFER TABLE
000167  * AND THEN CALLS FIND.SEG (MMGR) TO OBTAIN MEMORY FOR IT.
000168  * IF MEMORY IS FOUND THEN THE BUFFER ENTRY IS MARKED "ACTIVE"
000169  * AND THE BUFFER INFO IS INSERTED INTO THE ENTRY
000170  *
000171                  REP        60
000172  *
000173  REQBUF          EQU        *
000174  *
000175  * IF REQUESTED PGCT OUT OF BOUNDS THEN FATAL ERR
000176  *
000177                  TAY
000178                  BEQ        RQB.ERR2            ; FATAL ERR, INVALID BUFFER SIZE
000179                  CPY        #MAXPGCT+1
000180                  BCS        RQB.ERR2            ; FATAL ERR, INVALID BUFFER SIZE
000181                  STY        RQB.PGCT            ; SAVE PAGE COUNT
000182  *
000183  * FIND FREE ENTRY IN BUF.TBL
000184  *
000185                  JSR        GETFREE
000186                  BCS        RQB.ERR             ; ERR, BUFFER TABLE FULL
000187                  STX        RQB.BNUM
000188  *
000189  * FIND PGCT*256 BYTES OF FREE MEMORY
000190  *
000191                  LDA        RQB.PGCT
000192                  JSR        FSEG
000193                  BCS        RQB.ERR1            ; ERR, OUT OF MEMORY
000194  *
000195  * INSERT PGCT, XBYTE, ADRH, SEG#, CHK BYTE IN BUF.TBL(BUF#)
000196  *
000197                  LDX        RQB.BNUM
000198                  DEC        RQB.PGCT            ; PAGE COUNT FIELD
000199                  LDA        RQB.PGCT
000200                  STA        PGCT.T,X
000201  *
000202                  LDX        F.BASEX             ; XBYTE & ADRH FIELDS
000203                  LDY        F.BASEX+1
000204                  JSR        CNVRT.ADR
000205                  CPX        #$8F
000206                  BNE        RQB010
000207                  LDX        #$7F                ; IF XBYTE=$8F THEN XBYTE:=$7F
000208  RQB010          TXA
000209                  LDX        RQB.BNUM
000210                  STA        XBYTE.T,X
000211                  TYA
000212                  STA        ADRH.T,X
000213  *
000214                  LDA        F.NUMX              ; SEG# FIELD
000215                  STA        SEG.T,X
000216  *
000217                  LDA        #0                  ; INIT CHECK BYTE TO NULL
000218                  STA        CHK.T,X
000219  *
000220                  TXA                            ; RETURN BUF#
000221                  CLC
000222                  RTS                            ; NORMAL EXIT
000223  *
000224  *
000225  RQB.ERR         LDA        #BUFTBLFULL
000226                  JSR        SYSERR
000227  *
000228  RQB.ERR1        LDA        #OUTOFMEM
000229                  JSR        SYSERR
000230  *
000231  RQB.ERR2        LDA        #BADBUFSIZ
000232                  JSR        SYSDEATH
000233                  PAGE
000234                  REP        60
000235  *
000236  * REQFXBUF
000237  *
000238  * INPUT:  PAGE.CNT (A)
```



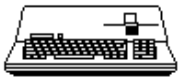
```
000239  * OUTPUT: BUFNUM   (A)
000240  * ERROR:  "BUFFER TABLE FULL"            - SYSERR
000241  *         "BAD SYSTEM.BUF PARM ADDRESS"  - SYSERR
000242  *         "BAD BUFFER SIZE"              - SYSDEATH
000243  *
000244  * THIS ROUTINE COMPUTES THE ACTUAL BUFFER ADDRESS IN THE OPEN
000245  * CALL (PARM "OPEN.LIST"), AND ALLOCATES A BUFFER ENTRY FOR IT.
000246  * NOTE:  THE SYSBUF PARAMETER MUST BE AN EXTENDED INDIRECT PTR!!
000247  *
000248                  REP        60
000249  *
000250  REQFXBUF        EQU        *
000251  *
000252  * IF REQUESTED PGCT OUT OF BOUNDS THEN FATAL ERR
000253  *
000254                  TAY
000255                  BEQ        RQFB.ERR2           ; FATAL ERR, BAD BUFFER SIZE
000256                  CPY        #MAXPGCT+1
000257                  BCS        RQFB.ERR2           ; FATAL ERR, BAD BUFFER SIZE
000258  *
000259                  STY        RQFB.PGCT           ; SAVE PAGE COUNT
000260  *
000261  * GET A FREE BUFFER ENTRY
000262  *
000263                  JSR        GETFREE
000264                  BCS        RQFB.ERR            ; ERR, BUFFER TABLE FULL
000265                  STX        RQFB.BNUM           ; SAVE BUF#
000266  *
000267  * FETCH SYSTEM.BUF PARAMETER IN OPEN SYSTEM CALL
000268  *
000269                  LDY        #3
000270                  LDA        (OPEN.LIST),Y
000271                  BNE        RQFB.ERR1           ; ERR, SYSBUF ADR
000272                  DEY
000273                  LDA        (OPEN.LIST),Y
000274                  TAY
000275                  LDA        CXPAGE+1,Y
000276                  BPL        RQFB.ERR1           ; ERR, SYSBUF ADR
000277                  CMP        #$8F
000278                  BCS        RQFB.ERR1           ; ERR, SYSBUF ADR
000279  *
000280  * INSERT XBYTE, ADRH, ADRL, PGCT, CHK BYTE INTO BUF.TBL(BUF#)
000281  *
000282                  LDX        RQFB.BNUM
000283                  STA        XBYTE.T,X
000284  *
000285                  LDA        CZPAGE+1,Y
000286                  BEQ        RQFB.ERR1           ; ERR SYSBUF ADR
000287                  CMP        #$81                ; CHECK FOR ADDRESS COMPENSATION
000288                  BCC        RQFB010
000289                  INC        XBYTE.T,X
000290                  AND        #$7F
000291  RQFB010         STA        ADRH.T,X
000292  *
000293                  LDA        CZPAGE,Y
000294                  STA        ADRL.T,X
000295  *
000296                  DEC        RQFB.PGCT
000297                  LDA        RQFB.PGCT
000298                  ORA        #ISFIXED
000299                  STA        PGCT.T,X            ; BUFFER ENTRY NOW "ACTIVE"
000300  *
000301                  LDA        #0                  ; INIT CHECK BYTE TO NULL
000302                  STA        CHK.T,X
000303  *
000304                  TXA                            ; RETURN BUF#
000305                  CLC
000306                  RTS                            ; NORMAL EXIT
000307  *
000308  RQFB.ERR        LDA        #BUFTBLFULL
000309                  JSR        SYSERR
000310  *
000311  RQFB.ERR1       LDA        #BADSYSBUF
000312                  JSR        SYSERR
000313  *
000314  RQFB.ERR2       LDA        #BADBUFSIZ
000315                  JSR        SYSDEATH
000316                  PAGE
000317                  REP        60
000318  *
000319  * GETBUFADR
```



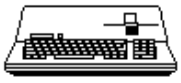
```
000320  *
000321  * INPUT:  BUFNUM   (A)
000322  *         ZPAGELOC (X)
000323  * OUTPUT: BUF ADR AT: X,X+1 & SXPAGE+1,X
000324  *         PAGE.CNT (A)
000325  *         BUFNUM   (Y)
000326  *
000327  * ERROR:  "BADBUFNUM" SYSDEATH
000328  *
000329                  REP        60
000330  *
000331  GETBUFADR       EQU        *
000332  *
000333  * IF BUF# OUT OF RANGE OR BUF.TBL(BUF#)=FREE
000334  * THEN FATAL ERR
000335  *
000336                  TAY
000337                  BEQ        GTBF.ERR             ; BUF#=0, FATAL ERR
000338                  CPY        #BUF.CNT
000339                  BCS        GTBF.ERR             ; BUF# > MAX BUF TABLE ENTRY, FATAL ERR
000340                  LDA        PGCT.T,Y
000341                  BMI        GTBF.ERR             ; BUF ENTRY MARKED "FREE", FATAL ERR
000342  *
000343  * OTHERWISE, CONSTRUCT BUFFER PTR ON SOS ZPAGE
000344  *
000345                  JSR        GETBUFADR1
000346  *
000347  * IF BUFFER NOT PREVIOUSLY REFERENCED ON THIS SOS CALL AND CHECK BYTE <> 0
000348  *    THEN COMPARE FIRST BYTE OF BUFFER WITH CHECK BYTE IN BUFFER TABLE.
000349  *         IF NO MATCH THEN KILL SYSTEM.
000350  *
000351                  STX        ZPAGEX
000352                  TYA
000353                  LDX        BUFREF
000354                  BEQ        GTBF020              ; BUFREF EMPTY
000355  *
000356  GTBF010         CMP        BUFREF,X             ; SEARCH FOR PREVIOUS REFERENCE
000357                  BEQ        GTBF030              ; MATCH FOUND
000358                  DEX
000359                  BNE        GTBF010
000360  *
000361  GTBF020         INC        BUFREF               ; LOG BUF # IN BUFREF TABLE
000362                  LDX        BUFREF
000363                  CPX        #BUFREF.CNT
000364                  BCS        GTBF.ERR             ; BUFREF TABLE OVFLOW, KILL SYSTEM
000365                  STA        BUFREF,X
000366  *
000367                  LDA        CHK.T,Y
000368                  BEQ        GTBF030              ; NO CHECK BYTE, SKIP CHECK
000369                  LDX        ZPAGEX
000370                  LDA        ($0,X)               ; COMPARE FIRST BYTE OF BUFFER
000371                  CMP        CHK.T,Y              ; WITH CHECK BYTE IN BUF TABLE
000372                  BNE        GTBF.ERR             ; NO MATCH, PULL THE PLUG
000373  *
000374  * RETURN PAGE.CNT TO CALLER
000375  *
000376  GTBF030         LDA        PGCT.T,Y
000377                  AND        #$3F                 ; STRIP OFF FREE,FIXED FLAGS
000378                  CLC
000379                  ADC        #1
000380  *
000381                  CLC
000382                  RTS
000383  *
000384  *
000385  GTBF.ERR        LDA        #BADBUFNUM
000386                  JSR        SYSDEATH
000387  *
000388  *
000389                  REP        60
000390  *
000391  * GETBUFADR1
000392  *
000393  * INPUT: PGCT.T(BUF#)  (A)
000394  *        ZPAGELOC      (X)
000395  *        BUF#          (Y)
000396  * ERROR: NONE.
000397  *
000398  * EXTRACTS THE BUFFER POINTER FROM THE BUFFER TABLE AND
000399  * PLACES IT ON ZERO PAGE AT X, X+1 & SXPAGE+1,X
000400  *
```



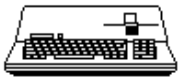
```
000401                  REP     60
000402  *
000403  GETBUFADR1      EQU     *
000404                  AND     #$40
000405                  BNE     GTB1010
000406                  LDA     #0                      ; "FIXED" BUFFER
000407                  BEQ     GTB1020                 ; ALWAYS TAKEN
000408  GTB1010         LDA     ADRL.T,Y                ; "FLOATING" BUFFER
000409  GTB1020         STA     0,X
000410                  LDA     ADRH.T,Y
000411                  STA     1,X
000412                  LDA     XBYTE.T,Y
000413                  ORA     #$80                    ; ENSURE $7F->$8F
000414                  STA     SXPAGE+1,X
000415                  RTS
000416                  PAGE
000417                  REP     60
000418  *
000419  * CHKBUF
000420  *
000421  * CHECK BUFFER.  FETCHES THE FIRST BYTE OF EACH BUFFER
000422  * REFERENCED DURING THE CURRENT SYSTEM CALL AND PLACES IT
000423  * IN CHK.T(BUF#).
000424  *
000425  * INPUT:  BUFREF TABLE
000426  *         BUFFER TABLE
000427  * OUTPUT: EMPTY BUFREF TABLE
000428  *         BUFFER TABLE'S CHECK BYTES UPDATED
000429  *         Z REG:=$18
000430  * ERROR:  NONE.
000431  *
000432                  REP     60
000433  *
000434  CHKBUF          EQU     *
000435                  LDY     BUFREF                  ; PICK UP COUNT
000436                  BEQ     CHKB.EXIT               ; EXIT IF BUFREF EMPTY
000437  *
000438                  LDA     #$18                    ; ENSURE SOS ZPAGE SWITCHED IN
000439                  STA     Z.REG
000440  *
000441  * UPDATE THE CHECK BYTE OF EACH BUF# IN THE BUFREF TABLE
000442  *
000443  CHKB010         LDX     #>SOURCE
000444                  LDA     BUFREF,Y
000445                  TAY
000446                  LDA     PGCT.T,Y
000447                  JSR     GETBUFADR1              ; PUT BUF#S ADR ON ZPAGE
000448                  LDA     ($0,X)
000449                  STA     CHK.T,Y
000450                  DEC     BUFREF
000451                  LDY     BUFREF
000452                  BNE     CHKB010                 ; IF COUNT<>0 THEN PROCESS NEXT BUF# IN BUFREF TABLE
000453  *
000454  CHKB.EXIT       RTS                             ; BUFREF TABLE IS EMPTY (COUNT=0)
000455                  PAGE
000456                  REP     60
000457  *
000458  * RELBUF
000459  *
000460  * INPUT:  BUFNUM   (A)
000461  * OUTPUT: NONE.
000462  * ERROR:  "BADBUFNUM" SYSDEATH
000463  *
000464  * THIS ROUTINE RELEASES THE BUFFER ENTRY, CALLS FIND.SEG TO
000465  * RELEASE THE CORRESPONDING MEMORY SEGMENT, AND CALLS
000466  * BUFCOMPACT TO PERFORM BUFFER COMPACTION.
000467  *
000468                  REP     60
000469  *
000470  RELBUF          EQU     *
000471  *
000472  * IF BUF# OUT OF RANGE OR BUF.TBL(BUF#)=FREE
000473  * THEN FATAL ERR
000474  *
000475                  TAY
000476                  BEQ     RLBF.ERR
000477                  CPY     #BUF.CNT
000478                  BCS     RLBF.ERR
000479                  LDA     PGCT.T,Y
000480                  BMI     RLBF.ERR
000481  *
```



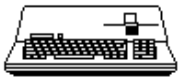
```
000482  * MARK BUF.TBL(BUF#)=FREE
000483  *
000484                  ORA         #ISFREE
000485                  STA         PGCT.T,Y
000486  *
000487  * IF BUF.TBL(BUF#)=FIXED THEN EXIT
000488  *
000489                  AND         #ISFIXED
000490                  BNE         RLBF.EXIT
000491  *
000492  * OTHERWISE CALL MEMORY MGR TO RELEASE BUFFER'S MEMORY SEG
000493  *
000494                  LDA         #RELSEG
000495                  STA         REQCODE
000496  *
000497                  LDA         SEG.T,Y
000498                  STA         RLS.NUM
000499  *
000500                  JSR         MMGR
000501                  BCS         RLBF.ERR            ; ANY ERR IS FATAL
000502  *
000503  * AND COMPACT BUFFERS
000504  *
000505                  JSR         BUFCOMPACT
000506  *
000507  RLBF.EXIT       CLC
000508                  RTS
000509  *
000510  RLBF.ERR        LDA         #BADBUFNUM
000511                  JSR         SYSDEATH
000512                  PAGE
000513                  REP         60
000514  *
000515  * BUFCOMPACT
000516  *
000517  * THIS ROUTINE IS RESPONSIBLE FOR PACKING ALL SOS BUFFERS UP
000518  * AGAINST THE HIGHEST AVAILABLE FREE MEMORY.  COULD IMPROVE THE
000519  * EFFICIENCY OF THIS COMPACTION CYCLE BY NOT RELEASING THE "RELEASED" BUFFER
000520  * UNTIL IT IS KNOWN THAT ANOTHER BUFFER WILL NOT BE MOVED INTO ITS LOC.
000521  *
000522                  REP         60
000523  *
000524  BUFCOMPACT      EQU         *
000525  *
000526  * FIND THE FLOATING BUFFER IN BUF.TBL WITH THE LOWEST ADDRESS.
000527  *
000528  BUFC010         LDY         #0
000529                  LDX         #BUF.CNT-1
000530  *
000531  BUFC020         LDA         PGCT.T,X
000532                  AND         #$C0                ; STRIP OUT PAGE COUNT BITS
000533                  BNE         BUFC030
000534  *
000535                  LDA         ADRH.T,X
000536                  CMP         ADRH.T,Y
000537                  LDA         XBYTE.T,X
000538                  SBC         XBYTE.T,Y
000539                  BCS         BUFC030
000540  *
000541                  TXA                             ; SMALLER BUFFER FOUND, SAVE IN Y
000542                  TAY
000543  *
000544  BUFC030         DEX
000545                  BNE         BUFC020
000546  *
000547  * IF NO BUFFER FOUND THEN DONE
000548  *
000549                  TYA
000550                  BNE         BUFC040
000551                  JMP         BUFC.EXIT
000552  BUFC040         STY         BUFC.BNUM           ; OTHERWISE SAVE BUF# IN Y REG.
000553  *
000554  * CALL FIND.SEG:  FINDS HIGHEST AVAILABLE FREE MEMORY
000555  *
000556                  LDA         PGCT.T,Y
000557                  AND         #$3F                ; STRIP OUT "FREE","FIXED" FLAGS
000558                  CLC
000559                  ADC         #1
000560                  JSR         FSEG
000561                  BCS         BUFC.EXIT           ; DONE IF NO FREE SEG FOUND
000562  *
```



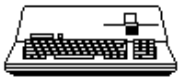
```
000563  * CONVERT BASE.BKPG TO BUFFER ADR
000564  *
000565                  LDX         F.BASEX             ; BASE BANK
000566                  LDY         F.BASEX+1           ; BASE PAGE
000567                  JSR         CNVRT.ADR
000568                  STX         F.BASEX             ; XBYTE
000569                  STY         F.BASEX+1           ; ADRH
000570  *
000571  * IF NEW SEG'S BASE < CURRENT BUFFER'S BASE ADR THEN DONE
000572  *
000573                  LDY         BUFC.BNUM
000574                  LDA         ADRH.T,Y
000575                  STA         SOURCE+1
000576                  CMP         F.BASEX+1
000577                  LDA         XBYTE.T,Y
000578                  STA         SXPAGE+SOURCE+1
000579                  SBC         F.BASEX
000580                  BCS         BUFC.EXIT1
000581  *
000582  * MOVE DATA FROM CURRENT BUFFER TO NEW BUFFER
000583  *
000584                  LDX         F.BASEX
000585                  STX         SXPAGE+DEST+1
000586                  LDY         F.BASEX+1
000587                  STY         DEST+1
000588                  LDA         #0
000589                  STA         SOURCE
000590                  STA         DEST
000591  *
000592                  TAY
000593                  LDX         F.PGCTX
000594  BUFC200         LDA         (SOURCE),Y          ; MOVE LOOP
000595                  STA         (DEST),Y
000596                  DEY
000597                  BNE         BUFC200
000598                  INC         SOURCE+1
000599                  INC         DEST+1
000600                  DEX
000601                  BNE         BUFC200
000602  *
000603  * UPDATE BUF.TBL(BUF#)
000604  *
000605                  LDY         BUFC.BNUM
000606                  LDA         F.BASEX
000607                  STA         XBYTE.T,Y
000608                  LDA         F.BASEX+1
000609                  STA         ADRH.T,Y
000610  *
000611                  LDX         SEG.T,Y
000612                  LDA         F.NUMX
000613                  STA         SEG.T,Y
000614  *
000615  * AND RELEASE OLD MEMORY SEGMENT
000616  *
000617                  STX         RLS.NUM
000618                  LDA         #RELSEG
000619                  STA         REQCODE
000620                  JSR         MMGR
000621                  BCS         BUFC.ERR
000622  *
000623                  JMP         BUFC010             ; REPEAT COMPACTION CYCLE
000624  *
000625  *
000626  BUFC.EXIT1      LDX         F.NUMX              ; DONE,
000627                  STX         RLS.NUM             ; RELEASE SEG BEFORE EXIT
000628                  LDA         #RELSEG
000629                  STA         REQCODE
000630                  JSR         MMGR
000631                  BCS         BUFC.ERR
000632  *
000633  BUFC.EXIT       LDA         #0
000634                  STA         SERR                ; MASK OUT ANY ERROR FROM MEMORY MGR
000635                  CLC
000636                  RTS                             ; NORMAL EXIT
000637  *
000638  *
000639  BUFC.ERR        LDA         #BADBUFNUM
000640                  JSR         SYSDEATH
000641                  PAGE
000642                  REP         60
000643  *
```



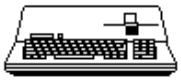
```
000644  * FSEG
000645  *
000646  * INPUT:  PAGE.CNT (A)
000647  * OUTPUT: PAGE.CNT (A) UNCHANGED IF FIND.SEG SUCCESSFUL
000648  * ERROR:  CARRY SET "UNABLE TO FIND MEMORY SEG OF PAGE.CNT*256 BYTES"
000649  *
000650  * THIS ROUTINE BUILDS THE PARAMETERS FOR A FIND.SEG SYSTEM CALL
000651  * AND THEN CALLS THE MEMORY MANAGER.
000652  *
000653                  REP         60
000654  *
000655  FSEG            EQU         *
000656  *
000657  * SETUP INPUT PARAMETERS FOR FIND.SEG CALL
000658  *
000659                  STA         F.PGCTX
000660                  LDA         #FINDSEG
000661                  STA         REQCODE
000662                  LDA         #2
000663                  STA         SRCHMODE
000664                  LDA         #4
000665                  STA         F.ID
000666  *
000667  * SETUP OUTPUT PARAMETER ADRESSES
000668  *
000669                  LDA         #>F.PGCTX
000670                  STA         F.PGCT
000671                  LDA         #<F.PGCTX
000672                  STA         F.PGCT+1
000673                  LDA         #>F.BASEX
000674                  STA         F.BASE
000675                  LDA         #<F.BASEX
000676                  STA         F.BASE+1
000677                  LDA         #>F.LIMX
000678                  STA         F.LIM
000679                  LDA         #<F.LIMX
000680                  STA         F.LIM+1
000681                  LDA         #>F.NUMX
000682                  STA         F.NUM
000683                  LDA         #<F.NUMX
000684                  STA         F.NUM+1
000685  *
000686                  LDA         #0
000687                  STA         F.PGCTX+1
000688                  STA         SXPAGE+F.PGCT+1
000689                  STA         SXPAGE+F.BASE+1
000690                  STA         SXPAGE+F.LIM+1
000691                  STA         SXPAGE+F.NUM+1
000692  *
000693                  JSR         MMGR
000694                  LDA         F.PGCTX
000695  *
000696                  RTS                             ; EXIT.  CARRY SET->ERR
000697                  PAGE
000698                  REP         60
000699  *
000700  * GETFREE
000701  *
000702  * INPUT:  NONE
000703  * OUTPUT: BUF# (X)
000704  * ERROR:  "BUFTBLFULL" SYSERR
000705  *
000706  * THIS ROUTINE SEARCHES THE BUFFER TABLE, LOOKING FOR A FREE
000707  * ENTRY.  IF FOUND, IT RETURNS THE BUFFER NUMBER, ELSE ERROR.
000708  *
000709                  REP         60
000710  *
000711  GETFREE         EQU         *
000712                  LDX         #BUF.CNT-1
000713  GFR010          LDA         PGCT.T,X
000714                  BMI         GFR.EXIT            ; FREE ENTRY FOUND
000715                  DEX
000716                  BNE         GFR010
000717  *
000718                  LDA         #BUFTBLFULL
000719                  JSR         SYSERR              ; ERR EXIT
000720  *
000721  GFR.EXIT        CLC
000722                  RTS                             ; NORMAL EXIT
000723                  PAGE
000724                  REP         60
```



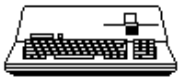
```
000725  *
000726  * CNVRT.ADR
000727  *
000728  * INPUT:  BANK VALUE (X)
000729  *         PAGE VALUE (Y)
000730  * OUTPUT: XBYTE (X)
000731  *         ADRH  (Y)
000732  * ERROR:  NONE.
000733  *
000734  * THIS ROUTINE CONVERTS A BASE.BKPG PARM (MMGR) INTO A
000735  * VIRTUAL POINTER
000736  *
000737                  REP        60
000738  *
000739  CNVRT.ADR       EQU        *
000740  *
000741  * IF PAGE <> $20 THEN GOTO L2
000742  *
000743                  CPY        #$20
000744                  BNE        CNVA020
000745  *
000746  * IF BANK <> 0 THEN GOTO L1
000747  *
000748                  TXA
000749                  BNE        CNVA010
000750  *
000751  * XBYTE=$8F
000752  * ADRH:=PAGE
000753  *
000754                  LDX        #$8F
000755                  BMI        CNVA.EXIT
000756  *
000757  * L1: XBYTE:=(BANK-1) ORA #$80
000758  *     ADRH:=#$80
000759  *
000760  CNVA010         ORA        #$80
000761                  TAX
000762                  DEX
000763                  LDY        #$80
000764                  BMI        CNVA.EXIT
000765  *
000766  * L2: XBYTE:=BANK ORA #$80
000767  *     ADRH:=ADRH-#$20
000768  *
000769  CNVA020         TXA
000770                  ORA        #$80
000771                  TAX
000772                  SEC
000773                  TYA
000774                  SBC        #$20
000775                  TAY
000776  *
000777  CNVA.EXIT       RTS
000778  *
000779                  LST        ON
000780  ZZEND           EQU        *
000781  ZZLEN           EQU        ZZEND-ZZORG
000782                  IFNE       ZZLEN-LENBUFMG
000783                  FAIL       2,"SOSORG            FILE IS INCORRECT FOR BUFMGR"
000784                  FIN
000785
000786  ********************************************************************
000787  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: BUFMGR.SRC
000788  ********************************************************************
000789

End of File -- Lines: 789 Characters: 18954
```



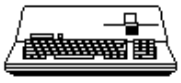
```
================================================================================
FILE: "SOS.C.BI2.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: C.BI2
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  :TABS 17,23,40
000007  ::PR#1,L58        132N
000008  SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
000009
000010  ********************************************************************
000011  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: C.BI2
000012  ********************************************************************

End of File -- Lines: 12 Characters: 523
```



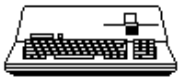
```
================================================================================
FILE: "SOS.C.S.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: C.S
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  :TABS 17,23,40
000007  ::PR#1,L58        132N
000008  SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
000009  SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
000010  END
000011
000012  ********************************************************************
000013  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: C.S
000014  ********************************************************************

End of File -- Lines: 14 Characters: 555
```



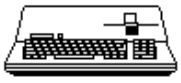
```
================================================================================
FILE: "SOS.C3.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: C3
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  :TABS 17,23,40
000007  ::PR#1,L58        132N
000008  SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
000009  SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
000010  SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
000011  END
000012
000013  ********************************************************************
000014  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: C3
000015  ********************************************************************

End of File -- Lines: 15 Characters: 590
```



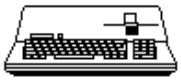
```
================================================================================
FILE: "SOS.CFMGR.SRC.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: CFMGR.SRC
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1  CHARACTER FILE MANAGER"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGCFM
000010  ZZORG           EQU        *
000011                  MSB        OFF
000012                  REP        60
000013  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000014  *                     ALL RIGHTS RESERVED
000015                  REP        60
000016  *
000017  * CHARACTER FILE MANAGER (VERSION = 1.1O   )
000018  *                        (DATE    = 8/04/81)
000019  *
000020  * THIS MODULE TRANSFORMS CHARACTER FILE SYSTEM CALLS INTO
000021  * DEVICE CALLS TO THE APPROPRIATE DEVICE HANDLER.  ONLY
000022  * OPEN, NEWLINE, READ, WRITE AND CLOSE CALLS ARE PERMITTED
000023  * ON CHARACTER FILES.
000024  *
000025                  REP        60
000026  *
000027                  ENTRY      CFMGR
000028  *
000029                  ENTRY      CFCB.MAX
000030                  ENTRY      CFCB.DEV
000031  *
000032                  EXTRN      DMGR
000033                  EXTRN      LEVEL
000034                  EXTRN      MAX.DNUM
000035                  EXTRN      SXPAGE
000036  *
000037                  EXTRN      SYSERR
000038                  EXTRN      SERR
000039                  EXTRN      BADSCNUM
000040                  EXTRN      CFCBFULL
000041                  EXTRN      BADREFNUM
000042                  EXTRN      FNFERR
000043                  PAGE
000044                  REP        60
000045  *
000046  * DATA DECLARATIONS
000047  *
000048                  REP        60
000049  *
000050  * FILE CALL PARM LOCATIONS ON SOS ZPAGE
000051  *
000052  F.TPARMX        EQU        $A0
000053  REQCODE         EQU        F.TPARMX
000054  O.PATH          EQU        F.TPARMX+1          ; OPEN'S PATHNAME LOC
000055  O.REFNUM        EQU        F.TPARMX+3          ; OPEN'S REFNUM LOC
000056  REFNUM          EQU        F.TPARMX+1          ; REFNUM'S LOC IN OTHER CALLS
000057  NL.ISNL         EQU        F.TPARMX+2          ; NEWLINE'S ISNEWLINE LOC
000058  NL.NLCHR        EQU        F.TPARMX+3          ; NEWLINE'S NEWLINECHAR LOC
000059  RW.BUF          EQU        F.TPARMX+2          ; READ/WRITE'S BUF LOC
000060  RW.BYTES        EQU        F.TPARMX+4          ; READ/WRITE'S BYTES LOC
000061  RD.BYTESRD      EQU        F.TPARMX+6          ; READ'S BYTESREAD LOC
000062  *
000063  * FILE REQUEST CODE VALUES
000064  *
000065  OPEN            EQU        8
000066  NEWLINE         EQU        9
000067  READ            EQU        $A
000068  WRITE           EQU        $B
000069  CLOSE           EQU        $C
000070                  PAGE
000071  * DEVICE CALL PARM LOCATIONS ON SOS ZPAGE
000072  *
000073  D.TPARMX        EQU        $C0
000074  D.SCNUM         EQU        D.TPARMX            ; DEVICE SYS CALL # LOC
000075  GDN.DNAME       EQU        D.TPARMX+1          ; GETDEVNUM DNAME LOC
000076  GDN.DNUM        EQU        D.TPARMX+3          ; GETDEVNUM DNUM LOC
```



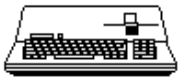
```
000077  D.DNUM          EQU         D.TPARMX+1          ; OPN/CLOSE/RD/WR/CTRL'S DNUM LOC
000078  DRW.BUF         EQU         D.TPARMX+2          ; RD/WR'S BUF LOC
000079  DRW.BYTES       EQU         D.TPARMX+4          ; RD/WR'S BYTES LOC
000080  DRD.BYTESRD     EQU         D.TPARMX+8          ; RD/WR'S BYTESREAD LOC
000081  DC.CCODE        EQU         D.TPARMX+2          ; DCTRL'S CTRLCODE LOC
000082  DC.CLIST        EQU         D.TPARMX+3          ; DCTRL'S CTRLLIST LOC
000083  *
000084  * DEVICE REQUEST CODE VALUES
000085  *
000086  DREAD           EQU         $0
000087  DWRITE          EQU         $1
000088  DCTRL           EQU         $3
000089  GETDEVNUM       EQU         $4
000090  DOPEN           EQU         $6
000091  DCLOSE          EQU         $7
000092  *
000093  CTRL.LIST       DS          2                   ; CONTAINER FOR NEWLINE DCTRL CALL
000094  NEWLINECC       EQU         2                   ; NEWLINE CTRL CODE
000095  *
000096  * GETDNUM VARS
000097  *
000098  DNUM.TEMP       DS          1
000099  *
000100  * CLOSEALL VARS
000101  *
000102  DCLOSE.ERR      EQU         F.TPARMX+$F
000103  DCLOSE.TBL      EQU         $200
000104  TRUE            EQU         $80
000105  FALSE           EQU         $0
000106  *
000107  *
000108                  REP         60
000109  *
000110  * CHARACTER FILE CONTROL BLOCK TABLE
000111  * (ENTRY 0 IS NOT USED)
000112  *
000113                  REP         60
000114  CFCB.MAX        EQU         17
000115  CFCB.DEV        DS          CFCB.MAX
000116  CFCB.LVL        DS          CFCB.MAX
000117                  PAGE
000118                  REP         60
000119  *
000120  * CHARACTER FILE MANAGER - MAIN ENTRY POINT
000121  *
000122                  REP         60
000123  CFMGR           EQU         *
000124  *
000125  * SWITCH, BASED ON REQUEST CODE
000126  *
000127                  LDA         REQCODE
000128                  CMP         #OPEN
000129                  BEQ         CFOPEN              ; "OPEN"
000130                  CMP         #NEWLINE
000131                  BEQ         CFNEWLINE           ; "NEWLINE"
000132                  CMP         #READ
000133                  BEQ         CFREAD              ; "READ"
000134                  CMP         #WRITE
000135                  BNE         CFM010
000136                  JMP         CFWRITE             ; "WRITE"
000137  CFM010          CMP         #CLOSE
000138                  BNE         CFM020
000139                  JMP         CFCLOSE             ; "CLOSE"
000140  CFM020          LDA         #BADSCNUM
000141                  JSR         SYSERR              ; ERR EXIT
000142                  PAGE
000143                  REP         60
000144  * OPEN(IN.PATHNAME; OUT.REFNUM; IN.OPENLIST,LENGTH) SYSTEM CALL
000145                  REP         60
000146  CFOPEN          EQU         *                   ; BUILD "D.OPEN" CALL
000147                  JSR         GETDNUM             ; MAP PATH TO DEV#
000148                  BCS         CFOP.ERR1           ; ERR - FILE NOT FOUND
000149                  STA         D.DNUM
000150  *
000151                  JSR         REQ.CFCB            ; BUILD NEW CFCB ENTRY
000152                  BCS         CFOP.ERR1           ; ERR - CFCB FULL
000153                  LDX         #0
000154                  STA         (O.REFNUM,X)        ; RETURN REFNUM TO CALLER
000155                  CPY         #1
000156                  BNE         CFOP.EXIT           ; DEVICE ALREADY OPEN
000157  *
```



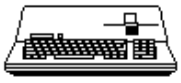
```
000158                  LDA         #DOPEN
000159                  STA         D.SCNUM
000160                  JSR         DMGR                ; DOPEN CALL
000161                  BCS         CFOP.ERR
000162  CFOP.EXIT       RTS                             ; NORMAL EXIT
000163  *
000164  CFOP.ERR        LDA         SERR                ;KLUDGE - 1.0 DRIVERS DON'T SUPPORT CARRY ERR PROTOCOL
000165                  BEQ         CFOP.EXIT           ;NO ERROR
000166                  LDX         #0                  ; RELEASE CFCB ENTRY
000167                  LDA         (O.REFNUM,X)
000168                  JSR         REL.CFCB
000169  CFOP.ERR1       RTS                             ; ERR EXIT
000170                  PAGE
000171                  REP         60
000172  * NEWLINE(IN.REFNUM,IS        .NEWLINE,NEWLINE.CHAR) SYSTEM CALL
000173                  REP         60
000174  CFNEWLINE       EQU         *                   ; BUILD "D.CONTROL" CALL
000175                  LDA         #DCTRL
000176                  STA         D.SCNUM
000177                  LDA         REFNUM
000178                  JSR         GET.CFCB            ; MAP REFNUM TO DEV #
000179                  BCS         CFNL.ERR            ; ERR - BAD REFNUM
000180  *
000181                  STA         D.DNUM
000182                  LDA         #NEWLINECC
000183                  STA         DC.CCODE
000184  *
000185                  LDA         #>CTRL.LIST
000186                  STA         DC.CLIST
000187                  LDA         #<CTRL.LIST
000188                  STA         DC.CLIST+1
000189                  LDA         #0
000190                  STA         SXPAGE+DC.CLIST+1
000191  *
000192                  LDA         NL.ISNL
000193                  STA         CTRL.LIST
000194                  LDA         NL.NLCHR
000195                  STA         CTRL.LIST+1
000196  *
000197                  JSR         DMGR                ; DCONTROL CALL
000198                  RTS                             ; NORMAL EXIT
000199  *
000200  CFNL.ERR        RTS                             ; ERR EXIT
000201                  PAGE
000202                  REP         60
000203  * READ(IN.REFNUM,BUF,BYTES,BYTESREAD) SYSTEM CALL
000204                  REP         60
000205  CFREAD          EQU         *                   ; BUILD "D.READ" CALL
000206                  LDA         #DREAD
000207                  STA         D.SCNUM
000208                  LDA         REFNUM
000209                  JSR         GET.CFCB            ; MAP REFNUM TO DEV #
000210                  BCS         CFRD.ERR            ; ERR - BAD REFNUM
000211  *
000212                  STA         D.DNUM
000213                  LDX         #3
000214  CFRD010         LDA         RW.BUF,X
000215                  STA         DRW.BUF,X
000216                  DEX
000217                  BPL         CFRD010
000218  *
000219                  LDA         RD.BYTESRD
000220                  STA         DRD.BYTESRD
000221                  LDA         RD.BYTESRD+1
000222                  STA         DRD.BYTESRD+1
000223  *
000224                  LDA         SXPAGE+RW.BUF+1
000225                  STA         SXPAGE+DRW.BUF+1
000226                  LDA         SXPAGE+RW.BYTES+1
000227                  STA         SXPAGE+DRW.BYTES+1
000228                  LDA         SXPAGE+RD.BYTESRD+1
000229                  STA         SXPAGE+DRD.BYTESRD+1
000230  *
000231                  JSR         DMGR                ; DREAD CALL
000232                  RTS                             ; NORMAL EXIT
000233  *
000234  CFRD.ERR        RTS                             ; ERR EXIT
000235                  PAGE
000236                  REP         60
000237  * WRITE(IN.REFNUM,BUF,BYTES) SYSTEM CALL
000238                  REP         60
```



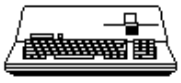
```
000239  CFWRITE         EQU     *                   ; BUILD "D.WRITE" CALL
000240                  LDA     #DWRITE
000241                  STA     D.SCNUM
000242                  LDA     REFNUM
000243                  JSR     GET.CFCB            ; MAP REFNUM TO DEV #
000244                  BCS     CFWR.ERR            ; ERR - BAD REFNUM
000245                  STA     D.DNUM
000246                  LDX     #3
000247  CFWR010         LDA     RW.BUF,X
000248                  STA     DRW.BUF,X
000249                  DEX
000250                  BPL     CFWR010
000251                  LDA     SXPAGE+RW.BUF+1
000252                  STA     SXPAGE+DRW.BUF+1
000253                  LDA     SXPAGE+RW.BYTES+1
000254                  STA     SXPAGE+DRW.BYTES+1
000255  *
000256                  JSR     DMGR                ; DWRITE CALL
000257                  RTS                         ; NORMAL EXIT
000258  *
000259  CFWR.ERR        RTS                         ; ERR EXIT
000260                  PAGE
000261                  REP     60
000262  * CLOSE(IN.REFNUM) SYSTEM CALL
000263                  REP     60
000264  CFCLOSE         EQU     *                   ; BUILD "D.CLOSE" CALL
000265                  LDA     #DCLOSE
000266                  STA     D.SCNUM
000267                  LDA     REFNUM
000268                  BEQ     CLOSEALL
000269  *
000270                  JSR     REL.CFCB            ; RELEASE CFCB ENTRY
000271                  BCS     CFCL010
000272                  STA     D.DNUM
000273                  TYA
000274                  BNE     CFCL010
000275                  JSR     DMGR                ; DCLOSE CALL
000276  CFCL010         RTS                         ; NORMAL EXIT
000277  *
000278                  PAGE
000279                  REP     60
000280  *
000281  * CLOSE ALL CHARACTER FILES W/LEVELS >= TO CURRENT SYSTEM FILE LEVEL.
000282  *
000283                  REP     60
000284  *
000285  CLOSEALL        EQU     *
000286                  LDA     #FALSE              ; SET ENTRIES IN DEV CLOSE TBL TO FALSE
000287                  LDX     MAX.DNUM
000288  CFCL020         STA     DCLOSE.TBL,X
000289                  DEX
000290                  BPL     CFCL020
000291  *
000292                  LDX     #CFCB.MAX-1         ; CLOSE ALL DEVICES >= TO CURRENT LEVEL
000293  CFCL030         LDA     CFCB.DEV,X          ; AND MARK TRUE IN DEV CLOSE TBL
000294                  TAY
000295                  BMI     CFCL050
000296                  LDA     CFCB.LVL,X
000297                  CMP     LEVEL
000298                  BCC     CFCL050
000299                  LDA     #TRUE
000300                  STA     DCLOSE.TBL,Y
000301                  SEC
000302                  ROR     CFCB.DEV,X
000303  CFCL050         DEX
000304                  BNE     CFCL030
000305  *
000306                  LDX     #CFCB.MAX-1         ; DON'T CLOSE DEVICES < CURRENT LEVEL
000307  CFCL060         LDA     CFCB.DEV,X
000308                  TAY
000309                  BMI     CFCL070
000310                  LDA     #FALSE
000311                  STA     DCLOSE.TBL,Y
000312  CFCL070         DEX
000313                  BNE     CFCL060
000314  *
000315                  LDA     #0
000316                  STA     DCLOSE.ERR
000317                  LDX     MAX.DNUM            ; ISSUE D'CLOSE CALLS TO ALL DEVICES MARKED AS TRUE
000318  CFCL080         LDA     DCLOSE.TBL,X        ; IN DEV CLOSE TABLE
000319                  BPL     CFCL090
```



```
000320                  TXA
000321                  PHA
000322                  STX         D.DNUM
000323                  JSR         DMGR
000324                  PLA
000325                  TAX
000326                  LDA         SERR
000327                  BEQ         CFCL090             ; IF ERROR,
000328                  STA         DCLOSE.ERR          ; THEN SAVE IT
000329  CFCL090         DEX
000330                  BNE         CFCL080
000331  *
000332                  LDA         DCLOSE.ERR          ; IF $0 THEN NO ERRORS FROM D.CLOSE CALLS
000333                  BNE         CFCL.ERR
000334                  RTS                             ; NORMAL EXIT
000335  CFCL.ERR        JSR         SYSERR              ; RETURN LAST D.CLOSE ERROR REPORTED
000336                  PAGE
000337                  REP         60
000338  *
000339  * GET DEVICE NUMBER
000340  *
000341  * INPUT:  CPATH
000342  * OUTPUT: DEVICE NUMBER (A)
000343  * ERROR:  CARRY SET ("FILE NOT FOUND")
000344  *
000345  * GETDNUM FIRST CALLS THE DMGR (GETDEVNUM) MAP THE PATHNAME
000346  * TO A DEVICE #.  GETDNUM THEN ENSURES THAT THE PATHNAME
000347  * IS NOT A BLOCK DEVICE BY CHECKING THE DBLKLST TABLE.
000348  *
000349                  REP         60
000350  *
000351  GETDNUM         EQU         *
000352                  LDA         #GETDEVNUM
000353                  STA         D.SCNUM
000354  *
000355                  LDA         O.PATH
000356                  STA         GDN.DNAME
000357                  LDA         O.PATH+1
000358                  STA         GDN.DNAME+1
000359  *
000360                  LDA         #>DNUM.TEMP
000361                  STA         GDN.DNUM
000362                  LDA         #<DNUM.TEMP
000363                  STA         GDN.DNUM+1
000364  *
000365                  LDA         SXPAGE+O.PATH+1
000366                  STA         SXPAGE+GDN.DNAME+1
000367                  LDA         #0
000368                  STA         SXPAGE+GDN.DNUM+1
000369  *
000370                  JSR         DMGR
000371                  BCS         GETD.ERR            ; D.NAME NOT FOUND
000372                  BMI         GETD.ERR            ; BLOCK DEVICE FOUND
000373                  LDA         DNUM.TEMP
000374                  RTS
000375  *
000376  GETD.ERR        LDA         #FNFERR
000377                  JSR         SYSERR
000378                  PAGE
000379                  REP         60
000380  * REQUEST FCB ENTRY
000381  *
000382  * INPUT: DNUM (A)
000383  * OUTPUT: REFNUM (A), OPENCT (Y)
000384  * ERROR:  CARRY SET ("CFCB FULL")
000385  *
000386  * REQ.CFCB FIRST SEARCHES THE CFCB TABLE USING THE DEV#
000387  * AS A KEY.  IF FOUND THE OPENCT IS INCREMENTED, OTHERWISE,
000388  * REQ.CFCB FINDS A FREE ENTRY AND STORES THE DEV# AND LEVEL #.
000389  *
000390                  REP         60
000391  *
000392  REQ.CFCB        EQU         *
000393                  LDX         #CFCB.MAX-1
000394                  TAY
000395  REQ010          LDA         CFCB.DEV,X
000396                  BMI         REQ020
000397                  DEX
000398                  BNE         REQ010
000399                  LDA         #CFCBFULL
000400                  JSR         SYSERR
```

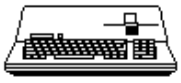


```
000401  REQ020          TYA
000402                  STA         CFCB.DEV,X
000403                  LDA         LEVEL
000404                  STA         CFCB.LVL,X
000405                  TXA
000406                  PHA
000407                  TYA
000408                  JSR         OPENCOUNT
000409                  PLA
000410                  ORA         #$80
000411                  CLC
000412                  RTS                             ; NORMAL EXIT
000413                  PAGE
000414                  REP         60
000415  *
000416  * RELEASE FCB ENTRY
000417  *
000418  * INPUT:  REFNUM (A)
000419  * OUTPUT:  DNUM (A), OPENCT (Y)
000420  * ERROR:   CARRY SET ("INVALID REFNUM")
000421  *
000422  * USES REFNUM AS AN CFCB TABLE INDEX TO RELEASE A CFCB ENTRY.
000423  *
000424                  REP         60
000425  REL.CFCB        EQU         *
000426                  AND         #$7F
000427                  CMP         #CFCB.MAX
000428                  BCS         REL.ERR
000429                  TAX
000430                  LDA         CFCB.DEV,X
000431                  BMI         REL.ERR
000432                  SEC                             ; MARK ENTRY FREE
000433                  ROR         CFCB.DEV,X
000434                  JSR         OPENCOUNT
000435                  CLC
000436                  RTS                             ; NORMAL EXIT
000437  *
000438  REL.ERR         LDA         #BADREFNUM
000439                  JSR         SYSERR
000440                  REP         60
000441  *
000442  * OPENCOUNT SUBROUTINE
000443  *
000444  * INPUT:   DEVNUM (A)
000445  * OUTPUT:  DEVNUM (A), OPENCTR (Y)
000446  *
000447  * OPENCTR:=COUNT OF ALL CFCB ENTRIES W/CFCB.DEV=DEVNUM
000448  *
000449                  REP         60
000450  OPENCOUNT       EQU         *
000451                  LDY         #0
000452                  LDX         #CFCB.MAX-1
000453  OPNCT010        CMP         CFCB.DEV,X
000454                  BNE         OPNCT020
000455                  INY
000456  OPNCT020        DEX
000457                  BNE         OPNCT010
000458                  RTS
000459                  PAGE
000460                  REP         60
000461  *
000462  * GET FCB ENTRY
000463  *
000464  * INPUT:   REFNUM (A)
000465  * OUTPUT:  DNUM (A)
000466  * ERROR:   CARRY SET ("INVALID REFNUM")
000467  *
000468  * USES REFNUM AS AN INDEX TO RETURN THE CORRESPONDING DEVICE #.
000469  * IF THE ENTRY INDICATED BY REFNUM IS A FREE ENTRY, THEN AN
000470  * ERROR, "INVALID REF NUM" IS RETURNED.
000471  *
000472                  REP         60
000473  GET.CFCB        EQU         *
000474                  AND         #$7F
000475                  CMP         #CFCB.MAX
000476                  BCS         GET.ERR
000477                  TAX
000478                  LDA         CFCB.DEV,X
000479                  BMI         GET.ERR
000480                  CLC
000481                  RTS                             ; NORMAL EXIT
```

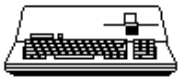


```
000482  *
000483  GET.ERR         LDA       #BADREFNUM
000484                  JSR       SYSERR              ; ERR EXIT
000485  *
000486                  LST       ON
000487  ZZEND           EQU       *
000488  ZZLEN           EQU       ZZEND-ZZORG
000489                  IFNE      ZZLEN-LENCFM
000490                  FAIL      2,"SOSORG           FILE IS INCORRECT FOR CFMGR"
000491                  FIN
000492
000493  ********************************************************************
000494  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: CFMGR.SRC
000495  ********************************************************************
000496
```

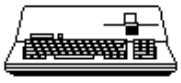
End of File -- Lines: 496 Characters: 12860



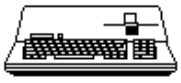
```
================================================================================
FILE: "SOS.CLOSE.EOF.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: CLOSE.EOF
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007  *
000008  *
000009  CLOSE           LDA         C.REFNUM            ; CLOSE ALL?
000010                  BNE         CLOSE1              ; NO, JUST ONE OF 'EM
000011                  STA         CFERR               ; CLEAR GLOBAL CLOSE ERROR
000012                  JSR         GFCBADR             ; SET UP POINTER TO FCB
000013  CLOSALL         LDA         #0                  ; BEGIN AT THE BEGINNING.
000014  CLSALL1         STA         FCBPTR              ; SAVE CURRENT LOW BYTE OF POINTER
000015                  LDY         #FCBLEVL            ; FETCH THE LEVEL AT WHICH
000016                  LDA         (FCBPTR),Y          ; FILE WAS OPENED
000017                  CMP         LEVEL               ; TEST AGAINST CURRENT GLOBAL LEVEL
000018                  BCC         NXTCLOS             ; DONT CLOSE IF FILES LEVEL IS < GLOBAL LEVEL
000019                  LDY         #FCBREFN            ; INDEX TO REFERENCE NUMBER
000020                  LDA         (FCBPTR),Y          ; IS THIS REFERENCE FILE OPEN?
000021                  BEQ         NXTCLOS             ; NO, TRY NEXT.
000022                  JSR         FLUSH2              ; CLEAN IT OUT...
000023                  BCS         CLOSERR             ; RETURN FLUSH ERRORS
000024                  JSR         CLOSE2              ; UPDATE FCB & VCB
000025                  LDY         C.REFNUM
000026                  BEQ         NXTCLOS             ; NO ERR IF CLOSE ALL
000027                  BCS         CLOSERR
000028  NXTCLOS         LDA         FCBPTR              ; BUMP POINTER TO NEXT FILE CONTROL BLOCK.
000029                  CLC
000030                  ADC         #$20
000031                  BCC         CLSALL1             ; BRANCH IF WITHIN SAME PAGE.
000032                  LDA         FCBPTR+1
000033                  INC         FCBPTR+1            ; BUMP TO NEXT PAGE.
000034                  CMP         FCBADDRH            ; HAVE WE CHECKED BOTH PAGES?
000035                  BEQ         CLOSALL             ; YES, RETURN NO ERROR.
000036                  CLC
000037                  LDA         CFERR               ; ON FINAL CLOSE OF CLOSE ALL REPORT LOGGED ERRORS
000038                  BEQ         C3                  ; BRANCH IF NO ERRORS
000039                  SEC
000040  C3              RTS
000041  *
000042  *
000043  CFERR           DS          1                   ; GLOBAL ERROR FLAG FOR FLUSH AND CLOSE ALL
000044  *
000045  *
000046  CLOSE1          JSR         FLUSH1              ; FLUSH FILE FIRST (INCLUDING UPDATING BIT MAP)
000047                  BCS         CLOSERR
000048  CLOSE2          LDY         #FCBBUFN
000049                  LDA         (FCBPTR),Y
000050                  JSR         RELBUF
000051                  BCS         CLOSERR
000052                  LDA         #0
000053                  LDY         #FCBREFN
000054                  STA         (FCBPTR),Y
000055                  INY                             ; BUMP TO 'FCBDEVN'
000056                  LDA         (FCBPTR),Y
000057                  STA         DEVNUM              ; GO LOOK FOR ASSOCIATED VCB.
000058                  JSR         DEVVCB
000059                  LDX         VCBPTR              ; GET VCBPTR
000060                  DEC         VCB+VCBOPNC,X       ; INDICATE ONE LESS FILE OPEN.
000061                  BNE         CLOSEND             ; BRANCH IF THAT WASN'T THE LAST...
000062                  LDA         VCB+VCBSTAT,X
000063                  AND         #$7F                ; STRIP 'FILES OPEN' BIT
000064                  STA         VCB+VCBSTAT,X
000065  CLOSEND         CLC
000066                  RTS
000067  CLOSERR         JMP         GLBERR              ; DON'T REPORT CLOSALL ERR NOW
000068  *
000069                  PAGE
000070  *
000071  FLUSH           LDA         C.REFNUM            ; FLUSH ALL?
000072                  BNE         FLUSH1              ; NO, JUST ONE OF 'EM
000073                  STA         CFERR               ; CLEAR GLOBAL FLUSH ERROR
000074                  JSR         GFCBADR             ; SET UP POINTER TO FCB
000075  FLSHALL         LDA         #0                  ; BEGIN AT THE BEGINNING.
000076  FLSHAL1         STA         FCBPTR              ; SAVE CURRENT LOW BYTE OF POINTER
```



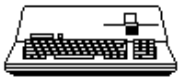
```
000077                  LDY     #FCBREFN            ; INDEX TO REFERENCE NUMBER
000078                  LDA     (FCBPTR),Y          ; IS THIS REFERENCE FILE OPEN?
000079                  BEQ     NXFLUSH             ; NO, TRY NEXT.
000080                  JSR     FLUSH2              ; CLEAN IT OUT...
000081                  BCS     FLSHERR             ; RETURN ANY ERRORS
000082  *
000083                  BCS     CLOSERR
000084  NXFLUSH         LDA     FCBPTR              ; BUMP POINTER TO NEXT FILE CONTROL BLOCK.
000085                  CLC
000086                  ADC     #$20
000087                  BCC     FLSHAL1             ; BRANCH IF WITHIN SAME PAGE.
000088                  LDA     FCBPTR+1
000089                  INC     FCBPTR+1            ; BUMP TO NEXT PAGE.
000090                  CMP     FCBADDRH            ; HAVE WE CHECKED BOTH PAGES?
000091                  BEQ     FLSHALL             ; YES, RETURN NO ERROR.
000092  FLUSHEND        CLC
000093                  LDA     CFERR               ; ON LAST FLUSH OF A FLUSH(0)
000094                  BEQ     F3                  ; BRANCH IF NO LOGGED ERRORS
000095                  SEC                         ; REPORT ERROR NOW
000096  F3              RTS
000097  FLSHERR         JMP     GLBERR              ; FLUSH ALL OR ONE?
000098  *
000099  FLUSH2          JSR     FNDFCBUF            ; MUST SET UP ASSOCIATED VCB AN BUFFER LOCATIONS FIRST.
000100                  BCC     FLUSH2A             ; BRANCH IF NO ERROR ENCOUNTERED.
000101                  JMP     GLBERR              ; CHECK FOR CLOSE OR FLUSH ALL
000102  *
000103  FLUSH1          LDA     #0                  ; CLEAR
000104                  STA     CFERR               ; GLOBAL ERROR FOR NORMAL REFNUM FLUSH
000105                  JSR     FINDFCB             ; SET UP POINTER TO FCB USER REFERENCES
000106                  BCS     FLSHERR             ; RETURN ANY ERRORS
000107  FLUSH2A         LDY     #FCBATTR            ; TEST TO SEE IF FILE IS
000108                  LDA     (FCBPTR),Y          ; MODIFIED. FIRST TEST WRITE ENABLED.
000109                  AND     #WRITEN
000110                  BEQ     FLUSHEND            ; BRANCH IF 'READ ONLY'
000111                  LDY     #FCBDIRTY           ; SEE IF EOF HAS BEEN MODIFIED
000112                  LDA     (FCBPTR),Y
000113                  BMI     FLUSH2B             ; BRANCH IF IT HAS
000114                  LDY     #FCBSTAT            ; NOW TEST FOR DATA MODIFIED.
000115                  LDA     (FCBPTR),Y          ; (IN OTHER WORDS: WAS FILE ACTUALLY
000116                  AND     #USEMOD+EOFMOD+DATMOD ; WRITTEN TO WHILE IT'S BEEN OPEN?)
000117                  BEQ     FLUSHEND            ; BRANCH IF FILE NOT MODIFIED.
000118  FLUSH2B         JSR     TWRPROT1            ; DISK SWITCH CHECKING
000119                  LDA     DSWGLOB
000120                  BEQ     FLUSH2C             ; BRANCH IF NO SWITCH
000121                  LDA     #XDISKSW
000122                  SEC
000123                  RTS                         ; FORCES A VERIFIED RETRY
000124  FLUSH2C         LDY     #FCBSTAT            ; NOW TEST FOR DATA MODIFIED.
000125                  LDA     (FCBPTR),Y
000126                  AND     #DATMOD             ; DOES CURRENT DATA BUFFER NEED TO BE
000127                  BEQ     FLUSH3              ; WRITTEN? BRANCH IF NOT.
000128                  JSR     WFCBDAT             ; IF SO, GO WRITE IT STUPID!
000129                  BCS     FLSHERR
000130  FLUSH3          LDY     #FCBSTAT            ; CHECK TO SEE IF THE INDEX BLOCK (TREE FILES ONLY)
000131                  LDA     (FCBPTR),Y          ; NEEDS TO BE WRITTEN.
000132                  AND     #IDXMOD
000133                  BEQ     FLUSH4              ; BRANCH IF NOT...
000134                  JSR     WFCBIDX
000135                  BCS     FLSHERR             ; RETURN ANY ERRORS.
000136                  PAGE
000137  *
000138  FLUSH4          LDY     #FCBENTN            ; NOW PREPARE TO UPDATE DIRECTORY
000139  OWNRMOV         LDA     (FCBPTR),Y          ; NOTE: THIS CODE DEPENDS ON THE
000140                  STA     D.DEV-FCBDEVN,Y     ; DEFINED ORDER OF THE FILE CONTROL
000141                  DEY                         ; BLOCK AND THE TEMPORARY DIRECTORY AREA IN 'WORKSPC'! *************
000142                  CPY     #FCBDEVN-1
000143                  BNE     OWNRMOV
000144                  LDA     D.HEAD              ; READ IN THE DIRECTORY HEADER FOR THIS FILE
000145                  STA     BLOKNML
000146                  LDA     D.HEAD+1
000147                  STA     BLOKNMH
000148                  LDA     D.DEV
000149                  STA     DEVNUM
000150                  JSR     RDGBUF              ; READ IT INTO THE GENERAL PURPOSE BUFFER
000151                  BCS     FLSHERR             ; BRANCH IF ERROR.
000152                  JSR     MOVHED0             ; MOVE HEADER INFO.
000153                  LDA     D.ENTBLK            ; GET ADDRESS OF DIRECTORY BLOCK THAT
000154                  LDY     D.ENTBLK+1          ; CONTAINS THE FILE ENTRY.
000155                  CMP     D.HEAD              ; TEST TO SEE IF IT'S THE SAME BLOCK THAT
000156                  BNE     FLSHEBLK            ; THE HEADER IS IN. BRANCH IF NOT.
000157                  CPY     D.HEAD+1
```



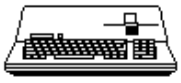
```
000158                  BEQ         FLUSH5                ; BRANCH IF HEADER BLOCK = ENTRY BLOCK.
000159  FLSHEBLK        STA         BLOKNML
000160                  STY         BLOKNMH
000161                  JSR         RDGBUF                ; GET BLOCK WITH FILE ENTRY IN GENERAL BUFFER.
000162  FLUSH5          JSR         ENTCALC               ; SET UP POINTER TO ENTRY
000163                  JSR         MOVENTRY              ; MOVE ENTRY TO TEMP ENTRY BUFFER IN 'WORKSPC'
000164                  LDY         #FCBUSE               ; UPDATE 'BLOCKS USED' COUNT.
000165                  LDA         (FCBPTR),Y
000166                  STA         DFIL+D.USAGE
000167                  INY
000168                  LDA         (FCBPTR),Y
000169                  STA         DFIL+D.USAGE+1        ; HI BYTE TOO...
000170                  LDY         #FCBEOF               ; AND MOVE IN END OF FILE MARK WHETHER
000171  EOFUPDTE        LDA         (FCBPTR),Y            ; WE NEED TO OR NOT.
000172                  STA         DFIL+D.EOF-FCBEOF,Y
000173                  INY                               ; MOVE ALL THREE BYTES.
000174                  CPY         #FCBEOF+3
000175                  BNE         EOFUPDTE
000176                  LDY         #FCBFRST              ; ALSO MOVE IN THE ADDRESS OF
000177                  LDA         (FCBPTR),Y            ; THE FILE'S FIRST BLOCK SINCE
000178                  INY                               ; IT MIGHT HAVE CHANGED SINCE THE FILE
000179                  STA         DFIL+D.FRST           ; FIRST OPENED.
000180                  LDA         (FCBPTR),Y
000181                  STA         DFIL+D.FRST+1
000182                  PAGE
000183                  LDY         #FCBSTYP              ; AND THE LAST THING TO UPDATE IS
000184                  LDA         (FCBPTR),Y            ; THE STORAGE TYPE.
000185                  ASL         A                     ; (SHIFT IT INTO THE HI NIBBLE)
000186                  ASL         A
000187                  ASL         A
000188                  ASL         A
000189                  STA         SCRTCH
000190                  LDA         DFIL+D.STOR           ; GET OLD TYPE BYTE (IT MIGHT BE THE SAME)
000191                  AND         #$F                   ; STRIP OFF OLD TYPE
000192                  ORA         SCRTCH                ; ADD IN THE NEW TYPE,
000193                  STA         DFIL+D.STOR           ; AND PUT IT AWAY.
000194                  JSR         DREVISE               ; GO UPDATE DIRECTORY!
000195                  BCS         FLUSHERR
000196                  LDY         #FCBDIRTY             ; MARK
000197                  LDA         (FCBPTR),Y            ; FCB/DIRECTORY
000198                  AND         #$FF-FCBMOD           ; AS
000199                  STA         (FCBPTR),Y            ; UNDIRTY
000200                  LDX         #0                    ; NOW CHECK TO SEE IF A BIT MAP
000201                  LDA         D.DEV                 ; IS LYING AROUND THAT SHOULD BE WRITTEN.
000202                  CMP         BMADEV                ; IS IT IN MAP BUFFER A?
000203                  BEQ         BMAPUP                ; YES, PUT IT ON THE DISK IF NECESSARY.
000204                  LDX         #BMTABSZ              ; SET INDEX TO BIT MAP TABLE 'B'
000205                  CMP         BMBDEV                ; NO, WHAT ABOUT BIT MAP BUFFER B?
000206                  BNE         FLSHEND1              ; NOPE, ALL DONE.
000207  BMAPUP          LDA         BMASTAT,X             ; TEST TO SEE IF IT'S BEEN MODIFIED.
000208                  BPL         FLSHEND1              ; NOPE, ALL DONE AS I SAID.
000209                  STX         BMTAB
000210                  JSR         WRTBMAP               ; GO PUT IT AWAY.
000211                  BCS         FLUSHERR
000212                  LDX         BMTAB                 ; MARK MAP AS UPDATED
000213                  LDA         #0
000214                  STA         BMASTAT,X
000215  FLSHEND1        CLC
000216                  RTS
000217  FLUSHERR        EQU         *                     ; DROP INTO GLBERR
000218  *
000219  GLBERR          EQU         *                     ; REPORT ERROR IMMEDIATELY
000220  * ONLY IF NOT A CLOSE ALL OR FLUSH ALL
000221                  LDX         C.REFNUM
000222                  BNE         GLBERR1               ; NOT AN 'ALL' SO REPORT NOW
000223                  CLC
000224                  STA         CFERR                 ; SAVE FOR LATER
000225  GLBERR1         RTS
000226  *
000227  *
000228  GFCBADR         LDA         FCBANKNM              ; GET BANK THAT FCB IS IN
000229                  STA         SISFCBP
000230                  LDA         FCBADDRH              ; AND HIGH BYTE ADDRESS OF FILE CONTORL BLOCK.
000231                  STA         FCBPTR+1
000232                  RTS                               ; SILLY THAT IT'S SO SHORT...
000233  *
000234  SETERR          LDA         #ACCSERR
000235                  SEC
000236  EOFRETN         RTS
000237                  PAGE
000238  *
```



```
000239  SETEOF          LDY       #FCBSTYP            ; ONLY KNOW HOW TO MOVE EOF OF TREE TYPE
000240                  LDA       (FCBPTR),Y
000241                  CMP       #TRETYP+1
000242                  BCS       SETERR              ; BRANCH IF OTHER THAN TREE
000243                  LDY       #FCBATTR            ; NOW CHECK TO INSURE WRITE IS ENABLED.
000244                  LDA       (FCBPTR),Y
000245                  AND       #WRITEN             ; CAN WE SET NEW EOF?
000246                  BEQ       SETERR              ; NOPE, ACCESS ERROR.
000247                  JSR       TSTWPROT            ; FIND OUT IF MOD IS POSIBLE (HARDWARE WRITE PROTECT)
000248                  BCS       SETERR
000249                  LDY       #FCBEOF+2           ; SAVE OLD EOF
000250                  LDX       #2                  ; SO IT CAN BE SEEN
000251  SETSAVE         LDA       (FCBPTR),Y          ; WHETHER BLOCKS NEED
000252                  STA       OLDEOF,X            ; TO BE RELEASED
000253                  DEY                           ; UPON
000254                  DEX                           ; CONTRACTION
000255                  BPL       SETSAVE             ; ALL THREE BYTES OF THE EOF
000256                  JSR       ADJMARK             ; GET ADJUSTED END OF FILE ACCORDING TO 'C.BASE' INTO TPOS.
000257                  BCS       EOFRETN             ; RETURN ANY ERROR IMMEDIATELY
000258                  LDX       #2
000259  NEOFPOS         LDA       TPOSLL,X            ; POSITION MARK TO NEW EOF
000260                  STA       C.NEWEOF,X
000261                  DEX
000262                  BPL       NEOFPOS
000263                  LDY       #FCBMARK+2          ; FIND OUT IF EOF < MARK.
000264                  LDX       #2
000265  NEOFTST         LDA       (FCBPTR),Y
000266                  CMP       C.NEWEOF,X          ; COMPARE UNTIL NOT EQUAL OR CARRY CLEAR
000267                  BCC       SETEOF1             ; BRANCH IF EOF>MARK
000268                  BNE       SETEOF0             ; BRANCH IF EOF<MARK
000269                  DEY
000270                  DEX
000271                  BPL       NEOFTST             ; LOOP ON ALL THREE BYTES
000272  SETEOF0         JSR       RDPOSN              ; READ IN NEW POSITION.
000273                  BCS       EOFRETN             ; RETURN ANY ERRORS.
000274  SETEOF1         LDX       #2
000275                  LDY       #FCBEOF+2           ; MOVE NEW EOF TO FCB.
000276  SETEOF2         LDA       C.NEWEOF,X
000277                  STA       (FCBPTR),Y
000278                  DEY
000279                  DEX
000280                  BPL       SETEOF2             ; MOVE ALL THREE BYTES.
000281                  JSR       FCBUSED             ; MARK FCB AS DIRTY (FOR FLUSH)
000282  *
000283                  LDX       #2                  ; POINT TO THIRD BYTE
000284  PURTEST         LDA       OLDEOF,X            ; SEE IF EOF MOVED BACKWARDS
000285                  CMP       C.NEWEOF,X          ; SO BLOCKS CAN
000286                  BCC       PURTEST1            ; BE RELEASED (BRANCH IF NOT)
000287                  BNE       PURGE               ; BRANCH IF BLOCKS TO BE RELEASED
000288                  DEX
000289                  BPL       PURTEST             ; ALL THREE BYTES
000290  PURTEST1        JMP       FLSHEND1            ; NEW EOF NOT SMALLER
000291  TRELEAS1        JMP       TRELEASE            ; OVERFLOW PREVENTER
000292  *
000293  PURGE           LDY       #FCBSTYP            ; FIND OUT WHAT TYPE OF TREE
000294                  LDA       (FCBPTR),Y          ; TO PERFORM THE PROPER
000295                  CMP       #SEEDTYP            ; STYLE OF BLOCK RELEASE
000296                  BEQ       EOFOUT              ; SEED DON'T DEALLOCATE
000297                  CMP       #TRETYP             ; FULL TREE?
000298                  BEQ       TRELEAS1            ; BRANCH IF YES
000299  *
000300  * IF WE GET HERE, WE ARE RELEASING
000301  * BLOCKS AT THE END OF A SAPLING FILE: CALCULATE CORRECT POSITION
000302  * WITHIN THE INDEX BLOCK AND ALLOW SUBROUTINE
000303  * PURGE LATTER BLOCKS TO DEALLOCATE
000304  * ALL THE DATA BLOCKS THAT FOLLOW
000305  *
000306                  JSR       FNDBMAP             ; REFRESH THE RIGHT MAP FOR THIS VOLUME
000307                  LDX       TPOSHI              ; PRELOAD
000308                  LDY       TPOSLH              ;  THE THREE EOF
000309                  LDA       TPOSLL              ;    BYTES
000310                  BNE       PUR1                ; BRANCH IF NO BOUNDARY ADJUSTMENT NEEDED
000311                  CPY       #0
000312                  BNE       PUR2                ; MIDDLE BYTE ZERO MEANS NO CARRY
000313                  CPX       #0                  ; ALL BYTES ZERO??
000314                  BEQ       PUR1                ; BRANCH IF YES
000315                  DEX
000316  *
000317  * THESE LINES IF CODE, SOMEWHAT CRYPTIC,
000318  * CALCULATE THE POINT AT WHICH THE
000319  * LAST BLOCK CONTAINING THE LAST BIT
```

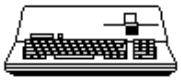


```
000320  * OF DATUM
000321  *
000322  * THE FOLLOWING IS ROUGHLY A /512
000323  * ALGORITHM
000324  *
000325  PUR2            DEY
000326  PUR1            TXA
000327                  LSR         A
000328                  TYA
000329                  ROR         A
000330  *
000331                  JSR         PURLBLKS            ; MAKES A GOOD PTR TO DO THE RELEASING
000332                  LDY         #FCBSTAT            ; MARK INDEX BLOCK
000333                  LDA         (FCBPTR),Y          ; AS DIRTY
000334                  ORA         #IDXMOD
000335                  STA         (FCBPTR),Y
000336                  LDA         PURUSE              ; INDICATE NEW NUMBER OF BLOCKS USED
000337                  CLC
000338                  ADC         #2                  ; ACCOUNT FOR CARDINAL AND INDEX
000339                  LDY         #FCBUSE
000340                  STA         (FCBPTR),Y          ; FILE LOW BYTE
000341                  INY
000342                  LDA         #0                  ; ANTICIPATE <257 BLOCKS
000343                  BCC         PURHI
000344                  LDA         #1                  ; >256 BLOCKS IN FILE
000345  PURHI           STA         (FCBPTR),Y          ; HIGH BYTE BLOCKS USED
000346  EOFOUT          CLC
000347                  RTS                             ; NO ERRORS POSSIBLE
000348  *
000349  PURLBLKS        EQU         *                   ; PURGE LATTER BLOCKS
000350  * INPUT ARG: A REGISTER CONTAINING
000351  * POINTER TO CURRENT DATA BLOCK WITHIN THE
000352  * CURRENT INDEX BLOCK (TINDX)
000353  * DEALLOCATE ALL LEGAL BLOCKS AFTER
000354  * THE A REGISTER PTR. NO ERRORS POSSIBLE
000355  *
000356                  TAY                             ; MAKE PROPER INDEX
000357                  STY         PURUSE              ; INDICATES NUMBER OF BLOCKS IN USE IN FILE
000358  PURLOOP         INY                             ; POINT TO A PTR TO DATA BLK TO DEALLOCATE
000359                  BEQ         PURLRTS             ; NO MORE BLOCKS IN INDEX
000360                  INC         TINDX+1             ; GET HIGH PART OF BLOCK ADDR
000361                  LDA         (TINDX),Y
000362                  TAX                             ; X IS A PASSING PARM
000363                  LDA         #0                  ; TELL INDEX BLOCK THAT THE DATA
000364                  STA         (TINDX),Y           ; BLOCK IS NOW FREE
000365                  TXA
000366                  DEC         TINDX+1             ; AND LOW PART
000367                  ORA         (TINDX),Y
000368                  BEQ         PURLOOP             ; INDICATED ADDR WAS ZERO-ZERO
000369                  LDA         (TINDX),Y           ; A REG IS ANOTHER PASSING PARM
000370                  PHA
000371                  LDA         #0
000372                  STA         (TINDX),Y           ; AND SET LOW DATA ADDR AS FREED
000373                  PLA
000374                  STY         PURPLACE            ; TEMP STORAGE
000375                  JSR         DEALLOC             ; DEALLOCATE BLOCK (ADDR: A (LOW), X ( HIGH)
000376                  LDY         #VCBTFRE
000377                  CLC
000378                  LDA         (VCBPTR),Y          ; ADJUST NUMBER OF FREE BLOCKS ON VOLUME
000379                  ADC         #1
000380                  STA         (VCBPTR),Y
000381                  INY
000382                  LDA         (VCBPTR),Y          ; HIGH BYTE OF TOTAL FREE
000383                  ADC         #0
000384                  STA         (VCBPTR),Y
000385                  LDY         PURPLACE
000386                  JMP         PURLOOP
000387  PURLRTS         RTS
000388  PURUSE          DS          1                   ; CURRENT NUMBER OF BLOCKS USED
000389  PURPLACE        DS          1                   ; CURRENT PLACE IN RELEASE-BLOCK CYCLE
000390  TRELEASE        EQU         *
000391                  JMP         EOFOUT              ; RELEASE TWO LEVEL TREE CODE GOES HERE
000392  *
000393  GETEOF          LDY         #FCBEOF             ; INDEX TO END OF FILE MARK
000394                  LDX         #0                  ; WE'VE GOT INDIRECT BOTH WAYS (IN & OUT)
000395  OUTEOF          LDA         (FCBPTR),Y
000396                  STA         (C.OUTEOF,X)
000397                  INY
000398                  CPY         #FCBEOF+3
000399                  BEQ         OFFRTS              ; BRANCH IF ALL THREE BYTES TRANSFERED.
000400                  INC         C.OUTEOF            ; BUMP USER'S POINTER.
```

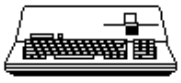


```
000401                  BNE         OUTEOF
000402                  INC         C.OUTEOF+1
000403                  BNE         OUTEOF              ; BRANCH ALWAYS
000404  *
000405                  CHN         DESTROY,4,2
000406
000407  ********************************************************************
000408  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: CLOSE.EOF
000409  ********************************************************************
000410
```

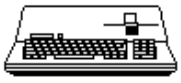
End of File -- Lines: 410 Characters: 16893



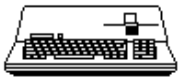
```
================================================================================================
FILE: "SOS.COMP.OPR.IPL.TEXT"
================================================================================================

000001  ****************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.OPR.IPL
000003  ****************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  :TABS 17,23,40
000007  ::PR#1,L58        132N
000008  SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
000009  SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
000010  SL4:DR1:A,6,1
000011  END
000012
000013  ****************************************************************************
000014  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.OPR.IPL
000015  ****************************************************************************

End of File -- Lines: 15 Characters: 579
```



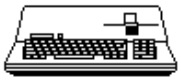
```
================================================================================
FILE: "SOS.COMP.SOS.NOLIST.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.SOS.NOLIST
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  :TABS 17,23,40
000007  SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
000008  SL4:DR1:ASM INIT.SRC,INIT.OBJ,6,1
000009  SL4:DR1:ASM SYSGLOB.SRC,SYSGLOB.OBJ,6,1
000010  SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
000011  SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
000012  SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
000013  SL4:DR1:ASM UMGR.SRC,UMGR.OBJ,6,1
000014  SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
000015  SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
000016  SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
000017  SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
000018  SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
000019  SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
000020  SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
000021  SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
000022  END
000023
000024  ************************************************************************
000025  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.SOS.NOLIST
000026  ************************************************************************
000027

End of File -- Lines: 27 Characters: 1028
```



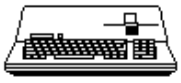
```
================================================================================
FILE: "SOS.COMPILE.BFM.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.BFM
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  :T 17,23,40
000007  ::PR#1,L58        132N
000008  ::SL4:DR1:ASM PRINT,BFM.OBJ,6,1
000009  ::END
000010
000011  ************************************************************************
000012  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.BFM
000013  ************************************************************************

End of File -- Lines: 13 Characters: 525
```



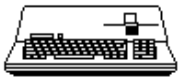
```
================================================================================
FILE: "SOS.COMPILE.SOS.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.SOS
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  :TABS 17,23,40
000007  ::PR#1,L58        132N
000008  SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
000009  SL4:DR1:ASM INIT.SRC,INIT.OBJ,6,1
000010  SL4:DR1:ASM SYSGLOB.SRC,SYSGLOB.OBJ,6,1
000011  SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
000012  SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
000013  SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
000014  SL4:DR2:ASM UMGR.SRC,UMGR.OBJ,6,1
000015  SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
000016  SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
000017  SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
000018  SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
000019  SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
000020  SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
000021  SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
000022  SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
000023  ::END
000024
000025  ************************************************************************
000026  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.SOS
000027  ************************************************************************
000028

End of File -- Lines: 28 Characters: 1039
```



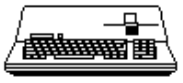
```
================================================================================================
FILE: "SOS.CREATE.TEXT"
================================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: CREATE
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007  CREATE          EQU     *
000008                  INC     CFLAG               ; SAY WE ARE IN CREATE (DIR EXTEND)
000009                  JSR     LOOKFILE            ; CHECK FOR DUPLICATE / GET FREE ENTRY
000010                  BCS     TSTFNF              ; ERROR CODE IN ACC MAY BE 'FILE NOT FOUND'
000011                  LDA     #DUPERR             ; TELL EM A FILE OF THAT NAME ALREADY EXISTS
000012  CRERR1          SEC                         ; INDICATE ERROR ENCOUNTERED
000013                  RTS                         ; RETURN ERROR IN ACC.
000014  *
000015  TSTFNF          CMP     #FNFERR             ; 'FILE NOT FOUND' IS WHAT WE WANT
000016                  BNE     CRERR1              ; PASS BACK OTHER ERROR.
000017                  LDA     NOFREE              ; TEST FOR DIRECTORY SPACE
000018                  BNE     CREAT1              ; BRANCH IF VALID FREE ENTRY WAS FOUND.
000019                  LDA     #DIRFULL            ; RETURN DIRECTORY FULL ERROR
000020                  SEC
000021                  RTS
000022  *
000023  CREAT1          LDY     #$9                 ; SET UP DEFAULT PARAMETERS FOR CREATE
000024                  LDA     #0                  ; IN THE SPACE DIRECTLY FOLLOWING THE
000025  ZERCALL         STA     C.FILID,Y           ; CALL SPECIFCATION AND THEN
000026                  DEY                         ; CHECK FOR ADDITIONAL PARAMETERS FROM
000027                  BPL     ZERCALL             ; USER'S CALL SPEC VIA 'C.CLIST'
000028                  LDA     #SEEDTYP            ; DEFAULT TYPE IS 'SEED' TREE INDEX
000029                  STA     C.STOR
000030                  LDY     C.XLEN              ; GET THE LENGTH OF THE CALL XTENSION LIST
000031                  BEQ     CRENAM              ; IF ZERO THEN USE DEFAULTS
000032                  DEY                         ; (SINCE THE POINTER IS AT BYTE 0)
000033                  CPY     #$9                 ; MAKE SURE WE DON'T HAVE TOO MANY PARAMETERS
000034                  BCC     MOVPARM             ; MOVE 'EM IF REASONABLE COUNT.
000035                  LDA     #BADLSTCNT          ; INVALID LIST COUNT
000036                  RTS                         ; RETURN ERROR.
000037  *
000038  MOVPARM         LDA     (C.XLIST),Y         ; MOVE IN THE USER SPECIFIED
000039                  STA     C.FILID,Y           ; PARAMETERS. VALIDITY IS CHECKED
000040                  DEY                         ; AT VARIOUS POINTS FURTHER ALONG IN
000041                  BPL     MOVPARM             ; THIS PROCESS.
000042  CRENAM          LDY     #0                  ; MOVE LOCAL FILE NAME TO ENTRY BUFFER.
000043                  LDA     (PATHNML),Y         ; GET LENGTH OF LOCAL NAME
000044                  TAY
000045  CRENAM1         LDA     (PATHNML),Y
000046                  STA     DFIL+D.STOR,Y
000047                  DEY                         ; (MOVE ALL, INCLUDING LENGTH BYTE.)
000048                  BPL     CRENAM1
000049                  LDA     C.FILID             ; MOVE FILE AND AUX ID.
000050                  STA     DFIL+D.FILID
000051                  LDA     C.AUXID
000052                  STA     DFIL+D.AUXID
000053                  LDA     C.AUXID+1
000054                  STA     DFIL+D.AUXID+1
000055                  LDA     #READEN+WRITEN+RENAMEN+DSTROYEN
000056                  STA     DFIL+D.ATTR
000057                  LDA     D.HEAD              ; SAVE FILE'S HEADER ADDRESS TOO.
000058                  STA     DFIL+D.DHDR
000059                  LDA     D.HEAD+1
000060                  STA     DFIL+D.DHDR+1
000061                  JSR     TWRPROT1            ; CAN WE WRITE TO THIS DISKETTE?
000062                  BCS     CRERR1
000063                  LDA     C.STOR              ; NOW TEST STORAGE TYPE FOR TREE TYPE FILES
000064                  CMP     #4                  ; NOTE: THIS IS HARD CODED SINCE ALL TREES ARE LESS THAN 4 ***********
000065                  BCC     SEED                ; BRANCH IF SOME TYPE OF TREE (SEED, SAPLING...)
000066                  JMP     NOTREE              ; GO TEST FOR SOME OTHER TYPE (SUCH AS DIRECTORY).
000067                  PAGE
000068  *
000069  SEED            LDX     #SEEDTYP            ; START OUT ASSUMING A SEED FILE
000070                  LDA     C.EOFHH             ; TEST FOR OUT OF RANGE PREALLOCATION
000071                  BEQ     SEED1               ; (HOPEFULLY BRANCH ALWAYS)
000072  OVFLOW          LDA     #OVRERR             ; REPORT UNABLE TO SATISFY REQUEST.
000073                  SEC                         ; INDICATE ERROR
000074                  RTS
000075  *
000076  SEED1           LDA     C.EOFHL             ; CALCULATE THE NUMBER OF
```



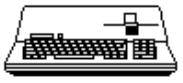
```
000077                STA       DFIL+D.EOF+2        ; BLOCKS NEEDED FOR PRE-ALLOCATION
000078                LSR       A
000079                TAY                           ; Y HOLDS THE NUMBER OF INDEX BLOCKS NEEDED
000080                STA       DATBLKH
000081                LDA       C.EOFLH             ; (CARRY UNDISTURBED FROM LAST SHIFT)
000082                STA       DFIL+D.EOF+1
000083                ROR       A                   ; WE NOW HAVE THE LOW ORDER COUNT OF NEEDED DATA BLOCKS
000084                STA       DATBLKL
000085                LDA       C.EOFLL
000086                STA       DFIL+D.EOF          ; (CARRY IN TACT FROM LOW COUNT)
000087                BNE       INCDATA             ; BUMP THE COUNT ON DATA BLOCKS IF REQUEST
000088                BCC       TSTSAP              ; IS NOT A MULTIPLE OF 512.
000089  INCDATA       INC       DATBLKL
000090                BNE       TSTSAP
000091                INY                           ; MUST INCREASE NUMBER OF INDEXES ALSO.
000092                INC       DATBLKH
000093  TSTSAP        TYA                           ; IF NON ZERO, THEN IT'S AT LEAST A SAPLING.
000094                BNE       SAPLING
000095                LDA       DATBLKL             ; TO QUALIFY AS AN HONEST SEED,
000096                BNE       TSTSEED             ; THEN ONE OR LESS DATA BLOCKS REQUESTED
000097                INC       DATBLKL             ; (MUST BE AT LEAST ONE BLOCK ALLOCATED
000098                BNE       CREALC              ; TYPE IS SEED. BRANCH ALWAYS
000099  TSTSEED       CMP       #1                  ; IF GREATER THAN ONE, IT'S NOT A SEED.
000100                BEQ       CREALC              ; IT IS A SEED. CONTINUE CREATION
000101                INX                           ; THE TYPE IS SAPLING.
000102                INY                           ; ONE INDEX BLOCK IS NEEDED.
000103                BNE       CREALC              ; BRANCH ALWAYS
000104                PAGE
000105  *
000106  SAPLING       INX                           ; TYPE IS AT LEAST SAPLING.
000107                CMP       #1                  ; NO MORE THAN ONE INDEX BLOCK FOR A SAPLING
000108                BNE       TREE
000109                LDA       DATBLKL             ; MUST BE SURE THIS IS REAL MAX SAPLING (128K FILE)
000110                BEQ       CREALC              ; BRANCH IF IT IS.
000111  TREE          INY                           ; ACCOUNT FOR ADDITIONAL 2ND LEVEL INDEX
000112  *
000113                INX                           ; TYPE IS TREE (2 LEVEL INDEX)
000114                INY                           ; ADD AN EXTRA INDEX BLOCK FOR TOP INDEX
000115  CREALC        STY       INDXBLK             ; STORE INDEX BLOCK COUNT
000116                TXA                           ; PUT STORAGE TYPE IN DIRECTORY ENTRY
000117                ASL       A
000118                ASL       A
000119                ASL       A
000120                ASL       A
000121                ORA       DFIL+D.STOR
000122                STA       DFIL+D.STOR
000123                STX       LEVELS              ; SAVE NUMBER OF INDEX LEVELS FOR PREALLOCATION.
000124                TYA                           ; NOW FIGURE THE TOTAL NUMBER OF
000125                CLC                           ; BLOCKS NEEDED (DATA + INDEX BLOCKS)
000126                ADC       DATBLKL
000127                STA       DFIL+D.USAGE        ; (MIGHT AS WELL RECORD IT IN DIR
000128                STA       REQL                ; WHILE WE'RE AT IT.)
000129                LDA       DATBLKH
000130                ADC       #0                  ; UPDATE HI BYTE TOO
000131                STA       DFIL+D.USAGE+1
000132                STA       REQH
000133                LDX       D.DEV               ; PASS ALONG THE DEVICE WE'RE TALKIN ABOUT.
000134                JSR       TSFRBLK             ; 'TEST FREE BLOCKS' FINDS OUT IF ENOUGH FREE SPACE EXISTS
000135                BCS       OVFLOW              ; BRANCH IF NOT ENOUGH SPACE.
000136                JSR       ALC1BLK             ; GO ALLOCATE FIRST BLOCK
000137                BCS       CRERR
000138                STA       DFIL+D.FRST         ; (RETURNS ACC=LOW Y=HIGH)
000139                STA       IDXADRL             ; SAVE AS ADDRESS FOR INCORE INDEX ALSO.
000140                STY       DFIL+D.FRST+1
000141                STY       IDXADRH
000142                JSR       ZERGBUF             ; GO CLEAN OUT GBUF
000143                JSR       GTTINDX             ; GET TEMPORARY SPACE FOR AN INDEX BLOCK
000144                JSR       ZTMPIDX             ; AND ZERO IT OUT.
000145                LDX       LEVELS
000146                DEX                           ; TEST FOR NUMBER OF LEVELS NEEDED.
000147                BEQ       ENDCRE              ; BRANCH IF SEED FILE.
000148                DEX                           ; IS IT A SAPLING PRE-ALLOCATION.
000149                BEQ       SAPFILE
000150                LDY       INDXBLK             ; LOAD NUMBER OF INDEX BLOCKS NEEDED
000151                DEY                           ; REMOVE THE ONE JUST ALLOCATED.
000152                STY       REQL
000153                STY       INDXBLK
000154                JSR       ALCIDXS             ; GO ALLOCATE INDEXES FOR LOWER INDEX BLOCKS.
000155                BCS       CRERR
000156                JSR       WRTDFRST            ; GO WRITE TREE TOP INDEX BLOCK.
000157                BCS       CRERR               ; BRANCH IF UNABLE TO DO THIS.
```



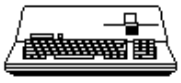
```
000158                  LDA     #0                  ; INIT INDEX POINTER
000159                  STA     TREPTR
000160                  PAGE
000161  FILLTREE        LDY     TREPTR
000162                  LDA     (TINDX),Y           ; GET ADDRESS OF LOWER BLOCK
000163                  STA     IDXADRL
000164                  INC     TINDX+1             ; BUMP TO PAGE 2 TO GET HI ADDRESS.
000165                  LDA     (TINDX),Y           ; GET HIGH ADDRESS.
000166                  STA     IDXADRH
000167                  DEC     TINDX+1             ; CLEAN UP AFTER SELF...
000168                  DEC     INDXBLK             ; IS THIS THE LAST BLOCK ALLOCATED?
000169                  BEQ     LSTSAP              ; YES, ALLOCATE PARTIAL FILLED INDEX BLOCK
000170                  LDA     #0                  ; ALLOCATE ALL 256 INDEXES
000171                  STA     REQL
000172                  JSR     SAPINDX             ; AND WRITE ZEROED DATA BLOCKS.
000173                  BCS     CRERR               ; STOP IF ERROR ENCOUNTERED.
000174                  JSR     WRTINDX             ; WRITE INDEX BLOCK
000175                  BCS     CRERR               ; HOPEFULLY NEVER TAKEN.
000176                  INC     TREPTR
000177                  JSR     RDFRST              ; READ IN TOP INDEX AGAIN.
000178                  BCC     FILLTREE            ; BRANCH IF NO ERROR.
000179  CRERR           SEC                         ; JUST IN CASE IT WAS CLEAR.
000180                  RTS                         ; RETURN ERROR.
000181  *
000182  *
000183  SAPFILE         EQU     *
000184  LSTSAP          LDA     DATBLKL             ; GET NUMBER OF DATA BLOCKS (LOW BYTE) REQUESTED.
000185                  STA     REQL
000186                  JSR     SAPINDX             ; GO ALLOCATE DATA BLOCKS AND WRITE EM.
000187                  BCS     CRERR
000188  ENDCRE          JSR     WRTINDX             ; GO WRITE INDEX BLOCK. (FOR SEED THIS IS DATA.)
000189                  BCS     CRERR
000190                  LDX     #3                  ; MOVE CREATION TIME FOR THIS ENTRY
000191  TRETIME         LDA     DATELO,X
000192                  STA     DFIL+D.CREDT,X
000193                  DEX
000194                  BPL     TRETIME
000195  ENDCRE0         INC     H.FCNT              ; ADD ONE TO TOTAL NUMBER OF FILES IN SPECIFIED DIRECTORY.
000196                  BNE     ENDCRE1
000197                  INC     H.FCNT+1
000198                  LDX     #3                  ; ENSURE MOD
000199  ENDCRX          LDA     DATELO,X            ; DATE/TIME
000200                  STA     DFIL+D.MODDT,X      ; IS
000201                  DEX                         ; INITIALIZED
000202                  BPL     ENDCRX
000203  ENDCRE1         LDX     D.DEV               ; UPDATE APPROPRIATE BIT MAP
000204                  JSR     UPBMAP
000205                  BCS     CRERR2              ; BRANCH ON BITMAP UPDATE ERR
000206                  JSR     DREVISE             ; UPDATE DIRECTORY LAST
000207                  RTS                         ; RETURN ERRORS OR OK RESULT
000208  *
000209                  PAGE
000210  SAPINDX         JSR     ZTMPIDX             ; ZERO OUT ANY STUFF LEFT OVER.
000211                  LDA     REQL                ; PRESERVE REQUEST COUNT
000212                  STA     TLINK
000213                  JSR     ALCIDXS             ; GO ALLOCATE REQUESTED NUMBER OF BLOCKS.
000214                  BCS     CRERR
000215                  LDY     #0                  ; THEN WRITE ZEROS TO DATA BLOCKS.
000216                  STY     SAPTR               ; USE AS POINTER TO INDEX BLOCK
000217                  LDA     (TINDX),Y           ; GET DATA BLOCK ADDRESS (LOW BYTE).
000218                  STA     BLOKNML
000219                  INC     TINDX+1
000220                  LDA     (TINDX),Y           ; GET HIGH ADRRESS OF PRE-ALLOCATED DATA BLOCK.
000221                  STA     BLOKNMH
000222                  DEC     TINDX+1             ; (RESET BUFFER ADDRESS)
000223                  JSR     WRTGBUF             ; WRITE DATA BLOCK
000224                  BCS     CRERR
000225                  LDA     TLINK               ; GET NUMBER REQUESTED AGAIN
000226                  STA     REQL
000227  DATINIT         LDY     SAPTR               ; GET POINTER TO INDEX BLOCK AGAIN.
000228                  INY                         ; ANTICIPATE DOIN' THE NEXT DATA BLOCK
000229                  DEC     REQL                ; DO WE INDEED HAVE ANOTHER BLOCK TO WRITE.
000230                  BEQ     DATDONE             ; NO, ALL DONE (CARRY CLEAR).
000231                  STY     SAPTR               ; USE AS POINTER TO INDEX BLOCK
000232                  LDA     (TINDX),Y           ; GET DATA BLOCK ADDRESS (LOW BYTE).
000233                  STA     BLOKNML
000234                  INC     TINDX+1             ; BUMP HI ADDR OF INDEX BUFFER TO ACCESS HIGH ADDR.
000235                  TAX                         ; WAS LOW ADDRESS A ZERO?
000236                  BNE     DATIT1              ; IF NOT, NO NEED TO CHECK VALIDITH OF HI BYTE
000237                  CMP     (TINDX),Y
000238                  BNE     DATIT1              ; BOTH BYTES CAN'T BE ZERO.
```



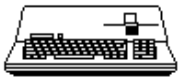
```
000239                  LDA     #ALCERR
000240                  JSR     SYSDEATH
000241  DATIT1          LDA     (TINDX),Y       ; GET HIGH ADRRESS OF PRE-ALLOCATED DATA BLOCK.
000242                  STA     BLOKNMH
000243                  DEC     TINDX+1         ; (RESET BUFFER ADDRESS)
000244                  LDA     #GBUF/256
000245                  STA     DBUFPH          ; RESET TO ADDR TO GBUF JUST TO BE SURE.
000246                  JSR     REPEATIO        ; WRITE DATA BLOCK
000247                  BCC     DATINIT
000248  DATDONE         RTS                     ; RETURN STATUS (CARRY SET IF ERROR)
000249  *
000250  REPEATIO        EQU     *
000251                  LDA     #RPTCMD
000252                  STA     DHPCMD
000253                  JMP     RPEATIO1
000254  *
000255  ZERGBUF         LDY     #0              ; ZERO OUT THE GENERAL PURPOSE BUFFER
000256                  TYA
000257  ZGBUF           STA     GBUF,Y          ; WIPE OUT BOTH PAGES
000258                  STA     GBUF+$100,Y     ; WITH SAME LOOP.
000259                  INY
000260                  BNE     ZGBUF
000261                  RTS
000262  *
000263  *
000264  ZTMPIDX         LDY     #0              ; ZERO OUT TEMPORARY INDEX BLOCK
000265                  TYA
000266  ZINDX1          STA     (TINDX),Y       ; THIS HAS TO BE DONE A
000267                  INY                     ; TIME SINCE IT'S INDIRECT.
000268                  BNE     ZINDX1
000269                  INC     TINDX+1
000270  ZINDX2          STA     (TINDX),Y
000271                  INY
000272                  BNE     ZINDX2
000273                  DEC     TINDX+1         ; RESTORE PROPER ADDRESS
000274  CRERR2          RTS
000275                  PAGE
000276  NOTREE          CMP     #DIRTYP         ; IS A DIRECTORY TO BE CREATED?
000277                  BEQ     ISDIR           ; YES, DO SO...
000278                  JMP     NOTDIR          ; NO, TRY NEXT TYPE.
000279  *
000280  ISDIR           LDA     C.EOFHH         ; CAN'T CREATE A DIRECTORY LARGER THAN
000281                  ORA     C.EOFHL         ; 127 BLOCKS (THAT'S HUGE!)
000282                  BEQ     ISDIR1          ; BRANCH IF WITHIN LIMITS, OTHEWISE
000283  DIROVR          LDA     #OVRERR         ; REQUESTED DIRECTORY SIZE CAN'T BE
000284                  SEC                     ; CREATED. SET CARRY TO INDICATE ERROR.
000285                  RTS
000286  *
000287  ISDIR1          LDA     C.EOFLH         ; CALCULATE HOW MANY BLOCKS WILL
000288                  LSR     A               ; BE NEEDED FOR THIS NEW DIRECTORY.
000289                  TAY                     ; (SAVE INITIAL COUNT IN Y)
000290                  LDA     C.EOFLL         ; IF REQUESTED EOF IS NOT AN EVEN BLOCK
000291                  BNE     DADD1           ; SIZE, THEN ROUND UP.
000292                  BCC     TSDIRSZ         ; BRANCH IF ROUNING UNNECESSARY.
000293  DADD1           INY                     ; ADD ONE TO BLOCK COUNT.
000294  TSDIRSZ         TYA                     ; TEST TO BE SURE SIZE IS GREATER THAN ZERO
000295                  BEQ     DADD1           ; IF ZERO THEN SIZE=1
000296                  STA     DFIL+D.USAGE    ; SAVE NUMBER OF BLOCKS TO BE USED.
000297                  STA     REQL
000298                  ASL     A               ; NOW SAVE ADJUSTED END OF FILE
000299                  STA     DFIL+D.EOF+1
000300                  LDA     #0
000301                  STA     DFIL+D.EOF
000302                  STA     DFIL+D.EOF+2
000303                  STA     REQH            ; REQUESTED NUMBER OF BLOCKS NEVER EXCEEDS 128.
000304                  JSR     TSFRBLK         ; TEST TO BE SURE ENOUGH DISK SPACE IS FREE.
000305                  BCS     DIROVR          ; BRANCH IF REQUEST TOO LARGE.
000306                  JSR     ZERGBUF         ; CLEAR CRAP FROM GBUF.
000307                  JSR     ALC1BLK         ; GET ADDRESS OF FIRST (HEADER) BLOCK.
000308                  BCS     CRERR2
000309                  STA     DFIL+D.FRST
000310                  STA     TLINK
000311                  STY     DFIL+D.FRST+1
000312                  STY     TLINK+1         ; (TLINK IS FOR REVERSE LINKAGE.)
000313                  LDA     SOSTMPL         ; STORE SOS STAMP IN NEW DIRECTORY
000314                  STA     GBUF
000315                  LDA     SOSTMPH
000316                  STA     GBUF+1
000317                  LDY     #4              ; MOVE OTHER VARIOUS THINGS
000318                  BNE     DRSTUF1         ; BRANCH ALWAYS
000319  DRSTUF          LDA     D.ENTBLK,Y      ; MOVE OWNING ENTRY'S
```



```
000320                  STA     GBUF+HRBLK+4,Y      ; BLOCK ADDRESSES AND NUMBER TO NEW HEADER.
000321  DRSTUF1         LDA     SOSVER,Y            ; MOVE VERSION, COMPATABLITY,
000322                  STA     GBUF+HVER+4,Y       ; ATTRIBUTES, AND ENTRY SIZE
000323                  DEY
000324                  BPL     DRSTUF
000325                  LDA     H.ENTLN             ; OVER WRITE LAST BYTE MOVED IN ABOVE LOOP WITH
000326                  STA     GBUF+HRELN+4        ; THE PARENT DIRECTORY ENTRY LENGTH.
000327                  LDA     DFIL+D.STOR         ; SET HEADER TYPE AND NAME
000328                  TAY
000329                  ORA     #HEDTYP*16
000330                  STA     GBUF+HNLEN+4
000331                  TYA                         ; (AND WHILE WE'RE AT IT SET DIRECTORY TYPE)
000332                  ORA     #DIRTYP*16
000333                  STA     DFIL+D.STOR
000334  *
000335  MVHNAME         LDA     DFIL+D.STOR,Y
000336                  STA     GBUF+HNLEN+4,Y      ; MOVE HEADER NAME
000337                  DEY
000338                  BNE     MVHNAME
000339                  LDX     #3                  ; GET CURRENT DATE.
000340  CRETIME         LDA     DATELO,X
000341                  STA     GBUF+HCRDT+4,X      ; SAVE AS HEADER CREATION TIME
000342                  STA     DFIL+D.CREDT,X      ; AND DATE OF FILE CREATE.
000343                  DEX
000344                  BPL     CRETIME
000345                  LDA     #$76
000346                  STA     GBUF+HPENAB+4       ; DUMMY PASSWORD
000347                  DEC     REQL                ; TEST FOR ONE BLOCK DIRECTORY
000348                  BEQ     DIRCREND            ; IT IS, FINISH UP.
000349                  JSR     DIRWRT              ; GO WRITE FIRST DIRECTORY BLOCK AND ALLOCATE NEXT
000350                  BCS     DERROR              ; PASS BACK ERROR.
000351                  JSR     ZERGBUF             ; CLEAN OUT GENERAL BUFFER AGAIN.
000352  CRNXTDIR        LDA     TLINK               ; MOVE LAST BLOCK ADDRESS
000353                  STA     GBUF                ; AS BACKWARD LINK.
000354                  LDA     TLINK+1
000355                  STA     GBUF+1
000356                  LDA     FLINK               ; MAKE FORWARD LINK INTO CURRENT ADDRESS
000357                  STA     TLINK
000358                  LDA     FLINK+1
000359                  STA     TLINK+1
000360                  DEC     REQL                ; IS THIS THE LAST BLOCK?
000361                  BEQ     DIRCREND
000362                  JSR     DIRWRT              ; WRITE THIS BLOCK AND ALLOCATE NEXT.
000363                  BCS     DERROR
000364                  LDA     #0                  ; ZERO OUT FORWARD LINK
000365                  STA     GBUF+2
000366                  STA     GBUF+3
000367                  BEQ     CRNXTDIR            ; BRANCH ALWAYS
000368  *
000369  DIRCREND        JSR     DIRWRT1             ; WRITE LAST BLOCK OF THIS DIRECTORY
000370                  BCS     DERROR
000371                  JMP     ENDCRE0             ; FINISH UP WRITING OWNER DIRECTORY STUFF.
000372  *
000373  DIRWRT          JSR     ALC1BLK             ; GET ADDRESS OF NEXT BLOCK.
000374                  BCS     DERROR
000375                  STA     GBUF+2
000376                  STY     GBUF+3              ; SAVE LINK ADDRESS
000377                  STA     FLINK
000378                  STY     FLINK+1
000379  DIRWRT1         LDA     TLINK               ; GET ADDRESS OF CURRENT BLOCK
000380                  STA     BLOKNML
000381                  LDA     TLINK+1
000382                  STA     BLOKNMH
000383                  JMP     WRTGBUF             ; GO WRITE IT OUT
000384                  PAGE
000385  *
000386  ERRGBUF         EQU     *
000387  DERROR          RTS
000388  *
000389  *
000390  SOSTMPL         DFB     $0                  ; THE FOLLOWING TWO BYTES ARE THE 'SOS STAMP'
000391  SOSTMPH         DFB     $0
000392  *
000393  SOSVER          DFB     0,0,0,$27,13
000394  *
000395  *
000396  RNDTAB          EQU     *
000397  ENTCALC         LDA     #GBUF/256           ; SET HIGH ADDRESS OF DIRECTORY ENTRY INDEX POINTER
000398                  STA     DRBUFPH
000399                  LDA     #4                  ; CALCULATE ADDRESS OF ENTRY BASED
000400                  LDX     D.ENTNUM            ; ON THE ENTRY NUMBER
```

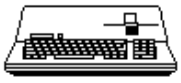


```
000401  ECALC0          CLC
000402  ECALC1          DEX                             ; ADDR=GBUF+((ENTNUM-1)*ENTLEN)
000403                  BEQ         ECALC2
000404                  ADC         H.ENTLN
000405                  BCC         ECALC1
000406                  INC         DRBUFPH             ; BUMP HI ADDRESS
000407                  BCS         ECALC0              ; BRANCH ALWAYS.
000408  *
000409  ECALC2          STA         DRBUFPL             ; SAVE NEWLY CALCULATED LOW ADDRESS
000410                  RTS
000411                  PAGE
000412  DERROR2         RTS
000413  *
000414  DREVISE         LDA         DATELO              ; IF NO CLOCK,
000415                  BEQ         DREVISE1            ; THEN DON'T TOUCH MOD T/D
000416                  LDX         #3                  ; MOVE LAST MODIFICATION DATE/TIME TO ENTRY BEING UPDATED.
000417  MODTIME         LDA         DATELO,X
000418                  STA         DFIL+D.MODDT,X
000419                  DEX
000420                  BPL         MODTIME
000421  *
000422  DREVISE1        LDA         DFIL+D.ATTR         ; MARK ENTRY AS BACKUPABLE
000423                  ORA         BKBITFLG            ; BIT 5 = BACKUP NEEDED BIT
000424                  STA         DFIL+D.ATTR
000425                  LDA         D.DEV               ; GET DEVICE NUMBER OF DIRECTORY
000426                  STA         DEVNUM              ; TO BE REVISED.
000427                  LDA         D.ENTBLK            ; AND ADDRESS OF DIRECTORY BLOCK
000428                  STA         BLOKNML             ; THAT CONTAINS THE ENTRY.
000429                  LDA         D.ENTBLK+1
000430                  STA         BLOKNMH
000431                  JSR         RDGBUF              ; READ BLOCK INTO GENERAL PURPOSE BUFFER.
000432                  BCS         ERRGBUF
000433                  JSR         ENTCALC             ; FIX UP POINTER TO ENTRY LOCATION WITHIN GBUF.
000434                  LDY         H.ENTLN             ; NOW MOVE 'D.' STUFF TO DIRECTORY.
000435                  DEY
000436  MVDENT          LDA         DFIL+D.STOR,Y
000437                  STA         (DRBUFPL),Y
000438                  DEY
000439                  BPL         MVDENT
000440                  LDA         D.HEAD              ; IS THE ENTRY BLOCK THE SAME AS THE
000441                  CMP         BLOKNML             ; ENTRY'S HEADER BLOCK?
000442                  BNE         SVENTDIR            ; NO, SAVE ENTRY BLOCK
000443                  LDA         D.HEAD+1            ; MAYBE, TEST HIGH ADDRESSES
000444                  CMP         BLOKNMH
000445                  BEQ         UPHEAD              ; BRANCH IF THEY ARE THE SAME BLOCK.
000446  SVENTDIR        JSR         WRTGBUF             ; WRITE UPDATED DIRECTORY BLOCK
000447                  BCS         DERROR2             ; RETURN ANY ERROR.
000448                  LDA         D.HEAD              ; GET ADDRESS OF HEADER BLOCK
000449                  STA         BLOKNML
000450                  LDA         D.HEAD+1
000451                  STA         BLOKNMH
000452                  JSR         RDGBUF              ; READ IN HEADER BLOCK FOR MODIFICATION
000453                  BCS         DERROR2
000454  UPHEAD          LDY         #1                  ; UPDATE CURRENT NUMBER OF FILES IN THIS DIRECTORY
000455  UPHED1          LDA         H.FCNT,Y
000456                  STA         GBUF+HCENT+4,Y      ; (CURRENT ENTRY COUNT)
000457                  DEY
000458                  BPL         UPHED1
000459                  LDA         H.ATTR              ; ALSO UPDATE HEADER'S ATTRIBUTES.
000460                  STA         GBUF+HATTR+4
000461                  JSR         WRTGBUF             ; GO WRITE UPDATED HEADER
000462  DERROR1         RTS                             ; IMPLICITLY RETURN ANY ERRORS
000463  *
000464                  PAGE
000465  *
000466  NOTDIR          LDA         #TYPERR             ; NOT TREE OR DIRECTORY- NOT A RECOGNIZED TYPE!
000467  TSTERR          SEC
000468                  RTS                             ; DO NOTHING.
000469  *
000470  *
000471  TSTSOS          LDA         GBUF                ; TEST SOS STAMP
000472                  CMP         SOSTMPL
000473                  BNE         TSTERR
000474                  LDA         GBUF+1
000475                  CMP         SOSTMPH
000476                  BNE         TSTERR
000477                  LDA         GBUF+4              ; TEST FOR HEADER
000478                  AND         #$E0
000479                  CMP         #HEDTYP*16
000480                  BNE         TSTERR              ; BRANCH IF NOT SOS HEADER (NO ERROR NUMBER)
000481                  CLC                             ; INDICATE NO ERROR
```

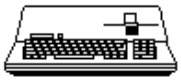


```
000482                  RTS
000483  *
000484                  CHN         FNDFIL,4,1
000485  NE              TSTERR
000486                  LDA         GBUF+4              ; TEST FOR HEADER
000487                  AND         #$E0
000488                  CMP         #HEDTYP*16
000489                  BNE         TSTERR              ; BRANCH IF NOT SOS HEADER (NO ERROR NUMBER)
000490                  CLC                             ; INDICATE NO ERROR
000491                  RTS
000492  *
000493                  CHN         FNDFIL,4,1
000494  O               ERROR
000495                  RTS
000496  *
000497                  CHN         FNDFIL,4,1
000498                  ENTRY       TOO.
000499                  LDY         #D.MODDT+3
000500  RIPTIME         LDA         DATELO,X
000501                  STA         (DRBUFPL),Y
000502                  DEY
000503                  DEX
000504                  BPL         RIPTIME             ;MOVE ALL FOR BYTES...
000505  RUPDATE         JSR         WRTGBUF             ;WRITE UPDATED ENTRY BACK TO DISK. (ASSUMES BLOKNM UNDISTURBEDD)
000506                  BCS         DERROR1             ;GIVE UP ON ANY ERROR.
000507                  LDY         #D.DHDR             ;NOW COMPARE CURRENT BLOCK NUMBER TO THIS
000508                  LDA         (DRBUFPL),Y         ; ENTRY'S HEADER BLOCK
000509                  INY
000510                  CMP         BLOKNML             ;ARE LOW ADDRESSES THE SAME?
000511                  STA         BLOKNML             ;(SAVE IT IN CASE IT'S NOT)
000512                  BNE         RIPPLE2             ;BRANCH IF ENTRY DOES NOT RESIDE IN SAME BLOCK AS HEADER.
000513                  LDA         (DRBUFPL),Y         ;CHECK HIGH ADDRESS JUST TO BE SURE.
000514                  CMP         BLOKNMH
000515                  BEQ         RIPPLE              ;THEY ARE THE SAME, CONTINUE RIPPLE TO ROOT DIRECTORY.
000516  RIPPLE2         LDA         (DRBUFPL),Y         ;THEY AREN'T THE SAME, READ IN THIS DIRECTORY'S HEADER.
000517                  STA         BLOKNMH
000518                  JSR         RDGBUF
000519                  BCC         RIPPLE              ;CONTINUE IF READ WAS GOOD.
000520  DERROR1         EQU         *
000521                  RTS
000522                  PAGE
000523  *
000524  NOTDIR          LDA         #TYPERR             ;NOT TREE OR DIRECTORY- NOT A RECOGNIZED TYPE!
000525  TSTERR          SEC
000526                  RTS                             ;DO NOTHING.
000527  *
000528  *
000529  TSTSOS          LDA         GBUF                ;TEST SOS STAMP
000530                  CMP         SOSTMPL
000531                  BNE         TSTERR
000532                  LDA         GBUF+1
000533                  CMP         SOSTMPH
000534                  BNE         TSTERR
000535                  LDA         GBUF+4              ;TEST FOR HEADER
000536                  AND         #$E0
000537                  CMP         #HEDTYP*16
000538                  BNE         TSTERR              ;BRANCH IF NOT SOS HEADER (NO ERROR NUMBER)
000539  DRVISDNE        CLC                             ;INDICATE NO ERROR./
000540                  RTS
000541  *
000542                  CHN         FNDFIL,4,1
000543
000544
000545  ************************************************************************
000546  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: CREATE
000547  ************************************************************************
```

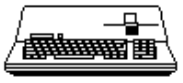
End of File -- Lines: 547 Characters: 22694



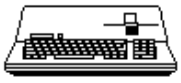
```
================================================================================================
FILE: "SOS.DESTROY.TEXT"
================================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: DESTROY
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007  *
000008  NEWLINE         LDY         #FCBATTR            ; ADJUST NEWLINE STATUS FOR OPEN FILE.
000009                  LDA         C.ISNEWL            ; ON OR OFF?
000010                  BPL         OFFNEWL             ; BRANCH IF NEW LINE IS TO BE CLEARED.
000011                  LDA         #NLINEN
000012                  ORA         (FCBPTR),Y          ; SET NEW LINE BIT IN ATTRIBUTES
000013                  STA         (FCBPTR),Y
000014                  LDY         #FCBNEWL            ; AND MOVE IN NEW 'NEW-LINE' BYTE.
000015                  LDA         C.NEWL
000016                  STA         (FCBPTR),Y
000017                  CLC
000018                  RTS                             ; NO ERROR POSSIBLE.
000019  *
000020  OFFNEWL         LDA         #$FF-NLINEN
000021                  AND         (FCBPTR),Y
000022                  STA         (FCBPTR),Y          ; CLEAR NEW-LINE BIT.
000023  OFFRTS          CLC                             ; THE NEW LINE CHARACTER DOES'T MATTER...
000024                  RTS
000025                  PAGE
000026  *
000027  GETINFO         JSR         FINDFILE            ; LOOK FOR FILE THEY WANT OT KNOW ABOUT.
000028                  BCC         GTINFO1             ; BRANCH IF NO ERRORS.
000029                  CMP         #BADPATH            ; WAS IT A ROOT DIRECTORY FILE?
000030                  SEC                             ; (IN CASE OF NO MATCH)
000031                  BNE         GINFOERR
000032                  LDA         #$F0
000033                  STA         DFIL+D.STOR         ; FOR GET INFO, REPORT PROPER STORAGE TYPE
000034                  LDA         #0                  ; FORCE A COUNT OF FREE BLOCKS.
000035                  STA         REQL
000036                  STA         REQH
000037                  JSR         TSFRBLK             ; (RETURNS IF IMMEDIATELY IF COUNT HAS PREVIOUSLY BEEN TAKEN)
000038                  LDY         #VCBTFRE+1
000039                  LDA         (VCBPTR),Y          ; RETURN TOTAL BLOCKS AND TOTAL IN USE.
000040                  STA         REQH                ; FIRST TRANSFER 'FREE' BLOCKS TO ZPAGE FOR LATER SUBTRACT
000041                  DEY
000042                  LDA         (VCBPTR),Y          ; TO DETERMINE THE 'USED' COUNT
000043                  STA         REQL
000044                  DEY
000045                  LDA         (VCBPTR),Y          ; TRANSFER TO 'D.' TABLE AS AUX I.D.
000046                  STA         DFIL+D.AUXID+1      ; (TOTAL BLOCK COUNT IS CONSIDERED AUX I.D. FOR THE VOLUME)
000047                  TAX
000048                  DEY
000049                  LDA         (VCBPTR),Y
000050                  STA         DFIL+D.AUXID
000051                  SEC                             ; NOW SUBTRACT AND REPORT THE NUMBER OF BLOCKS 'IN USE'
000052                  SBC         REQL
000053                  STA         DFIL+D.USAGE
000054                  TXA
000055                  SBC         REQH
000056                  STA         DFIL+D.USAGE+1
000057  GTINFO1         LDY         #0                  ; TRANSFER BYTES FROM THERE INTERNAL ORDER TO CALL SPEC VIA 'INFTABL'
TRANSLATION
000058  GTINFO2         LDA         INFTABL,Y
000059                  BPL         GTINFO3             ; BRANCH IF THIS IS DATA IS VALID AS IS.
000060                  AND         #$7F                ; IS THIS THE 4TH BYTE OF THE EOF PARAMETER?
000061                  BEQ         GTINFO4             ; YES, AND IT'S ALWAYS A ZERO.
000062                  CMP         #D.STOR+1           ; IS THIS THE STORAGE TYPE BYTE?
000063                  BNE         GINFOEND            ; NO, IT'S THE END OF INFO THAT CAN BE RETURNED.
000064                  LDA         DFIL+D.STOR         ; GET STORAGE TYPE
000065                  LSR         A
000066                  LSR         A
000067                  LSR         A
000068                  LSR         A                   ; MAKE IT A VALUE 1-$F BY SHIFTING OUT FILE NAME LENGTH.
000069                  BPL         GTINFO4             ; BRANCH ALWAYS
000070  *
000071  GTINFO3         TAX                             ; USE AS OFFSET INTO 'D.' TABLE.
000072                  LDA         DFIL,X
000073  GTINFO4         STA         (C.FILIST),Y        ; PASS TO USER'S BUFFER
000074                  INY
000075                  CPY         C.FILSTLN           ; HAS REQUEST BEEN FILLED?
```



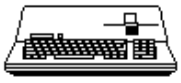
```
000076                  BNE         GTINFO2             ; NO, PASS NEXT
000077  GINFOEND        CLC                             ; INDICATE NO ERRORS
000078  GINFOERR        RTS
000079  *
000080  *
000081                  PAGE
000082  *
000083  SETINFO         JSR         FINDFILE            ; FIND WHAT USER WANTS...
000084                  BCS         SINFOERR            ; RETURN ANY FAILURE.
000085                  LDA         C.FILSTLN           ; TEST FOR NUL CHANGE
000086                  BEQ         SINFEND             ; BRANCH IF NOTHING TO CHANGE.
000087                  LDY         #0                  ; INIT POINTER TO USER SUPPLIED LIST.
000088                  LDA         (C.FILIST),Y        ; FETCH FILE ATTRIBUTES
000089                  AND         #$1C                ; FORBIDDEN BITS? <SRS 82.162>
000090                  BEQ         SETINF1             ; NO
000091                  LDA         #ACCSERR            ; YES
000092                  SEC
000093                  RTS                             ; RETURN AN ERROR
000094  SETINF1         LDA         BACKMASK            ; GET CURRENT BACKMASK <SRS 82.162>
000095  * BACKUP KNOWS HOW TO RESET THIS BIT. <SRS 82.162>
000096                  STA         BKBITFLG            ; BIT (USED BY DREVISE)
000097  SETINF1X        LDX         INFTABL,Y           ; GET INDEX INTO CORESPONDING 'D.' TABLE
000098                  BMI         SETINF2             ; BRANCH IF WE'VE REACHED STORAGE TYPE PARAMETER
000099                  LDA         (C.FILIST),Y
000100                  STA         DFIL,X
000101                  INY                             ; HAS USER'S REQUEST BEEN SATISFIED?
000102                  CPY         C.FILSTLN
000103                  BNE         SETINF1X            ; NO, MOVE NEXT BYTE.
000104  SINFEND         JMP         DREVISE             ; GO UPDATE DIRECTORY WITH CURRENT TIME.
000105  *
000106  SETINF2         LDY         C.FILSTLN           ; TEST TO SEE IF USER WANTS HIS TIME STAMP ADDED
000107                  CPY         #$F                 ; (LIST MUST BE AT LEAST $F BYTES LONG)
000108                  BCC         SINFEND             ; NO PUT CURRENT TIME INSTEAD.
000109                  LDY         #$B                 ; MOVE IN THE NEXT GROUP OF BYTES
000110  SETINF3         LDX         INFTABL,Y
000111                  BMI         SINFEND1
000112                  LDA         (C.FILIST),Y
000113                  STA         DFIL,X
000114                  INY
000115                  CPY         C.FILSTLN           ; SATISFACTION YET?
000116                  BNE         SETINF3             ; NOPE, KEEP EM PUMPIN'
000117  SINFEND1        JMP         DREVISE1
000118  *
000119  BKBITFLG        DS          1                   ; FOR TURNING OFF BACKUP BIT
000120  *
000121  *
000122  INFTABL         DFB         D.ATTR,D.FILID,D.AUXID,D.AUXID+1
000123                  DFB         D.STOR+1+$80,D.EOF,D.EOF+1,D.EOF+2 ; (D.STOR=0 THUS D.STOR+1 WAS NECESSARY)
000124                  DFB         $80,D.USAGE,D.USAGE+1,D.MODDT ; (THE $80 IS FOR THE FOURTH BYTE OF EOF)
000125                  DFB         D.MODDT+1,D.MODTM,D.MODTM+1,$FF ; TABLE ALWAYS ENDS IN $FF
000126                  PAGE
000127  *
000128  RENAME          JSR         LOOKFILE            ; LOOK FOR SOURCE (ORIGINAL) FILE.
000129                  BCC         RNAME0              ; BRANCH IF FOUND.
000130                  CMP         #BADPATH            ; TRYING TO RENAME A VOLUME?
000131                  BNE         RNAMERR             ; NO, RETURN OTHER ERROR.
000132                  JSR         RENPATH             ; SYNTAX NEW NAME.
000133                  BCS         RNAMERR
000134                  LDA         WRKPATH             ; FIND OUT IF ONLY ROOTNAME FOR NEW NAME
000135                  CMP         PATHNML
000136                  BNE         RNBADPTH            ; NOT SINGLE NAME, RETURN ERROR!
000137                  LDY         #VCBSTAT            ; TEST FOR OPEN FILES BEFORE CHANGING
000138                  LDA         (VCBPTR),Y
000139                  BPL         RNAMEVOL            ; BRANCH IF VOLUME NOT BUSY
000140                  LDA         #FILBUSY
000141  SINFOERR        EQU         *
000142                  RTS                             ; (CARRY IS SET)
000143  RNAMEVOL        LDY         #0                  ; GET NEWNAME'S LENGTH.
000144                  LDA         (WRKPATH),Y
000145                  TAY
000146                  ORA         #$F0                ; (ROOT FILE STORAGE TYPE)
000147                  JSR         MVROTNAM            ; UPDATE ROOT DIRECTORY.
000148                  BCS         RNAMERR
000149                  LDY         #0
000150                  LDA         (WRKPATH),Y         ; UPDATE VCB ALSO.
000151                  TAY
000152  RNMEVOL         LDA         (WRKPATH),Y
000153                  STA         (VCBPTR),Y
000154                  DEY
000155                  BPL         RNMEVOL
000156                  CLC
```



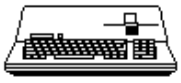
```
000157                  RTS
000158  *
000159  RNAME0          JSR     RENPATH             ; SET UP AND SYNTAX NEW NAME.
000160                  BCS     RNAMERR
000161                  LDY     #0                  ; VERIFY THAT BOTH NAMES HAVE SAME ROOT.
000162                  LDA     (PATHNML),Y
000163                  TAY
000164  TSTSMROT        LDA     (PATHNML),Y         ; COMPARE NEWNAME'S ROOT NAME WITH
000165                  CMP     (VCBPTR),Y          ; OLD NAME'S VOLUME NAME.
000166                  BNE     RNBADPTH            ; RETURN 'BADPATH' IF NOT SAME VOLUME.
000167                  DEY
000168                  BPL     TSTSMROT            ; (TEST SAME 'ROT')
000169                  JSR     LOOKFILE            ; TEST FOR DUPLICATE FILE NAME.
000170                  BCS     TSTFNF1             ; BRANCH IF ERROR TO TEST FOR FILE NOT FOUND.
000171                  LDA     #DUPERR             ; TELL USER THAT NEW NAME ALREADY EXISTS.
000172  RNAMERR         SEC
000173                  RTS
000174                  PAGE
000175  TSTFNF1         CMP     #FNFERR             ; WAS IT A VALID FILE NOT FOUND?
000176                  BNE     RNAMERR             ; NO, RETURN OTHER ERROR CODE.
000177                  LDX     #2                  ; NOW MOVE NEW NAME'S OWNERSHIP (DIRECTORY HEADER) I.D.
000178  SVENEWID        LDA     D.DEV,X             ; THIS CONSISTS OF THE UNIT NUMBER,
000179                  STA     NPATHDEV,X          ; AND THE ADDRESS OF THE DIRECTORY THE FILE
000180                  DEX                         ; WASN'T FOUND IN. LOGIC BY NEGATION...
000181                  BPL     SVENEWID
000182                  JSR     SETPATH             ; NOW SYNTAX THE PATHNAME OF THE FILE TO BE CHANGED.
000183                  BCS     RNAMERR
000184                  JSR     FINDFILE            ; GET ALL THE INFO ON THIS ONE.
000185                  BCS     RNAMERR
000186                  JSR     TSTOPEN             ; DON'T ALLOW RENAME TO OCCUR IF FILE IS IN USE.
000187                  LDA     #FILBUSY            ; ANTICIPATE ERROR
000188                  BCS     RNAMERR
000189                  LDA     DFIL+D.ATTR         ; TEST BIT THAT SAYS IT'S OK TO RENAME
000190                  AND     #RENAMEN
000191                  BNE     RNAME1              ; BRANCH IF IT'S ALRIGHT TO RENAME.
000192                  LDA     #ACCSERR            ; OTHERWISE REPORT ILLEGAL ACCESS.
000193                  SEC
000194                  RTS
000195  *
000196  RNAME1          LDX     #2                  ; NOW TEST TO SEE IF NEW PATHNAME FITS IN THE
000197  SAMOWNR         LDA     D.DEV,X             ; SAME DIRECTORY FILE.
000198                  CMP     NPATHDEV,X
000199                  BEQ     RNAME2
000200  RNBADPTH        LDA     #BADPATH            ; TELL USER THAT PATHNAMES INCOMPATABLE.
000201                  SEC
000202                  RTS
000203  *
000204  RNAME2          DEX                         ; TEST ALL THREE BYTES.
000205                  BPL     SAMOWNR
000206                  JSR     RENPATH             ; WELL... SINCE BOTH NAMES WOULD GO INTO THE
000207                  BCS     RNAMERR             ; DIRECTORY, RE-SYNTAX THE NEW NAME TO GET LOCAL NAME ADDRESS.
000208                  TYA                         ; (Y CONTAINS THE LOCAL NAME LENGTH+1)
000209                  BEQ     RNBADPTH            ; REPORT ERROR IF LENGTH INFO NOT IMMEDIATELY AVAILABLE.
000210                  DEY                         ; (REMOVE THE +1)
000211  RNAME3          LDA     (WRKPATH),Y         ; MOVE LOCAL NAME TO DIR ENTRY WORKSPACE.
000212                  STA     DFIL+D.STOR,Y
000213                  DEY
000214                  BNE     RNAME3
000215                  LDA     DFIL+D.STOR         ; PRESERVE FILE STORAGE TYPE.
000216                  AND     #$F0                ; STRIP OFF OLD NAME LENGTH.
000217                  TAX
000218                  ORA     (WRKPATH),Y         ; ADD IN NEW NAME'S LENGTH
000219                  STA     DFIL+D.STOR
000220                  CPX     #DIRTYP*16          ; THAT FILE MUST BE CHANGED ALSO.
000221                  BNE     RNAMDONE            ; BRANCH IF NOT DIRECTORY TYPE.
000222                  PAGE
000223                  LDA     DFIL+D.FRST         ; READ IN FIRST (HEADER) BLOCK OF SUB DIRECTORY
000224                  STA     BLOKNML
000225                  LDA     DFIL+D.FRST+1
000226                  STA     BLOKNMH
000227                  JSR     RDGBUF
000228                  BCS     RNAMERR             ; REPORT ERRORS
000229                  LDY     #0                  ; CHANGE THE HEADER'S NAME TO MATCH THE OWNER'S NEW NAME.
000230                  LDA     (WRKPATH),Y         ; GET LOCAL NAME LENGTH AGAIN
000231                  TAY
000232                  ORA     #HEDTYP*16          ; ASSUME IT'S A HEADER.
000233                  JSR     MVROTNAM
000234                  BCS     RNAMERR
000235  RNAMDONE        JMP     DREVISE1            ; END BY UPDATING ALL PATH DIRECTORIES
000236  *
000237  *
```



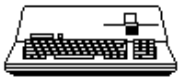
```
000238  MVROTNAM        STA       GBUF+4
000239  MVHEDNAM        LDA       (WRKPATH),Y
000240                  STA       GBUF+4,Y
000241                  DEY
000242                  BNE       MVHEDNAM
000243                  JMP       WRTGBUF             ; WRITE CHANGED HEADER BLOCK.
000244  *
000245  *
000246  RENPATH         LDA       C.NWPATH            ; GET ADDRESS TO NEW PATHNAME.
000247                  STA       TPATH
000248                  LDA       C.NWPATH+1          ; SET UP FOR SYNTAXING ROUTINE (SYNPATH).
000249                  STA       TPATH+1
000250                  LDA       SSNWPATH            ; (MOVE BYTE FOR SISTER PAGE, TOO.)
000251                  STA       SISTPATH
000252                  JMP       SYNPATH             ; GO SYNTAX IT. (RETURNS LAST LOCAL NAME LENGTH IN Y).
000253  *
000254  *
000255  DEALBLK         LDY       #0                  ; BEGIN AT THE BEGINNING.
000256  DALBLK1         STY       SAPTR               ; SAVE CURRENT INDEX.
000257                  LDA       GBUF,Y              ; GET ADDRESS (LOW) OF BLOCK TO BE DEALLOCATED.
000258                  CMP       GBUF+$100,Y         ; TEST FOR NUL BLOCK.
000259                  BNE       DALBLK2             ; BRANCH IF NOT NUL.
000260                  CMP       #0
000261                  BEQ       DALBLK3             ; SKIP IT IF NUL.
000262  DALBLK2         LDX       GBUF+$100,Y         ; GET THE REST OF THE BLOCK ADDRESS.
000263                  JSR       DEALLOC             ; FREE IT UP ON VOLUME BIT MAP.
000264                  BCS       DALBLKERR           ; RETURN ANY ERROR.
000265                  LDY       SAPTR               ; GET INDEX TO SAPLING LEVEL INDEX BLOCK AGAIN.
000266  DALBLK3         INY                           ; POINT AT NEXT BLOCK ADDRESS.
000267                  BNE       DALBLK1             ; BRANCH IF MORE TO DEALLOCATE (OR TEST).
000268                  CLC                           ; INDICATE NO ERROR.
000269  DALBLKERR       RTS
000270  *
000271  *
000272                  PAGE
000273  *
000274  DESTROY         JSR       FINDFILE            ; LOOK FOR FILE TO BE WIPED OUT.
000275                  BCS       DESTERR             ; PASS BACK ANY ERROR.
000276                  JSR       TSTOPEN             ; IS THIS FILE OPEN?
000277                  LDA       TOTENT
000278                  BEQ       DSTROY1             ; BRANCH IF FILE NOT OPEN.
000279                  LDA       #FILBUSY
000280                  SEC                           ; INFORM USER THAT FILE CAN'T BE DESTORYED AT THIS TIME.
000281                  RTS
000282  *
000283  DSTROY1         LDA       #0                  ; FORCE PROPER FREE COUNT IN VOLUME.
000284                  STA       REQL                ; (NO DISK ACCESS OCCURS IF ALREADY PROPER)
000285                  STA       REQH
000286                  JSR       TSFRBLK
000287                  BCC       DSTROY2
000288                  CMP       #OVRERR             ; WAS IT JUST A FULL DISK?
000289                  SEC
000290                  BNE       DESTERR             ; NOPE, REPORT ERROR.
000291  *
000292  DSTROY2         LDA       DFIL+D.ATTR         ; MAKE SURE IT'S OK TO DESTROY THIS FILE.
000293                  AND       #DSTROYEN
000294                  BNE       DSTROY3             ; BRANCH IF OK.
000295                  LDA       #ACCSERR            ; TELL USER IT'S NOT KOSHER.
000296                  JSR       SYSERR              ; (RETURNS TO CALLER OF DESTORY)
000297  *
000298  DSTROY3         JSR       TWRPROT1            ; BEFORE GOING THRU DEALLOCATION,
000299                  BCS       DESTERR             ; TEST FOR WRITE PROTECTED HARDWARE.
000300                  LDA       DFIL+D.STOR         ; FIND OUT WHICH STORAGE TYPE.
000301                  AND       #$F0                ; STRIP OFF NAME LENGTH.
000302                  CMP       #TRETYP+1*$10       ; IS IT A SEED, SAPLING, OR TREE?
000303                  BCC       DSTREE              ; BRANCH IF IT IS.
000304                  JMP       DSTDIR              ; OTHERWISE TEST FOR DIRECTORY DESTROY.
000305  *
000306  DSTREE          JSR       GTTINDX             ; GET A BIT MAP BUFFER AND TEMPORARY INDEX BUFFER.
000307                  BCS       DESTERR
000308                  LDA       DFIL+D.STOR         ; GET STORAGE TYPE AGAIN
000309                  AND       #$F0
000310                  CMP       #TRETYP*$10         ; IS THIS A TREE (FULL 2-LEVEL)?
000311                  BNE       DSTSAP              ; NO, TEST FOR SAPLING.
000312                  JSR       RDFRST              ; READ IN ROOT INDEX FOR THIS FILE.
000313                  BCC       DSTRE2              ; BRANCH IF ALL IS WELL.
000314  DESTERR         RTS                           ; OTHERWISE RETURN ERROR.
000315  *
000316  DSTSAP          CMP       #SAPTYP*$10         ; IS IT A SAPLING
000317                  BNE       DSTLAST             ; NO, JUST DEALLOCATE FIRST (AND ONLY) BLOCK.
000318                  JSR       ZTMPIDX             ; CLEAR OUT TEMPORARY INDEX BUFFER.
```



```
000319                  LDA     DFIL+D.FRST         ; MAKE THIS SAP LOOK LIKE A TREE...
000320                  LDY     #0                  ; THIS IS DONE BY PLACING THE FIRST BLOCK ADDRESS
000321                  STA     (TINDX),Y           ; IN THE TEMP (TOP) INDEX BUFFER AS
000322                  INC     TINDX+1
000323                  LDA     DFIL+D.FRST+1       ; A SUB INDEX WOULD APPEAR.
000324                  STA     (TINDX),Y
000325                  DEC     TINDX+1
000326  DSTRE2          LDY     #0                  ; BEGIN SCAN OF TOP LEVEL INDEX AT ZERO.
000327  DSTNXT          STY     TREPTR              ; SAVE POINTER TO TREE LEVEL.
000328                  LDA     (TINDX),Y           ; GET BLOCK ADDRESS OF A SUB INDEX BLOCK
000329                  INC     TINDX+1             ; (TEST FOR NUL BLOCK)
000330                  CMP     (TINDX),Y
000331                  BNE     DSTRE3              ; BRANCH IF WE'VE GOT AN BLOCK TO DEALLOCATE.
000332                  CMP     #0                  ; IS ENTIRE ADDRESS ZERO?
000333                  BEQ     DSTRE4              ; YES, DO NEXT. (CARRY SET)
000334  DSTRE3          CLC                         ; INDICATE THERE IS A BLOCK OF INDEXES TO FREE UP.
000335                  STA     BLOKNML
000336                  LDA     (TINDX),Y           ; GET HI ADDRESS TOO.
000337                  STA     BLOKNMH
000338  DSTRE4          DEC     TINDX+1             ; (RESTORE PROPER ADDRESS FOR BUFFER)
000339                  BCS     DSTNXT1             ; BRANCH IF NO SUB INDEX.
000340                  JSR     RDGBUF              ; USE GENERAL BUFFER FOR SUB INDEX BUFFER.
000341                  BCS     DESTERR
000342                  JSR     DEALBLK             ; GO FREE UP BLOCKS IN SUB INDEX
000343                  BCS     DESTERR
000344                  LDY     TREPTR              ; AND FREE UP SUB INDEX BLOCK TOO.
000345                  INC     TINDX+1
000346                  LDA     (TINDX),Y
000347                  TAX
000348                  DEC     TINDX+1
000349                  LDA     (TINDX),Y
000350                  JSR     DEALLOC
000351                  BCS     DESTERR
000352                  LDY     TREPTR
000353  DSTNXT1         INY                         ; HAVE ALL SUB INDEXES BEEN LOCATED?
000354                  BNE     DSTNXT              ; NO, DO NEXT...
000355  DSTLAST         LDA     DFIL+D.FRST         ; DEALLOCATE FIRST BLCOK OF FILE.
000356                  LDX     DFIL+D.FRST+1
000357                  JSR     DEALLOC
000358                  BCS     DESTERR
000359                  LDA     #0                  ; UPDATE DIRECTORY TO FREE ENTRY SPACE.
000360                  STA     DFIL+D.STOR
000361                  CMP     H.FCNT              ; FILE ENTRY WRAP?
000362                  BNE     DST1                ; BRANCH IF NO CARRY ADJUSTMENT
000363                  DEC     H.FCNT+1            ; TAKE CARRY FROM HIGH BYTE OF FILE ENTRIES
000364  DST1            DEC     H.FCNT              ; MARK HEADER WITH ONE LESS FILE
000365                  LDX     BMTAB               ; UPDATE (LAST) BITMAP.
000366                  JSR     BMAPUP
000367                  BCS     DESTERR
000368                  LDY     #VCBTFRE
000369                  LDA     DFIL+D.USAGE
000370                  ADC     (VCBPTR),Y
000371                  STA     (VCBPTR),Y          ; UPDATE CURRENT FREE BLOCK COUNT.
000372                  INY
000373                  LDA     DFIL+D.USAGE+1
000374                  ADC     (VCBPTR),Y
000375                  STA     (VCBPTR),Y
000376                  LDA     #0                  ; FORCE RESCAN FROM FIRST BITMAP
000377                  LDY     #VCBCMAP
000378                  STA     (VCBPTR),Y
000379                  JMP     DREVISE             ; UPDATE DIRECTORY LAST...
000380  *
000381                  PAGE
000382  *
000383  DSTDIR          CMP     #DIRTYP*16          ; IS THIS A DIRECTORY FILE?
000384                  BEQ     DSDIR1              ; YES, PROCEED.
000385                  LDA     #CPTERR             ; FILE IS NOT COMPATABLE.
000386                  JSR     SYSERR              ; GIVE UP.
000387  *
000388  DSDIR1          JSR     FNDBMAP             ; MAKE SURE A BUFFER IS AVAILABLE FOR THE BITMAP.
000389                  BCS     DSDIRERR
000390                  LDA     DFIL+D.FRST         ; READ IN FIRST BLOCK OF DIRECTORY INTO GBUF.
000391                  STA     BLOKNML
000392                  LDA     DFIL+D.FRST+1
000393                  STA     BLOKNMH
000394                  JSR     RDGBUF
000395                  BCS     DSDIRERR
000396                  LDA     GBUF+HCENT+4        ; FIND OUT IF ANY FILES EXIST ON THIS DIRECTORY.
000397                  BNE     DSDIRACC            ; BRANCH IF ANY EXIST.
000398                  LDA     GBUF+HCENT+5
000399                  BEQ     DSDIR2
```

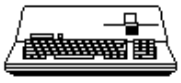


```
000400  DSDIRACC        LDA       #ACCSERR
000401                  JSR       SYSERR
000402  *
000403  DSDIR2          LDA       GBUF+2              ; GET FORWARD LINK.
000404                  CMP       GBUF+3              ; TEST FOR NO LINK.
000405                  BNE       DSDIR3
000406                  CMP       #0
000407                  BEQ       DSTLAST             ; IF NO LINK, THEN FINISHED.
000408  DSDIR3          LDX       GBUF+3
000409                  JSR       DEALLOC             ; FREE THIS BLOCK.
000410                  BCS       DSDIRERR
000411                  LDA       GBUF+2
000412                  STA       BLOKNML
000413                  LDA       GBUF+3
000414                  STA       BLOKNMH             ; READ IN LINKED BLOCK.
000415                  JSR       RDGBUF
000416                  BCC       DSDIR2              ; LOOP UNTIL ALL ARE FREED.
000417  DSDIRERR        RTS
000418  *
000419  *
000420                  PAGE
000421  WORKSPC         EQU       *
000422  V.STATUS        DS        1                   ; VOLUME STATUS, INCLUDES 'ACTIVE' IN BIT 7
000423  H.CREDT         DS        2                   ; DIRECTORY CREATION DATE
000424                  DS        2                   ; DIRECTORY CREATION TIME
000425                  DS        1                   ; VERSION UNDER WHICH THIS DIRECTORY WAS CREATED
000426                  DS        1                   ; EARLIEST VERSION THAT IT'S COMPATABLE WITH
000427  H.ATTR          DS        1                   ; ATTRIBUTES (PROTECT BIT, ETC.)
000428  H.ENTLN         DS        1                   ; LENGTH OF EACH ENTRY IN THIS DIRECTORY.
000429  H.MAXENT        DS        1                   ; MAXIMUM NUMBER OF ENTRIES PER BLOCK
000430  H.FCNT          DS        2                   ; CURRENT NUMBER OF FILES IN THIS DIRECTORY
000431                  DS        2                   ; ADDRESS OF FIRST ALLOCATION BIT MAP
000432                  DS        2                   ; TOTAL NUMBER OF BLOCKS ON THIS UNIT
000433                  DS        5                   ; (FOR FUTURE EXPANSION)
000434  *
000435  D.DEV           DS        1                   ; DEVICE NUMBER OF THIS DIRECTORY ENTRY
000436  D.HEAD          DS        2                   ; ADDRESS OF <SUB> DIRECTORY HEADER
000437  D.ENTBLK        DS        2                   ; ADDRESS OF BLOCK WHICH CONTAINS THIS ENTRY
000438  D.ENTNUM        DS        1                   ; ENTRY NUMBER WITHIN BLOCK.
000439  DFIL            EQU       *
000440  D.STOR          EQU       *-DFIL              ; STORAGE TYPE * 16 + FILE NAME LENGTH
000441                  DS        1
000442  ; *-DFIL ; FILE NAME
000443                  DS        15
000444  D.FILID         EQU       *-DFIL              ; USER'S IDENTIFICATION BYTE
000445                  DS        1
000446  D.FRST          EQU       *-DFIL              ; FIRST BLOCK OF FILE
000447                  DS        2
000448  D.USAGE         EQU       *-DFIL              ; NUMBER OF BLOCKS CURRENTLY ALLOCATED TO THIS FILE
000449                  DS        2
000450  D.EOF           EQU       *-DFIL              ; CURRENT END OF FILE MARKER
000451                  DS        3
000452  D.CREDT         EQU       *-DFIL              ; DATE OF FILE'S CREATION
000453                  DS        2
000454  ; *-DFIL ; TIME OF FILE'S CREATION
000455                  DS        2
000456  ;  EQU *-DFIL ; SOS VERSION THAT CREATED THIS FILE
000457                  DS        1
000458  D.COMP          EQU       *-DFIL              ; BACKWARD VERSION COMPATABILTY
000459                  DS        1
000460  D.ATTR          EQU       *-DFIL              ; 'PROTECT', READ/WRITE 'ENABLE' ETC.
000461                  DS        1
000462  D.AUXID         EQU       *-DFIL              ; USER AUXILLARY IDENTIFACATION
000463                  DS        2
000464  D.MODDT         EQU       *-DFIL              ; FILE'S LAST MODIFICATION DATE
000465                  DS        2
000466  D.MODTM         EQU       *-DFIL              ; FILE'S LAST MODIFICATION TIME
000467                  DS        2
000468  D.DHDR          EQU       *-DFIL              ; HEADER BLOCK ADDRESS OF FILE'S DIRECTORY
000469                  DS        2
000470  *
000471  CMDADR          DS        2
000472  SCRTCH          DS        13                  ; SCRATCH AREA FOR ALLOCATION ADDRESS CONVERSION
000473  OLDEOF          DS        3                   ; TEMP USED IN W/R
000474  OLDMARK         DS        3                   ; USED BY 'RDPOSN' AND 'WRITE'
000475  SCRHIGH         EQU       <SCRTCH             ; AND DEVICE NUMBERS FROM BOB'S CODE.
000476  *
000477                  CHN       SWAPOUT/IN,4,2
000478
000479  ********************************************************************
000480  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DESTROY
```

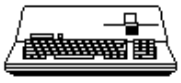


```
000481  ************************************************************************
000482
```

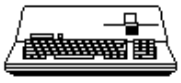
End of File -- Lines: 482 Characters: 20557



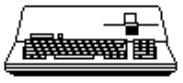
```
================================================================================
FILE: "SOS.DEVMGR.SRC.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: DEVMGR.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1 DEVICE MANAGER"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGDMGR
000010  ZZORG           EQU        *
000011                  MSB        OFF
000012                  REP        100
000013  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000014  *                     ALL RIGHTS RESERVED
000015                  REP        100
000016  *
000017  * DEVICE MANAGER (VERSION = 1.1O   )
000018  *                (DATE    = 8/04/81)
000019  *
000020  * THIS MODULE IS RESPONSIBLE FOR CALLING THE CORRECT DEVICE
000021  * DRIVER WHEN A D.READ...D.INIT SYSTEM CALL IS MADE.
000022  * (NOTE:  D.OPEN,D.CLOSE AND D.INIT ARE ONLY CALLABLE FROM
000023  * INSIDE THE OPERATING SYSTEM).  D.INFO AND GET.DNUM CALLS
000024  * ARE HANDLED INSIDE THIS MODULE. REPEAT.IO BYPASSES THIS MODULE.
000025                  REP        100
000026  *
000027                  ENTRY      DMGR
000028  *
000029                  ENTRY      MAX.DNUM
000030                  ENTRY      SDT.SIZE
000031                  ENTRY      SDT.DIBL
000032                  ENTRY      SDT.DIBH
000033                  ENTRY      SDT.ADRL
000034                  ENTRY      SDT.ADRH
000035                  ENTRY      SDT.BANK
000036                  ENTRY      SDT.UNIT
000037                  ENTRY      BLKD.SIZE
000038                  ENTRY      BLKDLST
000039  *
000040                  EXTRN      SYSERR
000041                  EXTRN      SERR
000042                  EXTRN      NODNAME
000043                  EXTRN      BADDNUM
000044                  EXTRN      SYSDEATH
000045                  EXTRN      BADSYSCALL
000046  *
000047                  EXTRN      SXPAGE
000048  *
000049  E.REG           EQU        $FFDF               ; ENVIRONMENT REGISTER
000050  B.REG           EQU        $FFEF               ; BANK REGISTER
000051                  PAGE
000052                  REP        100
000053  *
000054  * SYSTEM DEVICE TABLE (SDT)
000055  *
000056  * CONTAINS THE ADDRESS OF EACH DRIVER'S DIB (SDT.DIB), THE
000057  * ADDRESS OF EACH DRIVER'S ENTRY POINT (SDT.ADR), AND THE
000058  * UNIT # OF EACH DRIVER (SDT.UNIT).  THE TABLE IS INDEXED
000059  * BY DEVICE NUMBER.  ENTRY 0 IS RESERVED FOR FUTURE USE.
000060  *
000061                  REP        100
000062  *
000063  SDT.SIZE        EQU        25
000064  *
000065  MAX.DNUM        DS         1                   ;MAX DEV NUMBER IN SYSTEM+1
000066  SDT.DIBL        DS         SDT.SIZE            ;ADR OF DEVICE INFORMATION BLOCK
000067  SDT.DIBH        DS         SDT.SIZE
000068  *
000069  SDT.ADRL        DS         SDT.SIZE            ;ADR OF ENTRY POINT
000070  SDT.ADRH        DS         SDT.SIZE
000071  *
000072  SDT.BANK        DS         SDT.SIZE            ;BANK # OF DEVICE
000073  *
000074  SDT.UNIT        DS         SDT.SIZE            ;UNIT  # OF DRIVER
000075  *
000076                  REP        100
```



```
000077  * BLOCK DEVICE LIST TABLE
000078  *
000079  BLKD.SIZE       EQU         13
000080  BLKDLST         DFB         $00
000081                  DS          BLKD.SIZE-1
000082                  PAGE
000083                  REP         100
000084  *
000085  * DATA DECLARATIONS
000086  *
000087                  REP         100
000088  *
000089  D.TPARMX        EQU         $C0
000090  REQCODE         EQU         D.TPARMX
000091  *
000092  * D.READ/WRITE/CTRL/STATUS/OPEN/CLOSE/INIT/REPEAT PARMS
000093  *
000094  DNUM            EQU         D.TPARMX+1
000095  *
000096  * D.INFO PARMS
000097  *
000098  I.DNUM          EQU         D.TPARMX+1
000099  I.DNAME         EQU         D.TPARMX+2
000100  I.DLIST         EQU         D.TPARMX+4
000101  I.LENGTH        EQU         D.TPARMX+6
000102  *
000103  * GET.DEV.NUM PARMS
000104  *
000105  G.DNAME         EQU         D.TPARMX+1
000106  G.DNUM          EQU         D.TPARMX+3
000107  *
000108  * SDT ENTRY (=DIB) FIELDS
000109  *
000110  DIB.SLOT        EQU         $11                 ;DIB'S DEVICE SLOT FIELD
000111  DIB.DTYPE       EQU         $13                 ;DIB'S DEVICE TYPE FIELD
000112  *
000113  SDTP            EQU         D.TPARMX+$10        ; PTR TO CURRENT SDT ENTRY
000114                  PAGE
000115                  REP         100
000116  *
000117  * DEVICE MANAGER (MAIN ENTRY POINT)
000118  *
000119                  REP         100
000120  DMGR            EQU         *
000121  *
000122                  LDA         REQCODE
000123                  CMP         #4
000124                  BCC         DRIVER              ; D.READ/WRITE/CTRL/STATUS CALL
000125                  BNE         DM000
000126                  JMP         GET.DNUM            ; GET.DEV.NUM CALL
000127  DM000           CMP         #5
000128                  BEQ         D.INFO              ; D.INFO CALL
000129                  CMP         #$A
000130                  BCC         DRIVER              ; D.OPEN/CLOSE/INIT
000131                  LDA         #BADSYSCALL         ; ELSE FATAL ERROR
000132                  JSR         SYSDEATH            ; EXIT
000133                  PAGE
000134                  REP         100
000135  * D.READ/WRITE/CTRL/STATUS/OPEN/CLOSE/INIT CALLS
000136  * "JSR" TO DEVICE DRIVER
000137                  REP         100
000138  DRIVER          EQU         *
000139  *
000140                  LDX         DNUM                ; GET DNUM SYSCALL PARM
000141                  BEQ         DM005               ; WITHIN BOUNDS?
000142                  CPX         MAX.DNUM            ;      "
000143                  BCC         DM010
000144  *
000145  * DNUM TOO LARGE
000146  *
000147  DM005           LDA         #>BADDNUM           ; INVALID DEVICE NUMBER
000148                  JSR         SYSERR              ;   ERROR EXIT
000149  *
000150  * MAP DEV# TO UNIT#
000151  *
000152  DM010           LDA         SDT.UNIT,X
000153                  STA         DNUM
000154  *
000155  * "JSR" TO DEVICE DRIVER VIA JMP TABLE
000156  *
000157                  LDA         B.REG               ; STACK B.REG
```

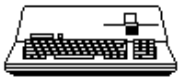


```
000158                  PHA
000159                  LDA         #<DM.RTN-1          ; STACK RETURN ADDRESS
000160                  PHA
000161                  LDA         #>DM.RTN-1
000162                  PHA
000163  *
000164                  LDA         SDT.BANK,X          ; SELECT RAM BANK
000165                  STA         B.REG
000166                  LDA         SDT.ADRH,X          ; STACK DRIVER ENTRY POINT ADDRESS
000167                  PHA
000168                  LDA         SDT.ADRL,X
000169                  PHA
000170  *
000171                  LDA         E.REG               ; SWITCH IN I/O BANK
000172                  ORA         #$40
000173                  STA         E.REG
000174                  RTS                             ; AND, "JSR" TO DEVICE DRIVER
000175  *
000176  DM.RTN          LDA         E.REG               ; SWITCH OUT I/O BANK
000177                  AND         #$BF
000178                  STA         E.REG
000179                  PLA                             ; RESTORE B.REG
000180                  STA         B.REG
000181                  SEC
000182                  LDA         SERR                ; RETRIEVE ERROR CODE
000183                  BNE         DM017               ; ENSURE CARRY CLEARED IF NO ERROR
000184                  CLC
000185  DM017           RTS                             ; AND, EXIT TO CALLER
000186                  PAGE
000187                  REP         100
000188  * D.INFO(IN.DNUM, OUT.DNAME, OUT.DEVLIST, IN.LENGTH) SYSTEM CALL
000189                  REP         100
000190  D.INFO          EQU         *
000191  *
000192                  LDX         I.DNUM              ; GET DNUM PARM
000193                  BEQ         DM020               ; WITHIN BOUNDS?
000194                  CPX         MAX.DNUM            ;      "
000195                  BCC         DM030
000196  DM020           LDA         #>BADDNUM           ; NO, DNUM TOO LARGE
000197                  JSR         SYSERR              ; ERROR EXIT
000198  *
000199  * MOVE PARMS FM SDT ENTRY (DEV INFO BLOCK) TO CALLER'S
000200  * PARM LIST
000201  *
000202  DM030           JSR         SETUP.SDT           ; SET UP ZPAGE PTR TO SDT ENTRY
000203  *
000204  * OUPUT DNAME PARM
000205  *
000206                  LDA         (SDTP),Y            ; LOAD PARM'S BYTE COUNT
000207                  TAY
000208  DM040           LDA         (SDTP),Y
000209                  STA         (I.DNAME),Y
000210                  DEY
000211                  BPL         DM040
000212  *
000213  * OUTPUT DEVINFO PARM (SLOT,UNIT,DEVID,PRODCODE)
000214  *
000215                  LDA         #DIB.SLOT
000216                  CLC                             ; ADVANCE SDTP TO 2ND PARM IN SDT
000217                  ADC         SDTP
000218                  STA         SDTP
000219                  BCC         DM045
000220                  INC         SDTP+1
000221  DM045           LDY         I.LENGTH            ; LOAD BYTE COUNT
000222                  BEQ         DM.EXIT             ; IF 0 THEN DONE
000223                  DEY
000224                  CPY         #$B
000225                  BCC         DM050
000226                  LDY         #$A
000227  DM050           LDA         (SDTP),Y
000228                  STA         (I.DLIST),Y
000229                  DEY
000230                  BPL         DM050
000231  *
000232  DM.EXIT         CLC
000233                  RTS                             ; NORMAL EXIT
000234                  PAGE
000235                  REP         100
000236  * GET.DEV.NUM(IN.DNAME; OUT.DNUM) SYSTEM CALL
000237                  REP         100
000238  *
```

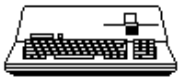


```
000239  GET.DNUM        EQU       *
000240  *
000241                  LDX       #1                  ; SETUP PTR TO 1ST SDT ENTRY
000242  *
000243  DM070           JSR       SETUP.SDT           ; SET UP ZPAGE PTR TO SDT ENTRY
000244  *
000245                  LDA       (SDTP),Y            ; COMPARE DNAME LENGTHS
000246                  CMP       (G.DNAME),Y
000247                  BNE       NXTSDT
000248  *
000249                  TAY                           ; LENGTHS MATCH, NOW COMPARE CHARS
000250  DM080           LDA       (G.DNAME),Y
000251                  CMP       #$60
000252                  BCC       DM090
000253                  AND       #$DF                ; UPSHIFT
000254  DM090           CMP       (SDTP),Y
000255                  BNE       NXTSDT
000256                  DEY
000257                  BNE       DM080
000258  *
000259                  TXA                           ; CHARS MATCH
000260                  LDY       #0
000261                  STA       (G.DNUM),Y          ; OUTPUT DEV NUM PARM
000262                  LDY       #DIB.DTYPE          ; SET "N" FLAG IN STATUS REG.
000263                  LDA       (SDTP),Y            ; N=1(BLOCK DEVICE) N=0(CHAR DEVICE)
000264                  CLC
000265                  RTS                           ; NORMAL EXIT
000266  *
000267  NXTSDT          INX                           ; LAST SDT ENTRY?
000268                  CPX       MAX.DNUM
000269                  BCC       DM070
000270  *
000271                  LDA       #>NODNAME           ; ERROR, DNAME NOT FOUND IN SDT
000272                  JSR       SYSERR              ;  RETURN TO CALLER
000273                  PAGE
000274                  REP       100
000275  * SETUP.SDT(IN.X=DNUM, OUT.SDTP, B.REG, Y=0)   X="UNCHANGED"
000276                  REP       100
000277  SETUP.SDT       EQU       *
000278                  LDA       SDT.DIBL,X          ; SET UP ZPAGE PTR TO SDT ENTRY
000279                  STA       SDTP                ; (POINTS TO DNAME FIELD)
000280                  LDA       SDT.DIBH,X
000281                  STA       SDTP+1
000282                  LDA       SDT.BANK,X
000283                  STA       B.REG
000284                  LDY       #0
000285                  STY       SXPAGE+SDTP+1
000286                  RTS
000287  *
000288                  LST       ON
000289  ZZEND           EQU       *
000290  ZZLEN           EQU       ZZEND-ZZORG
000291                  IFNE      ZZLEN-LENDMGR
000292                  FAIL      2,"SOSORG           FILE IS INCORRECT FOR DEVMGR"
000293                  FIN
000294
000295  ********************************************************************************
000296  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DEVMGR.SRC
000297  ********************************************************************************
000298
```

End of File -- Lines: 298 Characters: 7991

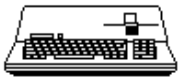


```
================================================================================
FILE: "SOS.DISK3.DATA.SRC.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.DATA.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007  * GENERAL DATA:
000008  *
000009  PREVUNIT        DS          1                   ;PRIOR UNIT ACCESSED (FOR REPEAT)
000010  PREVCMD         DS          1                   ;PRIOR CMD (FOR REPEAT)
000011  *
000012  ESAVE           DS          1                   ;SAVED E.REG
000013  VBLSAVE         DS          1                   ;SAVED E.IER
000014  INITFLAG        DFB         0                   ;<0 IS INITTED
000015                  DO          REV0ROM
000016  ROMREV          DS          1                   ;0=REV0, <>0=REV1
000017                  FIN
000018  *
000019  * MOTOR-UP TIMES PER COMMAND
000020  T50MS           EQU         $02                 ; 50MS FOR MONTIMEH
000021  T200MS          EQU         $08                 ;200 MS FOR MONTIMEH
000022  T1SEC           EQU         $27                 ;1-SEC FOR MONTIMEH
000023  *
000024  MTIMES          DFB         T200MS,T1SEC,T50MS  ;READ,WRITE,SENSE
000025  *
000026                  REP         40
000027  * DRIVE TABLES:
000028  *
000029  DRIVESEL        DS          4                   ;NONZERO IF SELECTED
000030  *
000031  UPTIME          DS          4                   ;MOTOR RUNTIME SINCE STARTED
000032  DRVTRACK        DS          4                   ;CURRENT HEAD POSITION
000033                  PAGE
000034                  DO          REV0ROM             ;ONLY IF SUPPORTING IT!
000035  * JUMP TABLE TO MONITOR ROUTINES.
000036  *  THIS TABLE FILLED IN BY 'INIT'.
000037  *
000038  JMPTAB          EQU         *
000039  RDADR           JMP         *
000040  READ            JMP         *
000041  WRITE           JMP         *
000042  SEEK            JMP         *
000043  MSWAIT          JMP         *
000044  PRENIB          JMP         *
000045  POSTNIB         JMP         *
000046  *
000047  REV0            EQU         *                   ;REV0 ADDRESSES
000048                  JMP         $F1BD               ;RDADR
000049                  JMP         $F148               ;READ
000050                  JMP         $F219               ;WRITE
000051                  JMP         $F400               ;SEEK
000052                  JMP         $F456               ;MSWAIT
000053                  JMP         $F2C6               ;PRENIB
000054                  JMP         $F311               ;POSTNIB
000055  VSIZE           EQU         *-REV0              ;TABLE SIZE
000056  *
000057  REV1            EQU         *                   ;REV1 ADDRESSES
000058                  JMP         $F1B9               ;RDADR
000059                  JMP         $F148               ;READ
000060                  JMP         $F216               ;WRITE
000061                  JMP         $F400               ;SEEK
000062                  JMP         $F456               ;MSWAIT
000063                  JMP         $F2C4               ;PRENIB
000064                  JMP         $F30F               ;POSTNIB
000065                  ELSE                            ;FOR REV1 WE USE EQUATES
000066  RDADR           EQU         $F1B9               ;RDADR
000067  READ            EQU         $F148               ;READ
000068  WRITE           EQU         $F216               ;WRITE
000069  SEEK            EQU         $F400               ;SEEK
000070  MSWAIT          EQU         $F456               ;MSWAIT
000071  PRENIB          EQU         $F2C4               ;PRENIB
000072  POSTNIB         EQU         $F30F               ;POSTNIB
000073                  FIN
000074
000075  ZZEND           EQU         *
000076  ZZLEN           EQU         *-ZZORG
```

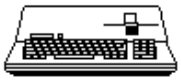


```
000077                  IFNE       ZZLEN-LENDISK3
000078                  FAIL       2,"SOSORG           FILE IS INCORRECT FOR DISK3"
000079                  FIN
000080
000081  ********************************************************************
000082  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.DATA.SRC
000083  ********************************************************************
000084
```

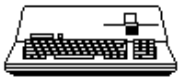
End of File -- Lines: 84 Characters: 2763



```
================================================================================================
FILE: "SOS.DISK3.MAIN.SRC.TEXT"
================================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.MAIN.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007  * MAIN ENTRY POINT:
000008  *
000009  * DISABLE NMI/RESET AND ENABLE ROM/IO SPACE
000010  *
000011  MAIN            EQU     *
000012                  LDA     E.REG               ;SAVE CALLER'S
000013                  AND     #$FF-$20            ;DROP SCREEN BIT
000014                  STA     ESAVE               ; ENVIRONMENT
000015                  DO      1-TEST              ;NO RESETLOCK FOR TESTING
000016                  LDA     E.REG               ;GET EREG AGAIN
000017                  AND     #$FF-$10            ;DISABLE NMI/RESET
000018                  FIN
000019                  ORA     #$03                ;ENABLE ROM/IO SPACE
000020                  STA     E.REG
000021  *
000022                  LDA     NOSCROLL            ;DISABLE SMOOTHSCROLL
000023  *
000024                  PHP                         ;IF ALREADY SEI'D, THEN WE
000025                  PLA                         ; STAY THAT WAY...
000026                  ROR     A
000027                  ROR     A
000028                  ROR     A
000029                  ROR     A
000030                  STA     IRQMASK             ;'I' BIT INTO BIT7
000031  *
000032  * MAKE SURE WE HAVE A VALID COMMAND:
000033  *
000034                  LDA     D.COMMAND           ;GET IT
000035                  BMI     BADCMD              ;=>WOW!
000036                  BEQ     IOSETUP             ;=>ZERO IS A READ
000037                  CMP     #10                 ;OFF THE END?
000038                  BCS     BADCMD              ;=>YES
000039                  CMP     #9                  ;REPEAT?
000040                  BNE     CMD1                ;=>NOPE
000041  *
000042  * REPEAT. SIMPLY GET PRIOR COMMAND:
000043  *
000044                  LDA     PREVUNIT            ;IS THIS REPEAT FOR
000045                  CMP     D.UNITNUM           ; SAME UNIT?
000046                  BNE     BADOP               ;=>NO? ILLEGAL!
000047                  LDA     PREVCMD             ;YES, SET COMMAND
000048                  BEQ     RPTOK               ;=>REPEAT'ED READ IS OK
000049                  CMP     #1                  ;IF NOT, IS IT REPEAT'ED WRITE?
000050                  BNE     BADOP               ;=>CAN'T REPEAT OTHER COMMANDS
000051  RPTOK           EQU     *
000052                  STA     D.COMMAND           ;SAME AS BEFORE
000053                  CMP     #0                  ;READ?
000054                  BEQ     IOSETUP             ;=>YES
000055  * NOW REPEAT GOES LIKE OTHERS:
000056  *
000057  *
000058  CMD1            EQU     *
000059                  CMP     #1                  ;WRITE?
000060                  BNE     CMD2                ;=>NOPE
000061                  JMP     IOSETUP             ;=>YES
000062  CMD2            EQU     *
000063                  CMP     #2                  ;STATUS?
000064                  BNE     CMD3                ;=>NOT STATUS
000065                  LDA     D.STATCODE          ;IS IT 'SENSE'?
000066                  BEQ     GOSTAT              ;=>YES
000067                  LDA     #XCTLCODE           ;ILLEGAL CODE
000068                  JMP     EXIT
000069  GOSTAT          EQU     *
000070                  JMP     DRVSETUP            ;=>YES
000071  *
000072  CMD3            EQU     *
000073                  CMP     #8                  ;INIT?
000074                  BNE     BADOP               ;=>NOPE
000075                  JMP     INIT                ;=>YES, DO INIT
000076  *
```

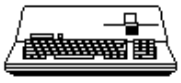


```
000077  BADOP           EQU     *
000078                  LDA     #XBADOP         ;ILLEGAL COMMAND
000079                  JMP     EXIT            ;BACK TO YOU
000080  *
000081  BADCMD          EQU     *
000082                  LDA     #XREQCODE       ;INVALID COMMAND
000083                  JMP     EXIT            ;BACK TO YOU
000084                  PAGE
000085  * SETUP WHAT WE HAVE TO BEFORE
000086  *  PERFORMING THE I/O OPERATION:
000087  *
000088  IOSETUP         EQU     *
000089                  LDA     D.BLOCK+1       ;VALIDATE BLOCKNUM
000090                  BEQ     CHKBYTE         ;=> IF <256, IT'S OK
000091                  CMP     #2              ;IS IT <512?
000092                  BCS     BADBLOCK        ;=>BAD BOY!
000093                  LDA     D.BLOCK         ;YES, CHECK LO HALF
000094                  CMP     #280-256        ; FOR RANGE
000095                  BCC     CHKBYTE         ;=>IT'S OK
000096  BADBLOCK        EQU     *
000097                  LDA     #XBLKNUM        ;BAD BLOCK NUMBER
000098                  JMP     EXIT            ;RETURN BAD NEWS
000099  *
000100  CHKBYTE         EQU     *
000101                  LDA     D.BYTES         ;GET LO COUNT
000102                  BNE     BADCOUNT        ;=>ERR, NOT INTEGRAL BLOCK(S)
000103                  LDA     D.BYTES+1       ;GET HI COUNT
000104                  LSR     A               ;MAKE BLOCK COUNT
000105                  BCS     BADCOUNT        ;=>BAD IF HALF-BLOCK COUNT
000106                  STA     BLKCOUNT        ;SAVE COUNT OF BLOCKS
000107  *
000108  * DOES REQUESTED BYTECOUNT CAUSE US
000109  *  TO RUN OFF END OF DISK?
000110  *
000111                  LDA     BLKCOUNT        ;NO. ADD STARTBLOCK
000112                  CLC                     ; AND BLKCOUNT AND SEE
000113                  ADC     D.BLOCK         ;  IF WE'RE TOO BIG
000114                  LDX     D.BLOCK+1       ;DID IT START OUT > 255?
000115                  BNE     BLKG255         ;=>YES
000116                  BCC     DRVSETUP        ;=>DEFINITELY < 256
000117                  BCS     CHKLO           ;=>IF CARRY,THEN >256
000118  BLKG255         EQU     *
000119                  BCS     BADCOUNT        ;>255+CARRY IS NOW >511
000120  CHKLO           EQU     *
000121                  CMP     #280-256+1      ;281..511 ?
000122                  BCC     DRVSETUP        ;=>NO, WE ARE OK
000123  BADCOUNT        EQU     *
000124                  LDA     #XBYTECNT       ;ILLEGAL BYTECOUNT
000125                  JMP     EXIT            ;SORRY...
000126                  PAGE
000127  *
000128  * SELECT THE APPROPRIATE DRIVE:
000129  *
000130  DRVSETUP        EQU     *
000131                  LDA     D.COMMAND       ;SAVE THIS COMMAND
000132                  STA     PREVCMD         ; AND DEVICE FOR
000133                  LDA     D.UNITNUM       ;  SUBSEQUENT
000134                  STA     PREVUNIT        ;   'REPEAT' CALL
000135                  LDA     E.REG           ;DOWNSHIFT TO
000136                  ORA     #$80            ; 1MHZ FOR REMAINDER
000137                  STA     E.REG           ;  OF DRIVER EXECUTION
000138                  JSR     UNITSEL         ;SELECT & START IT
000139  *
000140  * SEE IF THE MOTOR STARTED. IF NOT,
000141  *  THEN IT'S EITHER DISKSWITCH OR NODRIVE.
000142  *
000143                  JSR     CHKDRV          ;MOTOR RUNNING?
000144                  BNE     DOIO            ;=>YES, GREAT.
000145  *
000146  * IF WE GET A MOTOR WHEN WE MOVE
000147  *  THE HEAD, THEN IT'S DISKSWITCH.
000148  *
000149                  LDX     D.UNITNUM       ;FORCE HEAD MOTION
000150                  INC     DRVTRACK,X      ; EVEN IF ALREADY ON ZERO
000151                  INC     DRVTRACK,X      ;GIVE HIM A FIRM KNOCKER
000152                  LDA     #0              ;SEEK TO TRACK ZERO
000153                  JSR     MYSEEK          ; FOR BFM DIR READ
000154                  JSR     CHKDRV          ;RUNNING NOW?
000155                  BNE     DSWITCH         ;=>YES, A SWITCHEROO
000156                  LDA     #0
000157                  LDY     D.UNITNUM       ;FORGET THAT THIS
```

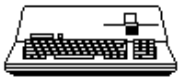


```
000158                  STA         DRIVESEL,Y          ; DRIVE WAS 'SELECTED'
000159                  LDA         #XNODRIVE           ;NO, A MISSING DRIVE!
000160                  JMP         EXIT
000161  *
000162  DSWITCH         EQU         *
000163                  LDA         #XDISKSW            ;USER PULLED A FAST ONE
000164                  JMP         EXIT                ; BUT HE CAN'T FOOL US.
000165                  PAGE
000166  * PREPARE TO DO THE OPERATION:
000167  *
000168  DOIO            EQU         *
000169                  LDA         D.BUFL              ;COPY USER BUFFER
000170                  STA         BUFTEMP             ; AND BLOCK NUMBER
000171                  LDA         D.BUFH              ;  TO OUR WORKSPACE
000172                  STA         BUFTEMP+1
000173                  LDA         $1400+D.BUFH
000174                  STA         $1400+BUFTEMP+1
000175                  LDA         D.BLOCK
000176                  STA         BLKTEMP
000177                  LDA         D.BLOCK+1
000178                  STA         BLKTEMP+1
000179  *
000180  * IF CALLER GAVE US A COUNT OF ZERO BYTES,
000181  *  THEN WE'RE ALL DONE!
000182  *
000183                  LDA         D.COMMAND           ;IS IT STATUS?
000184                  CMP         #2                  ;IF SO, THEN BYTECOUNT
000185                  BNE         DOIO2               ; IS MEANINGLESS
000186                  JMP         STATUS
000187  DOIO2           EQU         *
000188                  LDY         BLKCOUNT            ;BLKS=0?
000189                  BEQ         READOK              ;=>YES, YOU GET GOOD RETURN
000190                  CMP         #0                  ;READ COMMAND?
000191                  BEQ         READREQ             ;=>YES
000192                  JMP         WRITEREQ
000193                  PAGE
000194                  REP         40
000195  *  -- READ --
000196                  REP         40
000197  READREQ         EQU         *
000198                  LDA         #0                  ;CLEAR COUNT OF
000199                  LDY         #0
000200                  STA         (D.BYTRD),Y         ; BYTES READ
000201                  INY
000202                  STA         (D.BYTRD),Y
000203  READREQ2        EQU         *
000204                  JSR         BLK2SECT            ;COMPUTE TRK/SECTOR THIS BLOCK
000205  *
000206                  JSR         SECTORIO            ;READ IT PLEASE
000207                  BCS         READERR             ;=>WE LOSE.
000208                  INC         SECTOR              ;BUMP TO NEXT
000209                  INC         SECTOR              ; LOGICAL SECTOR
000210                  INC         BUF+1               ;BUMP SECTOR BUFFER
000211                  JSR         SECTORIO            ;READ IT TOO
000212                  BCS         READERR             ;=>WE LOSE.
000213                  LDY         #1
000214                  LDA         (D.BYTRD),Y         ;BUMP COUNT OF
000215                  CLC
000216                  ADC         #2
000217                  STA         (D.BYTRD),Y         ; BYTES READ
000218  *
000219  * MORE BLOCKS TO GO?
000220  *
000221                  JSR         MOREBLKS            ;SETUP FOR NEXT BLOCK
000222                  BNE         READREQ2            ;=>MORE TO READ...
000223  READOK          EQU         *
000224                  LDA         #0                  ;GOOD RETURN
000225                  JMP         EXIT                ;TELL HAPPY USER
000226  *
000227  READERR         EQU         *
000228                  JMP         EXIT                ;RETURN ERROR CODE
000229                  CHN         DISK3.WRT.SRC
000230
000231  ********************************************************************************
000232  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.MAIN.SRC
000233  ********************************************************************************
000234
```

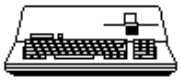
End of File -- Lines: 234 Characters: 7883



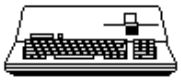
```
================================================================================================
FILE: "SOS.DISK3.SIO.SRC.TEXT"
================================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SIO.SRC
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP         40
000008  * NAME    : SECTORIO
000009  * FUNCTION: READ OR WRITE A SECTOR
000010  * INPUT   : IBSTRK, IBSECT, MONTIME,
000011  * RETURNS : CARRY CLEAR IF OK (AC=00)
000012  *         : CARRY SET   IF ERROR (AC=ERRCODE)
000013  *         : SEEKWAIT  ALL SETUP
000014  * DESTROYS: ALL REGISTERS
000015                  REP         40
000016  *
000017  SECTORIO        EQU         *
000018                  LDA         #R.RECAL            ;SETUP THE
000019  * R.RECAL MUST BE NON-ZERO!! (SEE BELOW)
000020                  STA         RECALCNT            ; RECAL TRIES
000021                  NOP                             ; PAD ONE BYTE
000022                  STA         E1908               ; A-REG MUST BE NON-ZERO !!!
000023  * E1908 = NON-ZERO LOCKOUT MOUSE
000024  *
000025                  LDY         D.UNITNUM           ;ARE WE ON-TRACK?
000026                  LDA         TRACK
000027                  CMP         DRVTRACK,Y
000028                  BEQ         SOUGHT              ;=>IF SO, FORGET SEEK & DELAY!
000029  *
000030  * WAIT BEFORE STEPPING:
000031  *
000032                  LDA         SEEKWAIT            ;SEEK DELAY NEEDED?
000033                  BEQ         GOSEEK              ;=>NAW...
000034                  LDA         #0
000035                  STA         SEEKWAIT            ;CLEAR THE FLAG
000036                  LDA         #4                  ;ADD SEEKDELAY TO
000037                  JSR         ADDTIME             ; THE TOTAL UPTIME(S)
000038                  TAY                             ;4*25 MS DELAY
000039  SEEKDEL         EQU         *
000040                  LDA         #0
000041                  JSR         MSWAIT
000042                  DEY
000043                  BNE         SEEKDEL
000044  *
000045  * ISSUE THE SEEK:
000046  *
000047  GOSEEK          EQU         *
000048                  LDA         TRACK               ;GET DESTINATION TRACK
000049                  JSR         MYSEEK              ;=>..AND YOU SHALL FIND...
000050  *
000051  SOUGHT          EQU         *
000052                  LDA         IRQMASK             ;SET IRQ MASK FOR
000053                  STA         IMASK               ; CORE ROUTINES
000054                  LDA         #R.IRQ              ;SETUP IRQ RETRIES
000055                  STA         INTRTRY
000056                  LDA         #R.IOERR            ; AND ERROR RETRIES
000057                  STA         RETRYCNT
000058  *
000059  * DELAY FOR ANY REMAINING MOTOR-UP TIME:
000060  *
000061  MDELAY          EQU         *
000062                  LDA         MONTIMEH            ;ANY TIME REMAINING?
000063                  BPL         FINDIT              ;=>NO, WE'RE UP TO SPEED.
000064                  LDA         #1                  ;YES, SO BUMP A SLICE OF
000065                  JSR         ADDTIME             ; UPTIME WHILE WE WAIT
000066                  LDA         #0
000067                  JSR         MSWAIT
000068                  JMP         MDELAY              ;=>GO TILL ENOUGH
000069  *
000070  * FIND THE DESIRED SECTOR:
000071  *
000072  * NOTE: FINDSECT RETURNS WITH
000073  *       IRQ INHIBITED!
000074  *
000075  FINDIT          EQU         *
000076                  PHP                             ;INHIBIT IRQ WHILE
```



```
000077                  SEI                             ; MESSING WITH VBL FLAGS
000078                  LDA     E.IER                   ;DISABLE VBL IRQ
000079                  AND     #$18                    ; DURING SECTOR I/O
000080                  STA     E.IER
000081                  ORA     #$80                    ;FOR 'SET' LATER
000082                  STA     VBLSAVE
000083                  PLP                             ;RESTORE IRQ STATUS
000084                  JSR     FINDSECT                ;FIND ME PLEASE
000085                  BCS     TRYRECAL                ;=>NO? RECAL OR GIVE UP!
000086                  LDX     #$60                    ;SET UP SLOT FOR CORE RTNS
000087                  LDA     D.COMMAND               ;WHAT'S YOUR PLEASURE?
000088                  BNE     SIOWRITE                ;=>WRITE
000089  *
000090                  REP     40
000091  * READ A SECTOR:
000092  *
000093                  JSR     READ                    ;READ THAT SECTOR
000094                  JSR     FIXIRQ                  ;ENABLE IRQ IF OK
000095                  LDA     VBLSAVE                 ;ALLOW VBL DURING
000096                  STA     E.IER                   ; POSTNIB
000097                  BCS     BADIO                   ;=>I/O ERR OR IRQ
000098                  LDA     E.REG                   ;SET 2MHZ FOR POSTNIB
000099                  AND     #$7F
000100                  STA     E.REG
000101                  JSR     POSTNIB                 ;POSTNIB/CHECKSUM
000102                  BCS     IORETRY                 ;=>I/O ERR:BAD CHKSUM
000103                  JMP     SIOGOOD                 ;=>GOOD READ
000104  *
000105                  REP     40
000106  * WRITE A SECTOR:
000107  *
000108  SIOWRITE        EQU     *
000109                  JSR     WRITE                   ;WRITE THE DATA
000110                  JSR     FIXIRQ                  ;RE-ENABLE IRQ IF OK
000111                  LDA     VBLSAVE                 ;RESTORE
000112                  STA     E.IER                   ; VBL IRQ
000113                  BCC     SIOGOOD                 ;=>GOOD WRITE
000114                  BVC     SIOWPROT                ;=>WRITE PROTECTED
000115  *
000116                  REP     40
000117  * IT DIDN'T GO WELL FOR US:
000118  *
000119  BADIO           EQU     *
000120                  DO      1-REV0ROM               ;FOR REV1
000121                  BVS     FINDIT                  ;=>IRQ. JUST RETRY IT.
000122                  ELSE                            ;FOR REV0
000123  *
000124  * THE REV1 ROM TAKES CARE OF THE
000125  *  IRQ RETRY COUNT, BUT REV0 DOESN'T:
000126  *
000127                  BVC     IORETRY                 ;=>I/O ERROR. RETRY IT
000128                  LDA     ROMREV                  ;WHICH ROM?
000129                  BNE     FINDIT                  ;=>REV1. HE DOES IT.
000130                  LDA     INTRTRY                 ;REV0. OUT OF RETRIES?
000131                  BPL     BADIO2                  ;=>NO.
000132                  STA     IMASK                   ;SET HI BIT FOR IRQ MASK
000133  BADIO2          EQU     *
000134                  DEC     INTRTRY                 ;ONE LESS RETRY
000135                  JMP     FINDIT                  ;=>RETRY AFTER IRQ
000136                  FIN
000137  *
000138  * RETRY AFTER AN I/O ERROR:
000139  *
000140  IORETRY         EQU     *
000141                  DEC     RETRYCNT                ;ANY RETRIES LEFT?
000142                  BNE     FINDIT                  ;=>YEAH, RETRY AFTER ERROR
000143  *
000144  * RETRIES EXHAUSTED. RECALIBRATE:
000145  *
000146  TRYRECAL        EQU     *
000147                  LDA     VBLSAVE                 ;ALLOW VBL IF RECAL
000148                  STA     E.IER                   ; OR UNRECOVERABLE ERROR
000149                  DEC     RECALCNT                ;HAVE WE RECALIBRATED YET?
000150                  BMI     SIOERR                  ;=>YUP. WE'RE DEAD.
000151                  JSR     RECAL                   ;NO, TRY OUR LUCK
000152                  LDY     D.UNITNUM               ;ARE WE ON-TRACK?
000153                  LDA     TRACK
000154                  CMP     DRVTRACK,Y
000155                  BNE     NOTSAME
000156                  JMP     SOUGHT                  ;=>IF SO, FORGET RESEEK
000157  NOTSAME         EQU     *
```

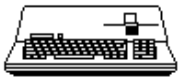


```
000158                  JMP         GOSEEK              ;TRY AGAIN ON TARGET TRACK
000159  *
000160                  REP         40
000161  SIOERR          EQU         *
000162                  LDA         #XIOERROR           ;RETURN CODE
000163                  SEC                             ;INDICATE HARD ERROR
000164                  BCS         SIORET
000165  SIOWPROT        EQU         *
000166                  LDA         #XNOWRITE           ;RETURN CODE
000167                  SEC                             ;INDICATE HARD ERROR
000168                  BCS         SIORET
000169  SIOGOOD         EQU         *
000170                  LDA         #0
000171                  CLC                             ;INDICATE GOOD COMPLETION
000172  SIORET          LDX         #0                  ; SAY OK TO MOUSE
000173                  STX         E1908               ; WITH THIS GLOBAL $1908
000174                  RTS
000175                  PAGE
000176                  REP         40
000177  * NAME    : FINDSECT
000178  * FUNCTION: LOCATE A DESIRED SECTOR
000179  * INPUT   : IBTRK, IBSECT SETUP
000180  * RETURNS : CARRY CLEAR IF OK,
000181  *         : CARRY SET   IF ERROR.
000182  * DESTROYS: ALL REGISTERS & 'TEMP'
000183  * NOTE    : RETURNS WITH IRQ DISABLED IF NO ERROR!
000184                  REP         40
000185  *
000186  FINDSECT        EQU         *
000187                  LDA         #R.FIND*16          ;SETUP NUMBER OF REVS
000188                  STA         RETRYADR            ; ALLOWED TO FIND SECTOR
000189                  LSR         TEMP                ;COMPUTE LATENCY FIRST TIME THRU
000190  FINDSEC2        EQU         *
000191                  LDX         #$60                ;FAKE SLOT FOR CORE ROUTINES
000192                  JSR         RDADR               ;GET NEXT ADDRESS FIELD
000193                  BCS         RDADERR             ;=>UGH! AN ERROR!
000194  *
000195  * MAKE SURE WE'RE ON THE CORRECT TRACK:
000196  *
000197                  LDA         TRACK               ;IS IT
000198                  CMP         CSSTV+2             ; CORRECT TRACK?
000199                  BNE         FINDERR             ;=>NO?!? IT'S USELESS!
000200                  LDA         SECTOR              ;IS IT
000201                  CMP         CSSTV+1             ; DESIRED SECTOR?
000202                  BEQ         FINDGOOD            ;=>YEAH. GOT IT!
000203  *
000204  * COMPUTE LATENCY. EACH TWO-SECTOR
000205  *  DISTANCE IS 25 MS OF UPTIME.
000206  *
000207                  LDA         TEMP                ;LATENCY ALREADY COMPUTED?
000208                  BMI         RDADERR             ;=>YES.
000209                  LDA         SECTOR              ;HOW FAR AWAY IS OUR
000210                  SEC                             ; DESIRED SECTOR?
000211                  ROR         TEMP                ;PREVENT RECOMPUTATION
000212                  SBC         CSSTV+1
000213                  AND         #$0F
000214                  LSR         A                   ;EACH 2-SECTORS IS 25 MS
000215                  JSR         ADDTIME
000216  *
000217  * KEEP LOOKING TILL WE FIND IT:
000218  *
000219  RDADERR         EQU         *
000220                  JSR         FIXIRQ              ;ENABLE IRQ IF APPROPRIATE
000221                  DEC         RETRYADR            ;ANY RETRIES LEFT?
000222                  BEQ         FINDERR             ;=>NO, WE CAN'T FIND IT.
000223  *
000224  * COMPENSATE FOR A BUG IN RDADR: IF WE TRY
000225  *  TO CALL RDADR AGAIN BEFORE THE DATA MARK
000226  *  GOES BY, THEN RDADR WILL ACCIDENTALLY CALL
000227  *  THAT AN ERROR. WE CAN AVOID THIS 'FAKE'
000228  *  ERROR BY DELAYING PAST THE DATA MARK.
000229                  LDY         #200                ;1 MS IS PLENTY
000230  ADRDELAY        EQU         *
000231                  DEY
000232                  BNE         ADRDELAY
000233                  JMP         FINDSEC2            ;=>NOW TRY LOOKING AGAIN
000234  *
000235                  REP         40
000236  FINDGOOD        EQU         *
000237                  LDA         #0                  ;CLEAR VOLNUM OUT OF
000238                  STA         MONTIMEH            ; MOTORTIME!
```

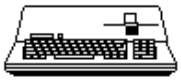


```
000239                  CLC                             ;INDICATE NO ERROR
000240                  RTS
000241  *
000242  FINDERR         EQU         *
000243                  JSR         FIXIRQ              ;ENABLE IRQ IF APPROPRIATE
000244                  LDA         #0                  ;CLEAR VOLNUM OUT OF
000245                  STA         MONTIMEH            ; MOTORTIME!
000246                  SEC                             ;INDICATE THE ERROR
000247                  RTS
000248
000249                  CHN         DISK3.USEL.SRC
000250
000251  ********************************************************************************
000252  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SIO.SRC
000253  ********************************************************************************
000254
000255
```

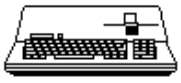
End of File -- Lines: 255 Characters: 8245



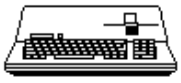
```
================================================================================
FILE: "SOS.DISK3.SRC.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SRC
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL        'SOS 1.1  DISK /// DRIVER'
000007  TEST            EQU         0                   ;FOR FUNNY-MODE TESTING
000008                  INCLUDE     SOSORG,6,1,254
000009                  DO          TEST
000010                  ORG         $2000
000011                  ELSE
000012                  REL
000013                  ORG         ORGDISK3
000014                  FIN
000015  ZZORG           EQU         *
000016                  CHR         '-'
000017                  MSB         OFF
000018  *
000019                  REP         40
000020  *    COPYRIGHT (C) APPLE COMPUTER INC.
000021  *          ALL RIGHTS RESERVED
000022                  REP         40
000023  *
000024  REV0ROM         EQU         0                   ;1=SUPPORT REV0 ROM
000025  *
000026                  DO          1-TEST
000027                  ENTRY       DIB1                ;DIB1
000028                  ENTRY       DIB2                ;DIB2
000029                  ENTRY       DIB3                ;DIB3
000030                  ENTRY       DIB4                ;DIB4
000031                  ENTRY       SEEKDSK3            ;SEEK CURRENT DRIVE
000032  *
000033                  EXTRN       SYSERR
000034  *
000035                  EXTRN       XREQCODE
000036                  EXTRN       XBADOP
000037                  EXTRN       XNODRIVE
000038                  EXTRN       XIOERROR
000039                  EXTRN       XNOWRITE
000040                  EXTRN       XBYTECNT
000041                  EXTRN       XBLKNUM
000042                  EXTRN       XDISKSW
000043                  EXTRN       XCTLCODE
000044  *
000045                  EXTRN       E1908               ; GLOBAL FLAG FOR MOUSE DRIVER
000046  * TO SAY WE CANNOT BE INTERRUPTED
000047  *
000048                  ELSE
000049  XREQCODE        EQU         $20
000050  XBADOP          EQU         $26
000051  XNODRIVE        EQU         $28
000052  XIOERROR        EQU         $27
000053  XNOWRITE        EQU         $2B
000054  XBYTECNT        EQU         $2C
000055  XBLKNUM         EQU         $2D
000056  XDISKSW         EQU         $2E
000057  XCTLCODE        EQU         $21
000058                  FIN
000059                  PAGE
000060  * DISK /// CONTROLLER EQUATES:
000061  *
000062  *       MOTOR SELECT BITS:
000063  *
000064  *    DRIVE    INT   EXT1   EXT2
000065  *    -----    ---   ----   ----
000066  *     .D1      1     X      X
000067  *     .D2      X     0      1
000068  *     .D3      X     1      0
000069  *     .D4      X     1      1
000070  *
000071  MS.INT          EQU         $C0D4               ;MOTOR   SELECT:INTERNAL DRIVE
000072  MD.INT          EQU         $C0D5               ;MOTOR DESELECT:INTERNAL DRIVE
000073  *
000074  MS.EXT1         EQU         $C0D3               ;MOTOR   SELECT:EXTERNAL DRIVE
000075  MS.EXT2         EQU         $C0D1               ;MOTOR   SELECT:EXTERNAL DRIVE
000076  MD.EXT1         EQU         $C0D2               ;MOTOR DESELECT:EXTERNAL DRIVE
```



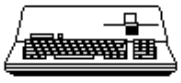
```
000077  MD.EXT2         EQU        $C0D0               ;MOTOR DESELECT:EXTERNAL DRIVE
000078  *
000079  IS.INT          EQU        $C0EA               ;I/O SELECT:INTERNAL DRIVE
000080  IS.EXT          EQU        $C0EB               ;I/O SELECT:EXTERNAL DRIVE
000081  *
000082  NOSCROLL        EQU        $C0D8               ;SMOOTHSCROLL OFF
000083  *
000084  MOTOROFF        EQU        $C0E8               ;MOTOR(S) START POWEROFF T/O
000085  MOTORON         EQU        $C0E9               ;MOTOR(S) POWER ON
000086  Q6L             EQU        $C08C               ;Q7L,Q6L=READ
000087  Q6H             EQU        $C08D               ;Q7L,Q6H=SENSE WPROT
000088  Q7L             EQU        $C08E               ;Q7H,Q6L=WRITE
000089  Q7H             EQU        $C08F               ;Q7H,Q6H=WRITE STORE
000090  *
000091  * OTHER EQUATES:
000092  *
000093  E.REG           EQU        $FFDF               ;ENVIRONMENT REGISTER
000094  E.IER           EQU        $FFEE               ;INTERRUPT ENABLE REGISTER
000095  *
000096  * RETRY COUNTERS:
000097  *
000098  R.RECAL         EQU        1                   ;MAX RECALIBRATES
000099  * R.RECAL MUST NOT BECOME ZERO! (MOUSE WILL BE LOCKED OUT)
000100  * SEE DISK3.SIO.SRC LINE 14 FOR DETAIL
000101  R.FIND          EQU        3                   ;MAX REVS TO FIND A SECTOR
000102  R.IOERR         EQU        4                   ;MAX RETRIES ON READ ERROR
000103  R.IRQ           EQU        6                   ;MAX IRQ'S TOLERATED BEFORE SEI
000104                  PAGE
000105  * ZPAGE EQUATES FOR CORE ROUTINES:
000106  *
000107                  DSECT
000108                  ORG        $81
000109  IBSLOT          DS         1                   ;SLOT=$60 FOR RTNS
000110                  DS         7                   ;N/A
000111                  DS         1                   ;RDADR:CHECKSUM
000112                  DS         1                   ;N/A
000113  IMASK           DS         1                   ;BIT7 SET IF IRQ ALLOWED
000114  CURTRK          DS         1                   ;SEEK:CURRENT TRACK
000115                  DS         2                   ;N/A
000116  INTRTRY         DS         1                   ;READ: IRQ RETRY COUNT
000117                  DS         5                   ;N/A
000118                  DS         1                   ;RDADR:'MUST FIND' COUNT
000119                  DS         1                   ;READ,WRITE: CHECKSUM
000120  CSSTV           DS         4                   ;RDADR:CKSUM,SEC,TRK,VOL
000121  MONTIMEL        EQU        CSSTV+2             ;MSWAIT:MOTOR-ON TIME
000122  MONTIMEH        EQU        MONTIMEL+1
000123  BUF             DS         2                   ;PRENIB,POSTNIB:USER BUFFER
000124                  DS         1                   ;SEEK:PRIOR PHASE
000125  TRKN            DS         1                   ;SEEK:TARGET TRACK
000126  *
000127  * LOCAL TEMPS:
000128  *
000129                  ORG        $D0                 ;WE'RE ALLOWED TO $FF
000130  BLKTEMP         DS         2                   ;LOCAL TEMP FOR BLKNUMBER
000131  BUFTEMP         DS         2                   ;LOCAL TEMP FOR BUFFER ADDRESS
000132  TRACK           DS         1                   ;LOCAL TEMP FOR TRACK
000133  SECTOR          DS         1                   ;LOCAL TEMP FOR SECTOR
000134  RETRYADR        DS         1                   ;LOCAL TEMP FOR SECTOR-FIND RETRIES
000135  RETRYCNT        DS         1                   ;LOCAL TEMP FOR I/O RETRIES
000136  RECALCNT        DS         1                   ;LOCAL TEMP FOR RECAL COUNT
000137  BLKCOUNT        DS         1                   ;BLKS REQD TO SATISFY BYTECOUNT
000138  SEEKWAIT        DS         1                   ;<>0 IF SEEK DELAY NEEDED
000139  IRQMASK         DS         1                   ;ENTRY 'I' BIT
000140  TEMP            DS         1                   ;JUST A TEMP
000141                  DEND
000142                  PAGE
000143  * DRIVER INTERFACE AREA:
000144  *
000145                  DSECT
000146                  ORG        $C0
000147  D.COMMAND       DS         1                   ;COMMAND CODE
000148  D.UNITNUM       DS         1                   ;UNIT NUMBER
000149  D.BUFL          DS         2                   ;BUFFER ADDRESS
000150  D.BUFH          EQU        D.BUFL+1
000151  D.STATCODE      EQU        D.BUFL              ;DSTATUS CODE
000152  D.STATBUF       EQU        D.BUFH              ;^DSTATUS LIST
000153  D.BYTES         DS         2                   ;BYTECOUNT
000154  D.BLOCK         DS         2                   ;REQUESTED BLOCKNUM
000155  D.BYTRD         DS         2                   ;BYTES READ (READ)
000156                  DS         6                   ;SPARES (OK AS TEMPS)
000157                  DEND
```



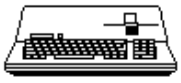
```
000158                  PAGE
000159  DIB1            EQU     *                   ;DIB FOR .D1
000160                  DW      DIB2                ;FLINK
000161                  DW      MAIN                ;ENTRY POINT
000162                  DFB     3                   ;NAME LENGTH
000163                  ASC     '.D1          '
000164                  DFB     $80                 ;DEVNUM: ACTIVE
000165                  DFB     0                   ;SLOT
000166                  DFB     0                   ;UNIT NUMBER
000167                  DFB     $E1,1,0             ;TYPE,SUB,FILLER
000168                  DW      280                 ;BLOCKCOUNT
000169                  DW      1                   ;MANUFACTURER=APPLE
000170                  DW      $1100               ;VERSION=1.1
000171  *
000172  DIB2            EQU     *                   ;DIB FOR .D2
000173                  DW      DIB3                ;FLINK
000174                  DW      MAIN                ;ENTRY POINT
000175                  DFB     3                   ;NAME LENGTH
000176                  ASC     '.D2          '
000177                  DFB     $80                 ;DEVNUM: ACTIVE
000178                  DFB     0                   ;SLOT
000179                  DFB     1                   ;UNIT NUMBER
000180                  DFB     $E1,1,0             ;TYPE,SUB,FILLER
000181                  DW      280                 ;BLOCKCOUNT
000182                  DW      1                   ;MANUFACTURER=APPLE
000183                  DW      $1100               ;VERSION=1.1
000184  *
000185  DIB3            EQU     *                   ;DIB FOR .D3
000186                  DW      DIB4                ;FLINK
000187                  DW      MAIN                ;ENTRY POINT
000188                  DFB     3                   ;NAME LENGTH
000189                  ASC     '.D3          '
000190                  DFB     $80                 ;DEVNUM: ACTIVE
000191                  DFB     0                   ;SLOT
000192                  DFB     2                   ;UNIT NUMBER
000193                  DFB     $E1,1,0             ;TYPE,SUB,FILLER
000194                  DW      280                 ;BLOCKCOUNT
000195                  DW      1                   ;MANUFACTURER=APPLE
000196                  DW      $1100               ;VERSION=1.1
000197  *
000198  DIB4            EQU     *                   ;DIB FOR .D4
000199                  DW      0                   ;NO FLINK
000200                  DW      MAIN                ;ENTRY POINT
000201                  DFB     3                   ;NAME LENGTH
000202                  ASC     '.D4          '
000203                  DFB     $80                 ;DEVNUM: ACTIVE
000204                  DFB     0                   ;SLOT
000205                  DFB     3                   ;UNIT NUMBER
000206                  DFB     $E1,1,0             ;TYPE,SUB,FILLER
000207                  DW      280                 ;BLOCKCOUNT
000208                  DW      1                   ;MANUFACTURER=APPLE
000209                  DW      $1100               ;VERSION=1.1
000210                  DW      1                   ;MANUFACTURER=APPLE
000211                  DW      $1100               ;VERSION=1.1
000212
000213                  CHN     DISK3.MAIN.SRC
000214                  INCLUDE SOSORG,6,1,254
000215
000216  ********************************************************************************
000217  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SRC
000218  ********************************************************************************
000219
000220

End of File -- Lines: 220 Characters: 7516
```



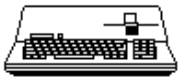
```
================================================================================
FILE: "SOS.DISK3.SUBS.SRC.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SUBS.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP         40
000008  * NAME    : CHKDRV
000009  * FUNCTION: CHECK IF MOTOR(S) RUNNING
000010  * INPUT   : NONE
000011  * RETURNS : 'BNE' IF RUNNING
000012  *         : 'BEQ' IF NOT
000013  * DESTROYS: AC,X
000014                  REP         40
000015  * NOTES: DUE TO A FLOATING PIN, THERE
000016  *  COULD BE A GLITCH WHICH CAUSES THE
000017  *  SHIFTER TO 'FLASH' ONTO THE BUS
000018  *  INSTEAD OF ALWAYS BEING TRISTATED.
000019  *  THIS COULD CAUSE CHKDRV TO THINK
000020  *  THAT THE MOTOR IS SPINNING WHEN IT
000021  *  IS NOT. THUS WE WILL SAMPLE THE SHIFTER
000022  *  FOR 40 US AT 6-US INTERVALS. IF, AFTER
000023  *  THREE (3) CONSECUTIVE PASSES, ANY OF
000024  *  THE PASSES SEES A 'LOCKED' SHIFTER,
000025  *  THEN WE SAY THE DRIVE IS STOPPED.
000026  *
000027  *
000028  CHKDRV          EQU         *
000029                  LDX         #3                  ;CHECK SHIFTER SEVERAL TIMES
000030  CHKD1           EQU         *
000031                  LDA         Q6L+$60             ;GET DATA
000032                  CMP         Q6L+$60             ;HAS IT CHANGED?
000033                  BNE         CHANGED             ;=>YES
000034                  CMP         Q6L+$60             ;HAS IT CHANGED?
000035                  BNE         CHANGED             ;=>YES
000036                  CMP         Q6L+$60             ;HAS IT CHANGED?
000037                  BNE         CHANGED             ;=>YES
000038                  CMP         Q6L+$60             ;HAS IT CHANGED?
000039                  BNE         CHANGED             ;=>YES
000040                  CMP         Q6L+$60             ;HAS IT CHANGED?
000041                  BNE         CHANGED             ;=>YES
000042                  CMP         Q6L+$60             ;HAS IT CHANGED?
000043                  BNE         CHANGED             ;=>YES
000044                  CMP         Q6L+$60             ;HAS IT CHANGED?
000045                  BNE         CHANGED             ;=>YES
000046                  RTS                             ;IF EVER LOCKED, IT'S STOPPED
000047  *
000048  CHANGED         EQU         *
000049                  DEX
000050                  BNE         CHKD1               ;TRY SEVERAL TIMES
000051                  DEX                             ;SET CC=BNE
000052                  RTS                             ;RETURN ZFLAG APPROPRIATELY
000053                  PAGE
000054                  REP         40
000055  * NAME    : ADDTIME
000056  * FUNCTION: ADD TO MOTOR UPTIME(S)
000057  * INPUT   : AC=NO. OF 25 MS INCREMENTS
000058  * DESTROYS: Y
000059                  REP         40
000060  *
000061  ADDTIME         EQU         *
000062                  PHA                             ;PRESERVE AC
000063                  LDY         #4                  ;TABLE INDEX/COUNT
000064  ADD2            EQU         *
000065                  LDA         DRIVESEL-1,Y        ;IS IT SELECTED?
000066                  BEQ         ADD3                ;=>NOPE
000067                  PLA
000068                  PHA                             ;RECOVER DELTA-T
000069                  CLC
000070                  ADC         UPTIME-1,Y          ;ADD TO MOTOR UPTIME
000071                  CMP         #T1SEC+2            ;IS IT AT MAX TIME?
000072                  BCC         ADD2A               ;=>NO, STORE NEW TIME
000073                  LDA         #T1SEC+1            ;YES, SET TO >1 SEC
000074  ADD2A           EQU         *
000075                  STA         UPTIME-1,Y
000076  ADD3            EQU         *
```



```
000077                  DEY
000078                  BNE         ADD2                ;=>DO ALL 4 DRIVES
000079  *
000080                  PLA                             ;RESTORE AC
000081                  RTS
000082                  PAGE
000083                  REP         40
000084  * NAME    : RECAL
000085  * FUNCTION: RECALIBRATE DRIVE HEAD
000086  * INPUT   : NONE
000087  * DESTROYS: ALL REGISTERS
000088  * NOTE    : A 'QUIET' RECALIBRATE IS DONE
000089  *         : USING TWO ITERATIONS. IF WE ARE
000090  *         : LOST, THEN SEEK 48-TRACKS
000091  *         : TOWARD TRACK ZERO. IF WE KNOW
000092  *         : WHAT TRACK WE'RE CURRENTLY
000093  *         : ON (+- 1/2 TRACK), THEN JUST
000094  *         : ADD A LITTLE EXTRA AND SEEK
000095  *         : TO TRACK ZERO. A 48-TRACK
000096  *         : SEEK WILL ALWAYS GET US BACK
000097  *         : ONTO THE MEDIA, EVEN IF WE
000098  *         : WERE "OFF THE CAM". FROM THAT
000099  *         : POINT, THE 2ND SEEK GETS US
000100  *         : BACK TO TRACK ZERO QUIETLY.
000101                  REP         40
000102  *
000103  RECAL           EQU         *
000104                  LDA         #2                  ;TWO ITERATIONS, PLEASE
000105  RECAL1          EQU         *
000106                  PHA                             ;SAVE LOOPCOUNT
000107                  LDX         #$60                ;SETUP SLOT FOR CORE RTNS
000108                  JSR         RDADR               ;WHERE ARE WE?
000109                  BCC         RECAL2              ;=>NOW WE KNOW
000110                  JSR         RDADR               ;GIVE SECOND SHOT
000111                  BCC         RECAL2              ;=>THAT GOT IT
000112                  LDA         #48                 ;LOST? TRY 48-TRACK SEEK
000113                  JMP         RECAL3
000114  RECAL2          EQU         *
000115                  LDA         CSSTV+2             ;HERE'S WHERE WE ARE
000116                  CLC                             ;ADD SOME SO WE GET A
000117                  ADC         #3                  ; HARDER SEEK TO ZERO
000118  RECAL3          EQU         *
000119                  LDY         D.UNITNUM           ;THIS IS NOW WHERE
000120                  STA         DRVTRACK,Y          ; WE ARE
000121                  JSR         FIXIRQ              ;ENABLE IRQ IF OK
000122  *
000123                  LDA         #0                  ;DESTINATION TRACK IS 00
000124                  STA         MONTIMEH            ;CLEAR MOTOR-UP TIME SO
000125                  STA         MONTIMEL            ; SEEK KNOWS HOW LONG RECAL TAKES
000126                  JSR         MYSEEK              ;=>SLAM IT BACK!
000127                  PLA                             ;HAVE WE DONE IT TWICE?
000128                  TAY
000129                  DEY
000130                  TYA
000131                  BNE         RECAL1              ;=>DO TWO ITERATIONS
000132                  RTS
000133                  PAGE
000134                  REP         40
000135  * NAME    : SEEKDSK3
000136  * FUNCTION: SEEK CURRENT DRIVE
000137  * INPUT   : AC=DESTINATION TRACK
000138  * OUTPUT  : NONE
000139  * DESTROYS: ALL REGISTERS
000140  * NOTE    : MUST BE CALLED WHILE
000141  *         :  MOTOR IS RUNNING, IN
000142  *         :  1MHZ+ROM+IO MODE
000143                  REP         40
000144  SEEKDSK3        EQU         *
000145                  LDY         PREVUNIT            ;GET DRIVENUM
000146                  STY         D.UNITNUM           ;SET IT UP
000147                  JSR         MYSEEK              ;MOVE IT!
000148                  RTS
000149                  REP         40
000150  * NAME    : MYSEEK
000151  * FUNCTION: SEEK TO DESIRED TRACK
000152  * INPUT   : AC=DESTINATION TRACK
000153  * DESTROYS: ALL REGISTERS
000154                  REP         40
000155  MYSEEK          EQU         *
000156                  STA         TRKN                ;TEMP HOLD OF AC
000157                  LDY         D.UNITNUM           ;GET DRIVENUM
```

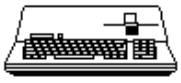


```
000158                  LDA         DRVTRACK,Y          ;SETUP CURRENT TRACK
000159                  ASL         A                   ;SET IN HALFTRACKS FOR SEEK
000160                  STA         CURTRK              ; FOR SEEK ROUTINE
000161                  LDX         #$60                ;SET UP SLOT FOR CORE RTNS
000162                  LDA         MONTIMEH            ;GET STARTING MOTOR TIME
000163                  STA         TEMP
000164  *
000165  * NOTE: IRQ'S WHICH SUSPEND SEEK MAY CAUSE A
000166  *  SEEK FAILURE. WE WILL HAVE TO RECALIBRATE
000167  *  SINCE WE WON'T BE ON-TRACK. WE CAN NOT GET
000168  *  ON A HALFTRACK SINCE SEEK ALLOWS SETTLING
000169  *  TIME OF THE PHASE. BECAUSE VBL IS A SERIOUS
000170  *  OFFENDER, WE INHIBIT HIM.
000171  *
000172                  PHP                             ;INHIBIT IRQ WHILE
000173                  SEI                             ; MESSING WITH VBL FLAGS
000174                  LDA         E.IER
000175                  AND         #$18
000176                  STA         VBLSAVE
000177                  STA         E.IER
000178                  PLP                             ;RESTORE IRQ STATUS
000179                  LDA         TRKN                ;RESTORE DESTINATION TRACK
000180                  STA         DRVTRACK,Y          ;DEST IS NOW CURRENT
000181                  ASL         A                   ;MAKE IT IN HALFTRACKS
000182                  JSR         SEEK                ;GO MOVE THE HEAD...
000183                  LDA         VBLSAVE             ;NOW ALLOW THAT
000184                  ORA         #$80                ; NASTY
000185                  STA         E.IER               ;  VBL INTERRUPT
000186  *
000187  * COMPUTE THE TIME USED BY SEEK:
000188  *
000189                  LDA         MONTIMEH            ;INCLUDE SEEKTIME IN
000190                  SEC
000191                  SBC         TEMP
000192                  JSR         ADDTIME             ; TOTAL MOTOR UPTIME(S)
000193                  RTS
000194                  PAGE
000195                  REP         40
000196  * NAME    : BLK2SECT
000197  * FUNCTION: COMPUTE TRACK/SECTOR FOR A BLOCK
000198  *           AND ADJUST BUFFER ADDRESS
000199  * INPUT   : D.BLOCK, D.BUF
000200  * OUTPUT  : TRACK, SECTOR, D.BUF
000201  * DESTROYS: AC,Y
000202                  REP         40
000203  *
000204  BLK2SECT        EQU         *
000205                  LDA         BLKTEMP+1           ;GET HI BLK HALF
000206                  ROR         A                   ;MOVE LO BIT TO CARRY
000207                  LDA         BLKTEMP             ;GET LO HALF
000208                  ROR         A                   ;COMBINE WITH HI BIT
000209                  LSR         A
000210                  LSR         A                   ;FINISH OFF DIVIDE-BY-8
000211                  STA         TRACK               ;THAT'S THE TRACK
000212                  LDA         BLKTEMP             ;GET LO HALF AGAIN
000213                  AND         #7
000214                  TAY
000215                  LDA         SECTABLE,Y          ;GET START SECTOR
000216                  STA         SECTOR
000217  *
000218  * ADJUST BUFFER ADDRESS SO THAT I/O
000219  *  WON'T WRAPAROUND IN THE BANK:
000220  * (THIS ALGORITHM RIPPED OFF FROM 1.0)
000221  *
000222                  LDA         BUFTEMP+1           ;GET BUFFER HI ADDRESS
000223                  LDY         $1400+BUFTEMP+1     ; AND XTND BYTE
000224                  CMP         #$82                ;IF RAM ADDR >=8200 THEN BUMP TO
000225                  BCC         NOADJ               ; NEXT BANK PAIR
000226                  CPY         #$80
000227                  BCC         NOADJ               ;=>NOT USING BANKPAIR
000228                  CPY         #$8F                ;SPECIAL BANK 0?
000229                  BEQ         NOADJ               ;=>YES
000230                  AND         #$7F                ;DROP HI ADDRESS AND
000231                  STA         BUFTEMP+1           ; BUMP BANK NUMBER
000232                  INC         $1400+BUFTEMP+1
000233  *
000234  NOADJ           EQU         *
000235                  LDA         BUFTEMP+1           ;COPY BUFFER ADDRESS
000236                  STA         BUF+1               ; FOR PRE & POSTNIB
000237                  LDA         BUFTEMP
000238                  STA         BUF
```

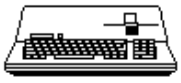


```
000239                  LDA         $1400+BUFTEMP+1
000240                  STA         $1400+BUF+1
000241                  RTS
000242  *
000243  SECTABLE        DFB         $00,$04,$08,$0C,$01,$05,$09,$0D
000244                  PAGE
000245                  REP         40
000246  * NAME    : MOREBLKS
000247  * FUNCTION: SETUP TO DO NEXT BLOCK
000248  * INPUT   : NONE
000249  * RETURNS : 'BNE' IF MORE TO DO
000250  *         : 'BEQ' IF NO MORE TO DO
000251  * DESTROYS:NOTHING
000252                  REP         40
000253  *
000254  MOREBLKS        EQU         *
000255                  INC         BUFTEMP+1           ;BUMP BUFFER ADDRESS
000256                  INC         BUFTEMP+1
000257                  INC         BLKTEMP             ;BUMP BLOCK NUMBER
000258                  BNE         MORE2
000259                  INC         BLKTEMP+1
000260  MORE2           EQU         *
000261                  DEC         BLKCOUNT            ;MORE BLOCKS TO GO?
000262                  RTS                             ;RETURN RESULT OF DEC
000263                  SKP         4
000264                  REP         40
000265  * NAME    : FIXIRQ
000266  * FUNCTION: ENABLE IRQ IF APPROPRIATE
000267  * INPUT   : NONE
000268  * DESTROYS: NOTHING
000269                  REP         40
000270  *
000271  FIXIRQ          EQU         *
000272                  PHA
000273                  LDA         IRQMASK             ;SHOULD IRQ BE ENABLED?
000274                  BMI         FIXRET              ;=>NO, LEAVE IT ALONE
000275                  CLI                             ;ENABLE IRQ
000276  FIXRET          EQU         *
000277                  PLA
000278                  RTS
000279
000280                  CHN         DISK3.DATA.SRC
000281
000282  ********************************************************************************
000283  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SUBS.SRC
000284  ********************************************************************************
000285
000286
```

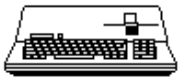
End of File -- Lines: 286 Characters: 9045



```
================================================================================
FILE: "SOS.DISK3.USEL.SRC.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.USEL.SRC
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP         40
000008  * NAME    : UNITSEL
000009  * FUNCTION: SELECT & START A DRIVE,
000010  *           SET UP MOTOR & SEEK DELAYS
000011  * INPUT   : NONE
000012  * OUTPUT  : MONTIME,SEEKTIME
000013  * DESTROYS: ALL REGISTERS
000014                  REP         40
000015  *
000016  UNITSEL         EQU         *
000017                  LDY         D.UNITNUM           ;GET DRIVENUM
000018                  LDA         #0                  ;ASSUME NO SEEKWAIT
000019                  STA         SEEKWAIT            ; WILL BE NEEDED
000020                  STA         MONTIMEL            ;CLEAR MONTIME
000021                  STA         MONTIMEH
000022  *
000023  * SEE IF MOTOR(S) STILL SPINNING:
000024  *
000025                  JSR         CHKDRV              ;MOTOR(S) POWERED UP?
000026                  BNE         SPINNING            ;=>YES. WHO IS IT?
000027  *
000028  * NO MOTOR(S) SPINNING. DESELECT
000029  *  ALL MOTORS AND START AFRESH:
000030  *
000031                  LDX         MD.INT              ;DESELECT ALL
000032                  LDA         #0                  ;SHOW INTERNAL AS
000033                  STA         DRIVESEL+0          ; NOT SELECTED
000034                  STA         UPTIME+0            ;INDICATE DRIVE IS FULLY STOPPED
000035                  JSR         EXTDESEL            ;DESELECT ALL EXTERNALS TOO
000036                  JMP         SETTIME             ;GO SETUP MOTOR DELAY
000037                  REP         40
000038  * MOTOR(S) SPINNING: OURS?
000039  *
000040  SPINNING        EQU         *
000041                  LDA         DRIVESEL,Y          ;HAD WE BEEN SELECTED?
000042                  BNE         GOFORIT             ;=>YES, GO FOR IT RIGHT AWAY.
000043  *
000044  * WE AREN'T SPINNING. SHUTDOWN ANOTHER
000045  *  DRIVE, IF NECESSARY, TO GET GOING:
000046  *
000047                  CPY         #0                  ;ARE WE THE INTERNAL DRIVE?
000048                  BEQ         SETTIME             ;=>YES, LEAVE EXT MOTOR ALONE
000049  *
000050  * WE'RE AN EXTERNAL DRIVE. STOP ALL EXTERNAL MOTORS
000051  *  UNCONDITIONALLY, BUT LEAVE THE INTERNAL MOTOR ALONE.
000052  * IF WE *DID* HAVE TO STOP ANOTHER EXTERNAL, THEN
000053  *  MAKE SURE WE SET THE CORRECT PRE-SEEK DELAY!
000054  *
000055                  LDA         #0                  ;SEE IF ANOTHER EXTERNAL
000056                  ORA         DRIVESEL+3          ; HAD BEEN
000057                  ORA         DRIVESEL+2          ;  SELECTED
000058                  ORA         DRIVESEL+1          ;   BEFORE...
000059                  BEQ         SETTIME             ;=>NO, SEEK DELAY IS UNNECESSARY
000060                  INC         SEEKWAIT            ;YES, DELAY BEFORE STEPPING
000061                  JSR         EXTDESEL            ;DESELECT ALL EXTERNALS
000062                  JMP         SETTIME             ;=>GO SETUP MOTOR DELAY
000063                  PAGE
000064                  REP         40
000065  * OUR DRIVE IS SPINNING. GO FOR IT!
000066  * DEPENDING OF HOW LONG THE MOTOR'S BEEN ON,
000067  *  THIS COMMAND MAY REQUIRE A MOTOR DELAY.
000068  *
000069  GOFORIT         EQU         *
000070                  LDX         D.COMMAND           ;GET CURRENT COMMAND
000071                  LDA         MTIMES,X            ;GET REQUIRED UPTIME FOR IT
000072                  SEC
000073                  SBC         UPTIME,Y            ;DRIVE RUNNING LONG ENOUGH?
000074                  BCS         SELECT              ;=>NO, AC NOW HAS DELTA-T
000075                  LDA         #0                  ;OTHERWISE, WAIT=0
000076                  JMP         SELECT              ;SET MONTIME & SELECT DRIVE
```

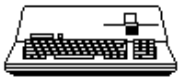


```
000077                  REP     40
000078  *
000079  * ALL MOTORS WERE OFF. CHOOSE THE
000080  *  APPROPRIATE MOTOR-ON TIME:
000081  *
000082  SETTIME         EQU     *
000083                  LDA     #0              ;INDICATE THAT
000084                  STA     UPTIME,Y        ; THE DRIVE WAS OFF
000085                  LDX     D.COMMAND       ;GET CURRENT COMMAND
000086                  LDA     MTIMES,X        ;GET CORRECT DELAY TIME
000087                  REP     40
000088  *
000089  * SELECT THE DRIVE & START IT:
000090  *
000091  SELECT          EQU     *
000092                  STA     MONTIMEH        ;NEGATE IT BECAUSE
000093                  LDA     #0              ; IT GETS INCREMENTED
000094                  SEC                     ;  INSTEAD OF
000095                  SBC     MONTIMEH        ;   DECREMENTED
000096                  STA     MONTIMEH        ;STUFF MOTOR DELAY
000097                  CPY     #1              ;ARE WE THE INTERNAL DRIVE?
000098                  BCS     SELEXT          ;=>NO, AN EXTERNAL
000099                  LDA     IS.INT          ;I/O SELECT INTERNAL
000100                  LDA     MS.INT          ;MOTOR SELECT INTERNAL
000101                  JMP     UNITRET         ;=>ALL DONE!
000102  *
000103  SELEXT          EQU     *
000104                  LDA     IS.EXT          ;I/O SELECT EXTERNAL
000105                  CPY     #2              ;ARE WE 2, 3, OR 4 ?
000106                  BCS     NOTD2           ;=>DEFINITELY 3 OR 4
000107                  LDA     MD.EXT1         ;MOTOR SELECT
000108                  LDA     MS.EXT2         ; ONLY .D2
000109                  JMP     UNITRET         ;=>ALL DONE!
000110  *
000111  NOTD2           EQU     *
000112                  BNE     ISD4            ;=>DEFINITELY NOT 3
000113                  LDA     MS.EXT1         ;MOTOR SELECT
000114                  LDA     MD.EXT2         ; ONLY .D3
000115                  JMP     UNITRET         ;=>ALL DONE!
000116  *
000117  ISD4            EQU     *
000118                  LDA     MS.EXT1         ;MOTOR SELECT
000119                  LDA     MS.EXT2         ; ONLY .D4
000120  *
000121  *
000122  UNITRET         EQU     *
000123                  LDA     MOTORON         ;PROVIDE MOTOR POWER
000124                  LDA     #1              ;SAY WE'VE SELECTED
000125                  STA     DRIVESEL,Y      ; THIS DRIVE
000126  *
000127  * IF WE HAVE MOTORTIME TO BURN,
000128  *  THEN DELAY 50 MS. THIS ENSURES
000129  *  A GOOD SOLID CHKDRV AFTER
000130  *  TURNING ON THE MOTOR.
000131  *
000132                  LDA     MONTIMEH        ;ANY MOTORTIME?
000133                  BPL     UNITRTS         ;=>NO, WE GO FOR IT.
000134                  LDY     #5              ;5*10 MS
000135  UNITDEL         EQU     *
000136                  LDA     #100            ;100*100US IS 10MS
000137                  JSR     MSWAIT
000138                  DEY
000139                  BNE     UNITDEL
000140                  LDA     #2              ;INCLUDE THE 50MS
000141                  JSR     ADDTIME         ; IN MOTOR UPTIME(S)
000142  UNITRTS         EQU     *
000143                  RTS
000144                  SKP     5
000145                  REP     40
000146  * NAME    : EXTDESEL
000147  * FUNCTION: DESELECT ALL EXTERNAL DRIVE MOTORS
000148  * INPUT   : NONE
000149  * DESTROYS: AC,X
000150                  REP     40
000151  *
000152  EXTDESEL        EQU     *
000153                  LDA     MD.EXT1         ;DESELECT ALL EXTERNAL
000154                  LDA     MD.EXT2         ; DRIVE MOTORS
000155                  LDX     #3              ;SHOW THAT THEY ARE
000156                  LDA     #0              ; ARE ALL DEAD DUCKS
000157  EDS1            STA     DRIVESEL,X
```

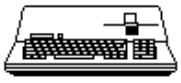


```
000158                  STA         UPTIME,X            ;DRIVE MOTORS ARE OFF
000159                  DEX
000160                  BNE         EDS1
000161                  RTS
000162
000163                  CHN         DISK3.SUBS.SRC
000164
000165  ********************************************************************
000166  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.USEL.SRC
000167  ********************************************************************
000168
```

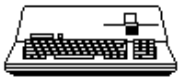
End of File -- Lines: 168 Characters: 5663



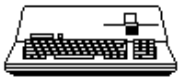
```
================================================================================
FILE: "SOS.DISK3.WRT.SRC.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.WRT.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP         40
000008  * --- WRITE ---
000009                  REP         40
000010  *
000011  WRITEREQ        EQU         *
000012                  JSR         BLK2SECT            ;COMPUTE TRK/SECTOR THIS BLOCK
000013                  LDA         E.REG               ;SET 2 MHZ
000014                  AND         #$7F
000015                  STA         E.REG
000016                  JSR         PRENIB              ;PRENIBBLIZE FOR WRITE
000017                  JSR         SECTORIO            ;WRITE IT OUT...
000018                  BCS         WRITERR             ;=>SOMETHING'S WRONG
000019  *
000020                  INC         SECTOR              ;BUMP TO NEXT
000021                  INC         SECTOR              ; LOGICAL SECTOR
000022                  INC         BUF+1               ;BUMP SECTOR BUFFER ADDRESS
000023                  LDA         E.REG               ;SET 2 MHZ
000024                  AND         #$7F
000025                  STA         E.REG
000026                  JSR         PRENIB              ;PRENIBBLIZE FOR WRITE
000027                  JSR         SECTORIO            ;WRITE IT OUT
000028                  BCS         WRITERR             ;=>SOMETHING'S WRONG
000029  *
000030  * MORE BYTES TO DO?
000031  *
000032                  JSR         MOREBLKS            ;SETUP FOR NEXT
000033                  BNE         WRITEREQ            ;=>MORE TO DO
000034                  LDA         #0                  ;GOOD RETURN
000035                  JMP         EXIT
000036  *
000037  WRITERR         EQU         *
000038                  JMP         EXIT                ;RETURN ERROR CODE
000039                  PAGE
000040                  REP         40
000041  *  --- STATUS ---
000042                  REP         40
000043  *
000044  STATUS          EQU         *
000045                  LDX         #$60                ;DUMMY SLOT
000046                  LDA         Q6H,X               ;SENSE WRITE PROTECT
000047                  LDA         Q7L,X
000048                  ASL         A                   ;PRESERVE IT IN CARRY
000049                  LDA         Q6L,X               ;BACK TO READ MODE
000050                  LDA         #0                  ;NOW MOVE BIT TO
000051                  ROL         A                   ; PROPER POSITION
000052                  ROL         A                   ; ($02)
000053                  LDY         #0
000054                  STA         (D.STATBUF),Y       ;RETURN IT
000055                  LDA         #0                  ;GOOD RETURN
000056                  JMP         EXIT                ;DONE
000057                  PAGE
000058                  REP         40
000059  * --- INIT ---
000060                  REP         40
000061  *
000062  INIT            EQU         *
000063                  LDA         INITFLAG            ;INIT'ED YET?
000064                  BMI         GOODINIT            ;=>YES, DONE
000065  *
000066                  LDA         #$60                ;SETUP SLOT FOR
000067                  STA         IBSLOT              ; CORE ROUTINES
000068                  LDA         #$FF                ;PREVENT SECOND
000069                  STA         INITFLAG            ; INIT
000070                  LDA         #0                  ;CLEAR STUFF OUT
000071                  STA         PREVUNIT            ;SOSBOOT JUST USED .D1
000072                  LDY         #4
000073  CLRDRVS         EQU         *
000074                  LDA         #0
000075                  STA         DRIVESEL-1,Y        ;NOBODY SELECTED
000076                  STA         UPTIME-1,Y          ;ALL OFF
```



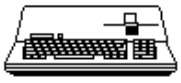
```
000077                  STA         DRVTRACK-1,Y
000078                  DEY
000079                  BNE         CLRDRVS
000080                  DO          1-TEST              ;ONLY IF NOT TESTING
000081  *
000082  * SET UP .D1 SINCE LOADER'S USING IT:
000083  *
000084                  LDA         E.REG               ;SET 1MHZ FOR THE
000085                  ORA         #$80                ; STATEMACHINE I/O
000086                  STA         E.REG
000087                  JSR         CHKDRV              ;IS .D1 MOTOR SPINNING?
000088                  BEQ         INIT2               ;=>NO, MOTOR'S OFF
000089                  LDA         #T200MS             ;UPTIME GOOD FOR READS
000090                  STA         UPTIME+0
000091  INIT2           EQU         *
000092                  LDA         #1
000093                  STA         DRIVESEL+0          ;.D1 IS THE CURRENT DRIVE
000094                  LDA         $0300+CURTRK        ;RETRIEVE CURRENT TRACK
000095                  STA         DRVTRACK+0          ;REMEMBER IT
000096                  FIN
000097  *
000098  * SET UP JMP TABLE FOR CORRECT ROM:
000099  *
000100                  DO          REV0ROM             ;ONLY IF SUPPORTING IT!
000101                  LDA         $F1B9               ;LOOK FOR START OF RDADR
000102                  CMP         #$A0                ;IS IT RDADR (REV1)?
000103                  BEQ         INITREV1            ;=>YES
000104                  CMP         #$60                ;IS IT END OF READ (REV0)?
000105                  BNE         INITERR             ;=>NEITHER!
000106                  LDY         #0                  ;REV=0
000107                  BEQ         INITVECT            ;(ALWAYS TAKEN)
000108  INITREV1        EQU         *
000109                  LDY         #VSIZE
000110  INITVECT        EQU         *
000111                  STY         ROMREV              ;SET ROM REVISION INDICATOR
000112                  LDX         #VSIZE
000113  MOVEVECT        EQU         *
000114                  LDA         REV0,Y              ;GET A BYTE
000115                  STA         JMPTAB,Y            ;MOVE IT
000116                  INY
000117                  DEX
000118                  BNE         MOVEVECT
000119                  FIN
000120  GOODINIT        EQU         *
000121                  LDA         #0                  ;RETCODE=GOOD, IF YOU CARE
000122                  CLC                             ;SAY 'GOOD INIT'
000123                  BCC         EXIT                ;(ALWAYS TAKEN)
000124                  DO          REV0ROM
000125  INITERR         EQU         *
000126                  SEC                             ;SAY 'BAD INIT'
000127  * FALL THRU TO EXIT
000128                  FIN
000129                  PAGE
000130                  REP         40
000131  * -- EXIT PATH --
000132                  REP         40
000133  *
000134  EXIT            EQU         *
000135                  PHA                             ;SAVE RETURN CODE
000136  *
000137  * UPDATE UPTIME BY 50 MS (3 SECTOR-TIMES)
000138  *  TO ACCOUNT FOR READ/WRITE TIME:
000139  *
000140                  LDA         D.COMMAND           ;GET COMMAND
000141                  CMP         #2                  ;SENSE OR INIT?
000142                  BCS         EXIT2               ;=>YES, NO TIME USED UP
000143                  LDA         #2                  ;TIME=50 MS (2 UNITS)
000144                  JSR         ADDTIME             ;BUMP UPTIME(S)
000145  *
000146  * RESTORE CALLER ENVIRONMENT:
000147  *
000148  EXIT2           EQU         *
000149                  LDA         E.REG               ;GET CURRENT STATE
000150                  AND         #$20                ; OF THE SCREEN
000151                  ORA         ESAVE               ;MERGE WITH CALLER STATE
000152                  STA         E.REG
000153                  JSR         FIXIRQ              ;RE-ENABLE IRQ IF OK
000154                  LDA         MOTOROFF            ;START MOTOR-OFF TIMEOUT
000155                  PLA                             ;RESTORE RETURN CODE
000156                  DO          TEST                ;IF TEST, NO SYSERR
000157                  RTS
```



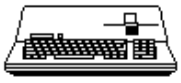
```
000158                  ELSE
000159                  BNE         GOERR               ;=>ERROR RETURN VIA SYSERR
000160                  CLC
000161                  RTS                             ;GOOD RETURN W/CARRY CLEAR
000162  GOERR           EQU         *
000163                  JSR         SYSERR              ;RETURN VIA SYSERR
000164                  FIN
000165
000166                  CHN         DISK3.SIO.SRC
000167
000168  ********************************************************************************
000169  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.WRT.SRC
000170  ********************************************************************************

End of File -- Lines: 170 Characters: 5441
```



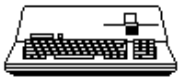
```
================================================================================
FILE: "SOS.EQUATES.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: EQUATES
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  *
000007                  ENTRY      BFMGR
000008  *
000009  * BFM INITIALIZATION ENTRIES
000010  * (INIT CODE FOUND IN INIT.SRC)
000011  *
000012                  ENTRY      BFMFCB1             ; FCB PAGE 1 ADDR
000013                  ENTRY      BFMFCB2             ; AND PAGE 2
000014                  ENTRY      FCBZPP
000015                  ENTRY      SISTER
000016                  ENTRY      PATHBUF
000017                  ENTRY      VCB
000018                  ENTRY      WORKSPC
000019                  ENTRY      PFIXPTR
000020                  ENTRY      BMAPAGE
000021                  ENTRY      BMBPAGE
000022                  ENTRY      FCBADDRH
000023                  ENTRY      BMAMADR
000024                  ENTRY      BMBMADR
000025  *
000026  *
000027                  EXTRN      LEVEL               ; FILE LEVEL (LOW BYTE)
000028                  EXTRN      OPMSGRPLY           ; OPERATOR MESSAGE
000029                  EXTRN      DATETIME            ; THANKS TOM...
000030                  EXTRN      DMGR                ; THANKS BOB...
000031                  EXTRN      REQBUF              ;   "
000032                  EXTRN      REQFXBUF            ;   "
000033                  EXTRN      GETBUFADR           ;  "
000034                  EXTRN      RELBUF              ;   "
000035                  EXTRN      BLKDLST             ;   "
000036                  EXTRN      SERR
000037                  EXTRN      BACKMASK
000038  *
000039  * ERRORS
000040  *
000041                  EXTRN      SYSERR
000042  *
000043                  EXTRN      BADPATH             ; INVALID PATHNAME SYNTAX
000044                  EXTRN      FCBFULL             ; FILE CONTROL BLOCK FULL
000045                  EXTRN      BADREFNUM           ; INVALID REFNUM
000046                  EXTRN      PATHNOTFND          ; PATHNAME NOT FOUND
000047                  EXTRN      VNFERR              ; VOLUME NOT FOUND
000048                  EXTRN      FNFERR              ; FILE NOT FOUND
000049                  EXTRN      DUPERR              ; DUPLICATE FILE NAME ERROR
000050                  EXTRN      DUPVOL              ; DUPLICATE VOLUME CAN'T BE LOGGED IN.
000051                  EXTRN      OVRERR              ; NOT ENOUGH DISK SPACE FOR PREALLOCATION
000052                  EXTRN      DIRFULL             ; DIRECTORY FULL ERROR
000053                  EXTRN      CPTERR              ; FILE INCOMPATIBLE SOS VERSION
000054                  EXTRN      TYPERR              ; NOT CURRENTLY SUPPORTED FILE TYPE
000055                  EXTRN      EOFERR              ; POSITION ATTEMPTED BEYOND END OF FILE
000056                  EXTRN      POSNERR             ; ILLEGAL POSITION (L.T. 0 OR G.T. $FFFFFF)
000057                  EXTRN      ACCSERR             ; FILE ACCESS R/W REQUEST CONFLICTS WITH ATTRIBUTES.
000058                  EXTRN      BTSERR              ; USER SUPPLIED BUFFER TOO SMALL
000059                  EXTRN      FILBUSY             ; EITHER WRITE WAS REQUESTED OR WRITE ACCESS ALREADY ALLOCATED.
000060                  EXTRN      NOTSOS              ; NOT A SOS DISKETTE
000061                  EXTRN      BADLSTCNT           ; INVALID VALUE IN LIST PARAMETER
000062                  EXTRN      XDISKSW             ; DISK SWITCHED
000063                  EXTRN      NOTBLKDEV           ; NOT A BLOCK DEVICE
000064                  EXTRN      XNOWRITE            ; DISK/MEDIA IS HARDWARE WRITE PROTECTED
000065                  EXTRN      XIOERROR            ; INFORMATION ON BLOCK DEVICE NOT ACCESSABLE
000066                  EXTRN      DIRERR              ; DIRECTORY ENTRY COUNT INCONSISTENT WITH ACTUAL ENTRIES
000067                  EXTRN      BITMAPADR           ; BIT MAP DISK ADDRESS IMPOSSIBLE
000068  *
000069  * FATAL ERRORS
000070  *
000071                  EXTRN      SYSDEATH
000072  *
000073                  EXTRN      VCBERR              ; VOLUME CONTROL BLOCK NOT USABLE
000074                  EXTRN      ALCERR              ; ALLOCATION BLOCKS INVALID
000075                  EXTRN      TOOLONG             ; PATHNAME BUFFER OVERFLOW
000076                  PAGE
```



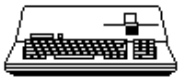
```
000077  *
000078  * CONSTANTS
000079  *
000080  DLIMIT          EQU        $2F                 ; DELIMITER IS CURRENTLY AN ASCII '/'
000081  SEEDTYP         EQU        1
000082  SAPTYP          EQU        2
000083  TRETYP          EQU        3
000084  DIRTYP          EQU        $D
000085  HEDTYP          EQU        $E
000086  RDCMD           EQU        $0
000087  WRTCMD          EQU        $1
000088  RPTCMD          EQU        $9
000089  STATCMD         EQU        $02                 ; REQUEST STATUS OF BLOCK DEVICE. (BIT 0 = WRITE PROTECTED)
000090  STATSUB         EQU        $0
000091  PRETIME         EQU        $20                 ; COMMAND NEEDS CURRENT DATE/TIME STAMP
000092  PREREF          EQU        $40                 ; COMMAND REQUIRES FCB ADDRESS AND VERIFICATION
000093  PREPATH         EQU        $80                 ; COMMAND HAS PATHNAME TO PREPROCESS
000094  SISTER          EQU        $1400
000095  *
000096  * VOLUME STATUS CONSTANTS (BITS)
000097  *
000098  DSWITCH         EQU        $40                 ; FOR DISK SWITCHED ERROR RECOVERY.
000099  *
000100  * FILE STATUS CONSTANTS
000101  *
000102  DATALC          EQU        $1                  ; DATA BLOCK NOT ALLOCATED.
000103  IDXALC          EQU        $2                  ; INDEX NOT ALLOCATED
000104  TOPALC          EQU        $4                  ; TOP INDEX NOT ALLOCATED
000105  STPMOD          EQU        $8                  ; STORAGE TYPE MODIFIED
000106  USEMOD          EQU        $10                 ; FILE USAGE MODIFIED
000107  EOFMOD          EQU        $20                 ; END OF FILE MODIFIED
000108  DATMOD          EQU        $40                 ; DATA BLOCK MODIFIED
000109  IDXMOD          EQU        $80                 ; INDEX BLOCK MODIFIED
000110  FCBMOD          EQU        $80                 ; HAS FCB/DIRECTORY BEEN MODIFIED? (FLUSH)
000111  *
000112  * FILE ATTRIBUTES CONSTANTS
000113  *
000114  READEN          EQU        $1                  ; READ ENABLED
000115  WRITEN          EQU        $2                  ; WRITE ENABLED
000116  NLINEN          EQU        $10                 ; NEW LINE ENABLED
000117  BKBITVAL        EQU        $20                 ; FILE NEEDS BACKUP IF SET (BKBITFLG)
000118  RENAMEN         EQU        $40                 ; RENAME OK WHEN ON.
000119  DSTROYEN        EQU        $80                 ; DESTROY OK WHEN ON.
000120                  PAGE
000121  * HEADER INDEX CONSTANTS
000122  *
000123  HNLEN           EQU        $0                  ; HEADER NAME LENGTH (OFFSET INTO HEADER)
000124  *HNAME EQU $1 ; HEADER NAME
000125  HPENAB          EQU        $10                 ; PASSWORD ENABLE BYTE
000126  HPASS           EQU        $11                 ; ENCODED PASSWORD
000127  HCRDT           EQU        $18                 ; HEADER CREATION DATE
000128  * HCRTM EQU $1A ; HEADER CREATION TIME
000129  HVER            EQU        $1C                 ; SOS VERSION THAT CREATED DIRECTORY
000130  HCMP            EQU        $1D                 ; BACKWARD COMPATIBLE WITH SOS VERSION
000131  HATTR           EQU        $1E                 ; HEADER ATTRIBUTES- PROTECT ETC.
000132  * HENTLN EQU $1F ; LENGTH OF EACH ENTRY
000133  * HMENT EQU $20 ; MAXIMUM NUMBER OF ENTRIES/BLOCK
000134  HCENT           EQU        $21                 ; CURRENT NUMBER OF FILES IN DIRECTORY
000135  HRBLK           EQU        $23                 ; OWNER'S DIRECTORY ADDRESS
000136  HRENT           EQU        $25                 ; OWNER'S DIRECTORY ENTRY NUMBER
000137  HRELN           EQU        $26                 ; OWNER'S DIRECTORY ENTRY LENGTH
000138  VBMAP           EQU        HRBLK
000139  VTBLK           EQU        HRENT               ; (USED FOR ROOT DIRECTORY ONLY)
000140  *
000141  * VOLUME CONTROL BLOCK INDEX CONSTANTS
000142  *
000143  VCBSIZE         EQU        $20                 ; CURRENT VCB IS 32 BYTES PER ENTRY (VER 0)
000144  VCBNML          EQU        0                   ; VOLUME NAME LENGTH BYTE
000145  VCBNAM          EQU        1                   ; VOLUME NAME
000146  VCBDEV          EQU        $10                 ; VOLUME'S DEVICE
000147  VCBSTAT         EQU        $11                 ; VOLUME STATUS. (80=FILES OPEN. 40=DISK SWITCHED.)
000148  VCBTBLK         EQU        $12                 ; TOTAL BLOCKS ON THIS VOLUME
000149  VCBTFRE         EQU        $14                 ; NUMBER OF UNUSED BLOCKS
000150  VCBROOT         EQU        $16                 ; ROOT DIRECTORY (DISK) ADDRESS
000151  *VCBMORG EQU $18 ; MAP ORGANIZATION (NOT SUPPORTED BY V 0)
000152  *VCBMBUF EQU $19 ; BIT MAP BUF NUM
000153  VCBDMAP         EQU        $1A                 ; FIRST (DISK) ADDRESS OF BITMAP(S)
000154  VCBCMAP         EQU        $1C                 ; RELATIVE ADDRESS OF BIT MAP WITH SPACE (ADD TO VCBDMAP)
000155  *VCBMNUM EQU $1D ; RELATIVE BIT MAP CURRENTLY IN MEMORY
000156  VCBOPNC         EQU        $1E                 ; CURRENT NUMBER OF OPEN FILES.
```



```
000157  VCBSWAP         EQU        $1F                 ; $8X IF VOLUME SWAPPED; $00 IF UNSWAPPED WHERE X=LOW ORDER BYTE OF VCB
ADR/16
000158  *
000159  * FILE CONTROL BLOCK INDEX CONSTANTS
000160  *
000161  FCBREFN         EQU        0                   ; FILE REFERENCE NUMBER (POSITION SENSITIVE)
000162  FCBDEVN         EQU        1                   ; DEVICE (NUMBER) ON WHICH FILE RESIDES
000163  *FCBHEAD EQU 2 ; BLOCK ADDRESS OF FILE'S DIRECTORY HEADER
000164  *FCBDIRB EQU 4 ; BLOCK ADDRESS OF FILE'S DIRECTORY
000165  FCBENTN         EQU        6                   ; ENTRY NUMBER WITHIN DIRECTORY BLOCK
000166  FCBSTYP         EQU        7                   ; STORAGE TYPE - SEED, SAPLING, TREE, ETC.
000167  FCBSTAT         EQU        8                   ; STATUS - INDEX/DATA/EOF/USAGE/TYPE MODIFIED.
000168  FCBATTR         EQU        9                   ; ATTRIBUTES - READ/WRITE ENABLE, NEWLINE ENABLE.
000169  FCBNEWL         EQU        $A                  ; NEW LINE TERMINATOR (ALL 8 BITS SIGNIFICANT).
000170  FCBBUFN         EQU        $B                  ; BUFFER NUMBER
000171  FCBFRST         EQU        $C                  ; FIRST BLOCK OF FILE
000172  FCBIDXB         EQU        $E                  ; BLOCK ADDRESS OF INDEX (0 IF NO INDEX)
000173  FCBDATB         EQU        $10                 ; BLOCK ADDRESS OF DATA
000174  FCBMARK         EQU        $12                 ; CURRENT FILE MARKER.
000175  FCBEOF          EQU        $15                 ; LOGICAL END OF FILE.
000176  FCBUSE          EQU        $18                 ; ACTUAL NUMBER OF BLOCKS ALLOCATED TO THIS FILE.
000177  FCBSWAP         EQU        $1A                 ; $8N = SWAPPED, $00 = UNSWAPPED VOLUME ("N" = VCB ENTRY NUMBER)
000178  FCBLEVL         EQU        $1B                 ; LEVEL AT WHICH THIS FILE WAS OPENED
000179  FCBDIRTY        EQU        $1C                 ; FCB MARKED AS MODIFIED
000180                  PAGE
000181  *
000182  * ZERO PAGE STUFF
000183  *
000184  PAR             EQU        $A0
000185  COMMAND         EQU        PAR
000186  C.DNAMP         EQU        PAR+1
000187  C.PATH          EQU        PAR+1
000188  C.REFNUM        EQU        PAR+1
000189  C.ISNEWL        EQU        PAR+2
000190  C.OUTEOF        EQU        PAR+2
000191  C.BASE          EQU        PAR+2
000192  C.MRKPTR        EQU        PAR+2
000193  C.OUTBUF        EQU        PAR+2
000194  C.NWPATH        EQU        PAR+3
000195  C.FILIST        EQU        PAR+3
000196  C.NEWL          EQU        PAR+3
000197  C.OUTVOL        EQU        PAR+3
000198  C.OUTREF        EQU        PAR+3
000199  C.XLIST         EQU        PAR+3
000200  C.MAXPTH        EQU        PAR+3
000201  C.MARK          EQU        PAR+3
000202  C.NEWEOF        EQU        PAR+3
000203  C.BYTES         EQU        PAR+4
000204  C.FILSTLN       EQU        PAR+5
000205  C.OUTBLK        EQU        PAR+5
000206  C.OPLIST        EQU        PAR+5
000207  C.XLEN          EQU        PAR+5
000208  C.FILID         EQU        PAR+6
000209  C.OUTCNT        EQU        PAR+6
000210  C.OPLSTLN       EQU        PAR+7
000211  C.AUXID         EQU        PAR+7
000212  C.STOR          EQU        PAR+9
000213  C.EOFLL         EQU        PAR+$A
000214  C.EOFLH         EQU        PAR+$B
000215  C.EOFHL         EQU        PAR+$C
000216  DEBUPTR         EQU        PAR+$D              ; NOTE SAME AS BELOW
000217  C.EOFHH         EQU        PAR+$D
000218  * C.SPARE EQU PAR+$E
000219  *
000220  DEVICE          EQU        $C0
000221  DHPCMD          EQU        DEVICE
000222  UNITNUM         EQU        DEVICE+1
000223  DSTATREQ        EQU        DEVICE+2
000224  DBUFPL          EQU        DEVICE+2
000225  DBUFPH          EQU        DBUFPL+1
000226  DSTATBFL        EQU        DEVICE+3            ; TO PASS BACK BUSY, WRITE PROTECT, READ PROTECT.
000227  DSTATBFH        EQU        DSTATBFL+1
000228  RQCNTL          EQU        DEVICE+4
000229  RQCNTH          EQU        RQCNTL+1
000230  BLOKNML         EQU        DEVICE+6
000231  BLOKNMH         EQU        BLOKNML+1
000232  BRDPTR          EQU        DEVICE+8            ; (AND 9)
000233  *
000234  DVNAMP          EQU        DEVICE+1            ; USED FOR 'VOLUME' TO CALL
000235  DVDNUM          EQU        DEVICE+3            ; 'GET.DNUM' IN DEVICE MANAGER.
000236  *
```

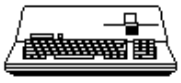


```
000237  SISBPH          EQU         SISTER+DBUFPH
000238  SISDSTAT        EQU         SISTER+DSTATBFH
000239  SSBRDPH         EQU         SISTER+BRDPTR+1
000240  *
000241                  PAGE
000242  *
000243  * ZERO PAGE TEMPORARIES
000244  *
000245  ZTEMPS          EQU         $B0
000246  PATHNML         EQU         ZTEMPS
000247  PATHNMH         EQU         PATHNML+1
000248  USRBUF          EQU         ZTEMPS
000249  TPATH           EQU         ZTEMPS+2
000250  WRKPATH         EQU         ZTEMPS+4
000251  TINDX           EQU         ZTEMPS+2
000252  DRBUFPL         EQU         ZTEMPS+4
000253  DRBUFPH         EQU         DRBUFPL+1
000254  VCBPTR          EQU         ZTEMPS+6
000255  BMADR           EQU         ZTEMPS+8
000256  FCBPTR          EQU         ZTEMPS+$A
000257  DATPTR          EQU         ZTEMPS+$C
000258  POSPTR          EQU         ZTEMPS+$E
000259  *
000260  MAXTEMPS        EQU         $F
000261  SISTEMPS        EQU         SISTER+ZTEMPS
000262  SSTIDXH         EQU         SISTER+TINDX+1
000263  SISPATH         EQU         SISTER+C.PATH+1
000264  SSNWPATH        EQU         SISTER+C.NWPATH+1
000265  SISUSRBF        EQU         SISTER+USRBUF+1
000266  SISOUTBF        EQU         SISTER+C.OUTBUF+1
000267  SISTPATH        EQU         SISTER+TPATH+1
000268  SISBMADR        EQU         SISTER+BMADR+1
000269  SISFCBP         EQU         SISTER+FCBPTR+1
000270  SISDATP         EQU         SISTER+DATPTR+1
000271  SISPOSP         EQU         SISTER+POSPTR+1
000272  *
000273  *
000274  * ADDRESSES
000275  *
000276  PATHBUF         EQU         $1000               ; NOTE: THIS IS $100 BYTES LONG.
000277  VCB             EQU         $1100
000278  GBUF            EQU         $1200               ; THRU $13FF
000279  *
000280  * INITIALIZATION EQUATES
000281  *
000282  BFMFCB1         EQU         $1C                 ; FCB PAGE 1 ADDR
000283  BFMFCB2         EQU         $1D                 ; FCB PAGE 2 ADDR
000284  BMAPAGE         EQU         <$B800              ; BIT MAP A ADDR
000285  BMBPAGE         EQU         <$BA00              ; BIT MAP B ADDR
000286  FCBZPP          EQU         FCBPTR
000287  *
000288  *
000289  *
000290                  PAGE
000291                  DSECT
000292                  ORG         $0                  ; (THE FOLLOWING DO NOT NEED TO BE ON ZERO PAGE. 7/16/80 JRH.)
000293  DATBLKL         DS          1
000294  DATBLKH         DS          1
000295  IDXADRL         DS          1                   ; DISK ADDRESS OF INDEX BLOCK
000296  IDXADRH         DS          1
000297  REQL            DS          1
000298  REQH            DS          1
000299  INDXBLK         DS          1
000300  LEVELS          DS          1
000301  TOTENT          DS          1
000302  ENTCNTL         DS          1
000303  ENTCNTH         DS          1
000304  CNTENT          DS          1
000305  NOFREE          DS          1
000306  BMCNT           DS          1
000307  SAPTR           DS          1
000308  TREPTR          DS          1
000309  TLINK           DS          2
000310  FLINK           DS          2
000311  PATHCNT         DS          1
000312  PFIXPTR         DS          2
000313  BMPTR           DS          1
000314  BASVAL          DS          1
000315  HALF            DS          1
000316  *
000317  *
```

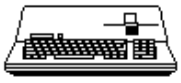


```
000318                  PAGE
000319  *
000320  * BIT MAP INFO TABLES (A & B)
000321  *
000322  BMTABSZ         EQU         $6
000323  BMTAB           DS          1
000324  BMBUFBNK        DS          1
000325  BMASTAT         DS          1
000326  BMADEV          DS          1
000327  BMAMADR         DS          1
000328  BMADADR         DS          2
000329  BMACMAP         DS          1                   ; SIMILAR TO VCBCMAP
000330  BMBSTAT         DS          1
000331  BMBDEV          DS          1
000332  BMBMADR         DS          1
000333                  DS          2                   ; BMBDADR
000334                  DS          1                   ; BMBCMAP
000335  *
000336  FCBADDRH        DS          1                   ; FILE CONTROL BLOCK'S BUFFER ADDRESS.
000337  FCBANKNM        DS          1                   ; AND BANK (SISTER PAGE) BYTE.
000338  TPOSLL          DS          1
000339  TPOSLH          DS          1
000340  TPOSHI          DS          1
000341  RWREQL          DS          1
000342  RWREQH          DS          1
000343  BULKCNT         DS          1
000344  NLCHAR          DS          1
000345  NPATHDEV        DS          3                   ; FOR NEW PATHNAME DEVICE AND DIRECTORY HEADER ADDRESS
000346  IOACCESS        DS          1                   ; USED TO DETERMINE IF A CALL HAS BEEN MADE TO THE DISK DEVICE HANDLER
000347  DEVNUM          DS          1                   ; CURRENT DEVICE TO BE ACCESSED.
000348  TOTDEVS         DS          1                   ; USED FOR ACCESSING DRIVES IN NUMERIC ORDER
000349  CMDTEMP         DS          1                   ; USED FOR TESTING REFNUM, TIME, AND DSKSWTCH (PRE)PROCESSING.
000350  DATELO          DS          1                   ; DATE AND TIME MUST RESIDE ON ZERO PAGE.
000351  DATEHI          DS          1
000352  TIMELO          DS          1
000353  TIMEHI          DS          1
000354  *
000355  DUPLFLAG        DS          1                   ; USED FOR DIFFERENCE BETWEEN VNFERR AND DUPVOL BY SYNPATH
000356  ZPGTEMP         DS          1                   ; A ONE-BYTE UNSTABLE TEMPORARY
000357  VCBENTRY        DS          1                   ; POINTER TO CURRENT VCB ENTRY
000358  *
000359                  DEND
000360  *
000361                  CHN         PATH,4,1
000362
000363  ********************************************************************************
000364  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: EQUATES
000365  ********************************************************************************
000366
```

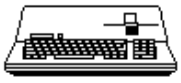
End of File -- Lines: 366 Characters: 13607



```
================================================================================
FILE: "SOS.FEB01.1982.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: FEB01.1982
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
000007  SL4:DR1:ASM INIT.SRC,INIT.OBJ,6,1
000008  SL4:DR1:ASM SYSGLOB.SRC,SYSGLOB.OBJ,6,1
000009  SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
000010  SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
000011  SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
000012  SL4:DR1:ASM UMGR.SRC,UMGR.OBJ,6,1
000013  SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
000014  SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
000015  SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
000016  SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
000017  SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
000018  SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
000019  SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
000020  SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
000021  END
000022
000023  ************************************************************************
000024  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: FEB01.1982
000025  ************************************************************************

End of File -- Lines: 25 Characters: 1004
```

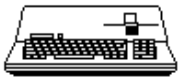


```
================================================================================
FILE: "SOS.FMGR.SRC.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: FMGR.SRC
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1  FILE MANAGER"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGFMGR
000010  ZZORG           EQU        *
000011                  MSB        OFF
000012                  REP        60
000013  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000014  *                     ALL RIGHTS RESERVED
000015                  REP        60
000016  *
000017  * FILE MANAGER (VERSION = 1.1O   )
000018  *              (DATE    = 8/04/81)
000019  *
000020  * THIS MODULE IS ENTERED FROM THE SYSTEM CALL MANAGER, AND
000021  * IS RESPONSIBLE FOR SWITCHING TO EITHER THE BLOCK FILE
000022  * MANAGER, OR THE CHARACTER FILE MANAGER.
000023  *
000024                  REP        60
000025  *
000026                  ENTRY      FMGR
000027                  ENTRY      LEVEL
000028  *
000029                  EXTRN      BFMGR
000030                  EXTRN      CFMGR
000031                  EXTRN      SYSERR
000032                  EXTRN      SERR
000033                  EXTRN      BADPATH
000034                  EXTRN      FNFERR
000035                  EXTRN      LVLERR
000036  *
000037  F.TPARMX        EQU        $A0                  ; LOC OF FILE SYSTEM CALL PARMS
000038  OPEN            EQU        $8
000039  CLOSE           EQU        $C
000040  SETLEVEL        EQU        $12
000041  GETLEVEL        EQU        $13
000042  F.REQCODE       EQU        F.TPARMX
000043  F.LEVEL         EQU        F.TPARMX+$1
000044  PATHNAME        EQU        F.TPARMX+$1
000045  REFNUM          EQU        F.TPARMX+$1
000046  PERIOD          EQU        $2E
000047  LEVEL           DFB        $1
000048                  PAGE
000049                  REP        60
000050  *
000051  * FILE MANAGER
000052  *
000053                  REP        60
000054  FMGR            EQU        *
000055  *
000056                  LDA        F.REQCODE
000057                  CMP        #OPEN
000058                  BCC        FMGR010
000059                  BEQ        FMGR020
000060                  CMP        #CLOSE
000061                  BCC        FMGR030
000062                  BEQ        FMGR040
000063                  CMP        #SETLEVEL
000064                  BEQ        SLEVEL
000065                  CMP        #GETLEVEL
000066                  BEQ        GLEVEL
000067  *
000068  FMGR010         JMP        BFMGR                ; EXIT
000069  *
000070  FMGR020         LDY        #1
000071                  LDA        (PATHNAME),Y
000072                  CMP        #PERIOD
000073                  BNE        FMGR010
000074                  JSR        CFMGR
000075                  BCC        FMGR024
000076                  LDA        SERR
```

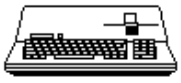


```
000077                  CMP         #FNFERR
000078                  BEQ         FMGR026
000079  FMGR024         RTS                             ; EXIT
000080  *
000081  FMGR026         LDA         #0
000082                  STA         SERR
000083                  JMP         BFMGR               ; EXIT
000084  *
000085  FMGR030         LDA         REFNUM
000086  FMGR031         BPL         FMGR010
000087                  JMP         CFMGR               ; EXIT
000088  *
000089  FMGR040         LDA         REFNUM
000090                  BNE         FMGR031
000091                  JSR         BFMGR               ; CLOSE (0)
000092                  JMP         CFMGR               ; EXIT
000093  *
000094  SLEVEL          LDA         F.LEVEL
000095                  BEQ         LVL.ERR
000096                  CMP         #4
000097                  BCS         LVL.ERR
000098                  STA         LEVEL
000099                  RTS
000100  LVL.ERR         LDA         #LVLERR
000101                  JSR         SYSERR
000102  *
000103  GLEVEL          LDY         #0
000104                  LDA         LEVEL
000105                  STA         (F.LEVEL),Y
000106                  RTS
000107  *
000108                  LST         ON
000109  ZZEND           EQU         *
000110  ZZLEN           EQU         ZZEND-ZZORG
000111                  IFNE        ZZLEN-LENFMGR
000112                  FAIL        2,"SOSORG           FILE IS INCORRECT FOR FMGR"
000113                  FIN
000114
000115  ********************************************************************************
000116  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: FMGR.SRC
000117  ********************************************************************************
000118
000119
```

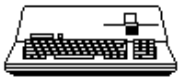
End of File -- Lines: 119 Characters: 2884



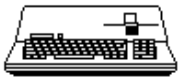
```
================================================================================
FILE: "SOS.FNDFIL.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: FNDFIL
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007  *
000008  *
000009  FINDFILE        JSR         LOOKFILE            ; SEE IF FILE EXISTS
000010                  BCS         NOFIND              ; BRANCH IF AN ERROR WAS ENCOUNTERED
000011  MOVENTRY        LDY         H.ENTLN             ; MOVE ENTIRE ENTRY INFO TO A SAFE AREA
000012  MOVENT1         LDA         (DRBUFPL),Y
000013                  STA         DFIL+D.STOR,Y
000014                  DEY
000015                  BPL         MOVENT1
000016                  LDA         #0                  ; TO INDICATE ALL IS WELL
000017  NOFIND          RTS                             ; RETURN CONDITION CODES.
000018                  PAGE
000019  *
000020  *
000021  LOOKFILE        JSR         PREPROOT            ; FIND VOLUME AND SET UP OTHER BORING STUFF
000022                  BCS         FNDERR              ; PASS BACK ANY ERROR ENCOUNTERED
000023                  LDY         #0                  ; TEST TO SEE IF ONLY ROOT WAS SPECIFIED.
000024                  LDA         (PATHNML),Y
000025                  BNE         LOOKFIL0            ; BRANCH IF MORE THAN ROOT.
000026                  LDA         #GBUF/256           ;  OTHERWISE, REPORT A BADPATH ERROR
000027                  STA         DRBUFPH             ; (BUT FIRST CREATE A PHANTOM ENTRY FOR OPEN)
000028                  LDA         #4
000029                  STA         DRBUFPL
000030                  LDY         #D.AUXID            ; FIRST MOVE IN ID, AND DATE STUFF.
000031  PHANTM1         LDA         (DRBUFPL),Y
000032                  STA         DFIL,Y
000033                  DEY
000034                  CPY         #D.CREDT-1
000035                  BNE         PHANTM1
000036  PHANTM2         LDA         ROOTSTUF-D.FILID,Y
000037                  STA         DFIL,Y
000038                  DEY
000039                  CPY         #D.FILID-1
000040                  BNE         PHANTM2
000041                  LDA         #DIRTYP*$10         ; FAKE DIRECTORY FILE
000042                  STA         DFIL+D.STOR
000043                  LDA         #BADPATH            ; (CARRY IS SET)
000044                  RTS
000045  *
000046  ROOTSTUF        DFB         0,2,0,4
000047                  DFB         0,0,8,0
000048  *
000049  LOOKFIL0        LDA         #0                  ; RESET FREE ENTRY INDICATOR
000050                  STA         NOFREE
000051                  SEC                             ; INDICATE THAT THE DIRECTORY TO BE SEARCHED HAS HEADER IN THIS BLOCK
000052  LOOKFIL1        LDA         #0                  ; RESET ENTRY COUNTER
000053                  STA         TOTENT
000054                  JSR         LOOKNAM             ; LOOK FOR NAME POINTED TO BY 'PATHNML'
000055                  BCC         NAMFOJMP            ; BRANCH IF NAME WAS FOUND.
000056                  LDA         ENTCNTL             ; HAVE WE LOOKED AT ALL OF THE
000057                  SBC         TOTENT              ;  ENTRIES IN THIS DIRECTORY?
000058                  BCC         DCRENTH             ; MAYBE, CHECK HI COUNT.
000059                  BNE         LOOKFIL2            ; NO, READ NEXT DIRECTORY BLOCK
000060                  CMP         ENTCNTH             ; HAS THE LAST ENTRY BEEN LOOKED AT (ACC=0)
000061                  BEQ         ERRFNF              ; YES, GIVE 'FILE NOT FOUND' ERROR.
000062                  BNE         LOOKFIL2            ; BRANCH ALWAYS.
000063  DCRENTH         DEC         ENTCNTH             ; SHOULD BE AT LEAST 1
000064                  BPL         LOOKFIL2            ; (THIS SHOULD BE BRANCH ALWAYS...)
000065  ERRDIR          LDA         #DIRERR             ; REPORT DIRECTORY MESSED UP.
000066  FNDERR          SEC                             ; INDICATE ERROR HAS BEEN ENCOUNTERED.
000067                  RTS
000068  NAMFOJMP        JMP         NAMFOUND            ; AVOID BRANCH OUT OF RANGE
000069  *
000070                  PAGE
000071  LOOKFIL2        STA         ENTCNTL             ; KEEP RUNNING COUNT
000072                  LDA         #GBUF/256           ; RESET INDIRECT POINTER
000073                  STA         DRBUFPH
000074                  LDA         GBUF+2              ; GET LINK TO NEXT DIRECTORY BLOCK
000075                  BNE         NXTDIR0             ; (IF THERE IS ONE)
000076                  CMP         GBUF+3              ; ARE BOTH ZERO, I.E. NO LINK?
```



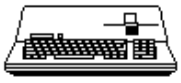
```
000077                  BEQ         ERRDIR              ; IF SO, THEN NOT ALL ENTRIES WERE ACCOUNTED FOR.
000078  NXTDIR0         STA         BLOKNML
000079                  LDA         GBUF+3
000080                  STA         BLOKNMH
000081                  JSR         RDGBUF              ; GO READ THE NEXT LINKED DIRECTORY IN.
000082                  BCC         LOOKFIL1            ; BRANCH IF NO ERROR.
000083                  RTS                             ; RETURN ERROR (IN ACCUMULATOR).
000084  TELFREEX        JMP         TELFREE
000085  *
000086  FNF0X           JMP         FNF0                ; AVOID BRANCH OUT OF RANGE
000087  *
000088  CFLAG           DS          1                   ; AM I CREATING?
000089  TTSAVE          DS          2                   ; CURRENT BLOCK ADDR
000090  BLOKSAVE        DS          2                   ; PARENT DIR ADDR
000091  *
000092  ERRFNF          LDA         NOFREE              ; WAS ANY FREE ENTRY FOUND?
000093                  BNE         FNF0X
000094                  LDA         GBUF+2              ; TEST LINK
000095                  BNE         TELFREEX
000096                  CMP         GBUF+3              ; IF BOTH ARE ZERO, THEN GIVE UP
000097                  BNE         TELFREEX            ; BRANCH IF NOT LAST DIR BLOCK
000098                  LDA         CFLAG               ; DOING A CREATE?
000099                  BEQ         FNF0X               ; NO, SIMPLY REPORT NOT FOUND
000100  *
000101  * EXTEND THE DIRECTORY BY A BLOCK
000102  *
000103                  LDA         BLOKSAVE            ; BUT NOT
000104                  ORA         BLOKSAVE+1          ;    IF A ROOT DIRECTORY!
000105                  BEQ         FNF0X               ;       FORU BLOCKS HARD CODED
000106                  LDA         TTLINK              ; FETCH CURRENT DIRECTORY
000107                  STA         TLINK               ; ADDR (GBUF)
000108                  LDA         TTLINK+1            ; AND ALLLOCATE A NEW
000109                  STA         TLINK+1             ; BY LINKING TO CURRENT
000110                  JSR         DIRWRT
000111                  BCS         FNF0                ; RATS! NO SPACE SAY "DIRFULL"
000112  *
000113  * SAVE CURRENT BLOCK ADDR
000114  *
000115                  LDA         TTLINK
000116                  STA         TTSAVE
000117                  LDA         TTLINK+1
000118                  STA         TTSAVE+1
000119  *
000120  * FETCH DESCENDENT
000121  *
000122                  LDA         GBUF+2
000123                  STA         BLOKNML
000124                  LDA         GBUF+3
000125                  STA         BLOKNMH
000126                  JSR         ZERGBUF             ; INIT THE NEW DIR BLOCK
000127  *
000128  * AND INSERT BACK POINTER
000129  * TO "CURRENT BLOCK"
000130  *
000131                  LDA         TTSAVE
000132                  STA         GBUF
000133                  LDA         TTSAVE+1
000134                  STA         GBUF+1
000135                  JSR         WRTGBUF
000136                  BCS         ERTS
000137  *
000138  * UPDATE DIR'S HEADER IN PARENT
000139  *
000140                  LDA         BLOKSAVE
000141                  STA         BLOKNML             ; PREPARE TO READ PARENT
000142                  LDX         BLOKSAVE+1
000143                  STX         BLOKNMH
000144                  JSR         RDGBUF              ; FETCH PARENT
000145                  LDY         #D.USAGE            ; BUMP BLOCKS USED BY HEADER
000146                  LDA         (DEBUPTR),Y
000147                  SEC
000148                  ADC         #0                  ; BY JUST ONE BLOCK
000149                  STA         (DEBUPTR),Y
000150                  INY
000151                  LDA         (DEBUPTR),Y         ; TWO BYTE BLOCKS USED
000152                  ADC         #0
000153                  STA         (DEBUPTR),Y
000154                  LDY         #D.EOF+1            ; INCREASE EOF BY $200
000155                  LDA         (DEBUPTR),Y
000156                  CLC
000157                  ADC         #2
```



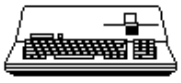
```
000158                  STA         (DEBUPTR),Y
000159                  INY
000160                  LDA         (DEBUPTR),Y
000161                  ADC         #0
000162                  STA         (DEBUPTR),Y
000163                  JSR         WRTGBUF             ; REWRITE PARENT DIR BLOCK
000164                  LDA         TTSAVE+1            ; REFETCH CURRENT DIR BLOCK
000165                  STA         BLOKNMH
000166                  LDA         TTSAVE
000167                  STA         BLOKNML
000168                  JSR         RDGBUF              ; BACK FROM THE SHADOWS AGAIN
000169                  JMP         ERRFNF              ; VOILA! WE HAVE EXTENDED THE DIRECTORY!
000170  *
000171  TELFREE         STA         D.ENTBLK
000172                  LDA         GBUF+3
000173                  STA         D.ENTBLK+1          ; ASSUME FIRST ENTRY OF NEXT BLOCK
000174                  LDA         #1                  ;  IS FREE FOR USE.
000175                  STA         D.ENTNUM
000176                  STA         NOFREE              ;  MARK D.ENTNUM AS VALID (FOR CREATE)
000177  FNF0            LDY         #0                  ; TEST FOR 'FILE NOT FOUND' VERSUS 'PATH NOT FOUND'
000178                  LDA         (PATHNML),Y
000179                  TAY
000180                  INY
000181                  LDA         (PATHNML),Y         ; IF NON-ZERO THEN 'PATH NOT FOUND'
000182  ERRPATH1        SEC                             ; IN EITHER CASE, INDICATE ERROR.
000183                  BEQ         FNF1
000184                  LDA         #PATHNOTFND         ; REPORT NO SUCH PATH.
000185  ERTS            RTS
000186  FNF1            LDA         #FNFERR             ; REPORT FILE NOT FOUND.
000187                  RTS
000188                  PAGE
000189  *
000190  NAMFOUND        LDA         (PATHNML),Y         ; (Y=0)
000191                  SEC
000192                  ADC         PATHNML             ; TEST FOR LAST NAME IN PATH
000193                  TAY                             ; IF ZERO, THEN THAT WAS LAST NAME
000194                  CLC                             ; TO INDICATE SUCCESS
000195                  LDA         PATHBUF,Y
000196                  BEQ         FILFOUND
000197  *NOW CHANGE THE PATHNAME POINTER TO POINT AT THE NEXT NAME IN THE PATH
000198                  STY         PATHNML
000199                  LDA         DRBUFPL             ; SAVE PARENTS
000200                  STA         DEBUPTR             ; ENTRY POINTER
000201                  LDA         DRBUFPH
000202                  STA         DEBUPTR+1           ; IN CASE ENTRY ON PAGE 2
000203                  LDA         BLOKNML             ; ADDRESS (DIR EXTEND)
000204                  STA         BLOKSAVE
000205                  LDA         BLOKNMH
000206                  STA         BLOKSAVE+1
000207                  LDY         #D.STOR             ; BE SURE THIS IS A DIRECTORY ENTRY
000208                  LDA         (DRBUFPL),Y         ; HIGH NIBBLE WILL TELL
000209                  AND         #$F0
000210                  CMP         #DIRTYP*16          ; IS IT A SUB-DIRECTORY?
000211                  BNE         ERRPATH1            ; REPORT THE USER'S MISTAKE
000212                  LDY         #D.FRST             ; GET ADDRESS OF FIRST SUB-DIRECTORY BLOCK
000213                  LDA         (DRBUFPL),Y
000214                  STA         BLOKNML             ; (NO CHECKING IS DONE HERE FOR A VALID
000215                  INY                             ;  BLOCK NUMBER... )
000216                  STA         D.HEAD              ; SAVE AS FILE'S HEADER BLOCK TOO.
000217                  LDA         (DRBUFPL),Y
000218                  STA         BLOKNMH
000219                  STA         D.HEAD+1
000220                  JSR         RDGBUF              ; READ SUB-DIRECTORY INTO GBUF
000221                  BCS         FNDERR1             ; RETURN IMMEDIATELY ANY ERROR ENCOUNTERED.
000222                  LDA         GBUF+HCENT+4        ; GET THE NUMBER OF FILES
000223                  STA         ENTCNTL             ;  CONTAINED IN THIS DIRECTORY
000224                  LDA         GBUF+HCENT+5
000225                  STA         ENTCNTH
000226                  LDA         GBUF+HCMP+4         ; TEST BACKWARD COMPATIBILITY
000227                  BEQ         MOVHEAD
000228  ERRCOMP         LDA         #CPTERR             ; TELL THEM THIS DIRECTORY IS NOT COMPATABLE
000229  NONAME          EQU         *
000230  FNDERR1         SEC
000231                  RTS
000232  MOVHEAD         JSR         MOVHED0             ; MOVE INFO ABOUT THIS DIRECTORY
000233                  JMP         LOOKFIL0            ; DO NEXT LOCAL PATHNAME
000234  *
000235  MOVHED0         LDX         #$A                 ; MOVE INFO ABOUT THIS DIRECTORY
000236  MOVHED1         LDA         GBUF+HCRDT+4,X
000237                  STA         H.CREDT,X
000238                  DEX
```



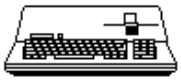
```
000239                  BPL         MOVHED1
000240                  RTS
000241  *
000242                  PAGE
000243  *
000244  *
000245  FILFOUND        EQU         *
000246  ENTADR          LDA         H.MAXENT            ; FIGURE OUT WHICH IS ENTRY NUMBER THIS IS.
000247                  SEC
000248                  SBC         CNTENT              ; MAX ENTRIES - COUNT ENTRIES + 1 = ENTRY NUMBER
000249                  ADC         #0                  ; (CARRY IS/WAS SET)
000250                  STA         D.ENTNUM
000251                  LDA         BLOKNML
000252                  STA         D.ENTBLK
000253                  LDA         BLOKNMH             ; AND INDICATE BLOCK NUMBER OF THIS DIRECTORY.
000254                  STA         D.ENTBLK+1
000255                  CLC
000256                  RTS
000257  *
000258  LOOKNAM         LDA         H.MAXENT            ; RESET COUNT OF FILES PER BLOCK
000259                  STA         CNTENT
000260                  LDA         #GBUF/256
000261                  STA         DRBUFPH
000262                  LDA         #4
000263  LOKNAM1         STA         DRBUFPL             ; RESET INDIRECT POINTER TO GBUF
000264                  BCS         LOKNAM2             ; BRANCH IF THIS BLOCK CONTAINS A HEADER
000265                  LDY         #D.STOR
000266                  LDA         (DRBUFPL),Y         ; GET LENGTH OF NAME IN DIRECTORY
000267                  BNE         ISNAME              ; BRANCH IF THERE IS A NAME.
000268                  LDA         NOFREE              ; TEST TO SEE IF A FREE ENTRY HAS BEEN DECLARED.
000269                  BNE         LOKNAM2             ; YES BUMP TO NEXT ENTRY
000270                  JSR         ENTADR              ; SET ADDRESS FOR CURRENT ENTRY
000271                  INC         NOFREE              ; INDICATE A FREE SPOT HAS BEEN FOUND
000272                  BNE         LOKNAM2             ; BRANCH ALWAYS.
000273  *
000274  ISNAME          AND         #$F                 ; STRIP TYPE (THIS IS CHECKED BY 'FILFOUND')
000275                  INC         TOTENT              ; (BUMP COUNT OF VALID FILES FOUND)
000276                  CMP         (PATHNML),Y         ; ARE BOTH NAMES OF THE SAME LENGTH?
000277                  BNE         LOKNAM2             ; NO, BUMP TO NEXT ENTRY
000278                  TAY
000279  CMPNAME         LDA         (DRBUFPL),Y         ; COMPARE NAMES LETTER BY LETTER
000280                  CMP         (PATHNML),Y
000281                  BNE         LOKNAM2
000282                  DEY                             ; HAVE ALL LETTERS BEEN COMPARED?
000283                  BNE         CMPNAME             ; NO, CONTINUE..
000284                  CLC                             ; BY GOLLY, WE GOT US A MATCH!
000285                  RTS
000286  *
000287  LOKNAM2         DEC         CNTENT              ; HAVE WE CHECKED ALL POSSIBLE ENTRIES IN THIS BLOCK?
000288                  BEQ         NONAME              ; YES, GIVE UP.
000289                  LDA         H.ENTLN             ; ADD ENTRY LENGTH TO CURRENT POINTER
000290                  CLC
000291                  ADC         DRBUFPL
000292                  BCC         LOKNAM1             ; BRANCH IF WE'RE STILL IN THE FIRST PAGE.
000293                  INC         DRBUFPH             ; LOOK ON SECOND PAGE
000294                  CLC                             ; CARRY SHOULD ALWAYS BE CLEAR BEFORE LOOKING AT NEXT.
000295                  BCC         LOKNAM1             ; BRANCH ALWAYS...
000296                  PAGE
000297  *
000298  *
000299  PREPROOT        JSR         FINDVOL             ; FIND CORRECT VOLUME AND DEVICE NUMBER
000300                  BCC         ROOT1               ; BRANCH IF IT WAS FOUND.
000301  ROOT0           JSR         LOOKVOL             ; OTHERWISE LOOK ON ALL DEVICES.
000302                  BCS         SRITZ               ; CAN'T FIND IT.
000303  ROOT1           LDA         #0                  ; ZERO OUT DIRECTORY TEMPS
000304                  LDY         #42                 ; (DECIMAL)
000305  CLRDSP          STA         D.DEV,Y
000306                  DEY
000307                  BPL         CLRDSP
000308                  LDY         #VCBDEV             ; SET UP DEVICE NUMBER
000309                  LDA         (VCBPTR),Y
000310                  STA         DEVNUM
000311                  STA         D.DEV               ; FOR FUTURE REFERENCE
000312                  INY
000313                  LDA         (VCBPTR),Y          ; GET CURRENT STATUS OF THIS VOLUME
000314                  STA         V.STATUS
000315                  LDY         #VCBROOT            ; GET BLOCK ADDRESS OF ROOT DIRECTORY TOO.
000316                  LDA         (VCBPTR),Y
000317                  STA         BLOKNML
000318                  STA         D.HEAD              ; PRESERVE AS HEADER
000319                  INY
```



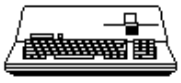
```
000320                  LDA     (VCBPTR),Y
000321                  STA     BLOKNMH
000322                  STA     D.HEAD+1
000323                  JSR     RDGBUF          ; GO READ IN ROOT
000324                  BCC     ROOT2           ; BRANCH IF NO ERROR
000325                  PHA                     ; SAVE ERROR CODE
000326                  LDY     #VCBSTAT        ; CHECK THIS BUGGER FOR AN OPEN FILE.
000327                  LDA     (VCBPTR),Y
000328                  ASL     A               ; (SHIFT OPEN STATUS INTO CARRY)
000329                  PLA                     ; GET ERROR CODE AGAIN
000330                  BCS     ROOTERR         ;  BRANCH IF ERROR NEEDS TO BE REPORTED
000331                  BNE     ROOT0           ; OTHERWISE, LOOK ELSEWHERE (BRANCH ALWAYS).
000332  *
000333  ROOT2           JSR     CHKROOT         ; VERIFY ROOT NAME
000334                  BEQ     ROOT3           ; BRANCH IF MATCHED.
000335                  LDY     #VCBSTAT        ; TEST FOR OPEN FILES ON THIS VOLUME BEFORE
000336                  LDA     (VCBPTR),Y      ;  LOOKING FOR IT ELSEWHERE.
000337                  BPL     ROOT0
000338                  JSR     USRREQ          ;  REQUEST USER MOUNT VOLUME
000339                  BCC     ROOT1           ;  USER SAID S/HE DID-- CHECK IT
000340                  LDA     #VNFERR         ; REPORT VOLUME NOT FOUND ERR IF REFUSE TO INSERT
000341  SRITZ           RTS
000342  *
000343                  PAGE
000344  ROOT3           LDY     #$F             ; (NOTE: X CONTAINS THE LENGTH OF THE ROOT NAME)
000345  ROOTINFO        LDA     GBUF+HCRDT+3,Y  ; SAVE HEADER INFO.
000346                  STA     V.STATUS,Y
000347                  DEY
000348                  BNE     ROOTINFO        ; LOOP TIL ALL 15 BYTES MOVED
000349                  LDA     H.FCNT
000350                  STA     ENTCNTL
000351                  LDA     H.FCNT+1
000352                  STA     ENTCNTH
000353                  TXA                     ; NOW THAT ROOT IS IDENTIFIED, ADJUST
000354                  SEC                     ;  PATH NAME POINTER TO NEXT NAME IN THE PATH
000355                  ADC     PATHNML
000356                  STA     PATHNML
000357                  CLC                     ; INDICATE NO ERROR
000358  ROOTERR         RTS
000359  *
000360  *
000361  CHKROOT         LDY     #0              ; GET LENGTH OF NAME
000362                  LDA     (PATHNML),Y
000363                  TAY
000364                  TAX                     ; SAVE IN X FOR LATTER ADJUSTMENT TO PATH POINTER
000365                  EOR     GBUF+4
000366                  AND     #$F             ; DOES PATHNAME HAVE SAME LENGTH AS DIRECTORY NAME?
000367                  BNE     NOTROOT         ; BRANCH IF NOT
000368  CKROOT1         LDA     (PATHNML),Y     ; COMPARE CHARACTER BY CHARACTER
000369                  CMP     GBUF+4,Y
000370                  BNE     NOTROOT
000371                  DEY
000372                  BNE     CKROOT1         ; LOOP UNTIL ALL CHARACTERS MATCH
000373  NOTROOT         RTS
000374  *
000375                  PAGE
000376  FINDVOL         LDA     #VCB/256        ; SEARCH VCB FOR VOLUME NAME
000377                  STA     VCBPTR+1
000378                  LDA     #0
000379                  STA     D.DEV
000380                  STA     VCBPTR
000381  FNDVOL1         PHA                     ; SAVE LAST SEARCH POSITION
000382                  TAX
000383                  LDY     #0              ; (INDEX TO PATHNAME POINTER)
000384                  LDA     VCB,X           ; GET LENGTH OF VOLUME NAME TO COMPARE
000385                  BEQ     NXTVCB          ; BRANCH IF VCB ENTRY IS EMPTY
000386                  CMP     (PATHNML),Y     ; ARE NAMES OF SAME LENGTH?
000387                  BNE     NXTVCB          ; NO, INDEX NEXT VCB
000388                  CLC                     ; SCAN NAME BACKWARDS
000389                  TAY
000390                  TXA
000391                  ADC     VCB,X
000392                  TAX                     ; NOW BOTH INDEXES POINT TO LAST CHARACTER OF THE NAMES TO COMPARE
000393  VOLNAM          LDA     (PATHNML),Y
000394                  CMP     VCB,X
000395                  BNE     NXTVCB
000396                  DEX
000397                  DEY
000398                  BNE     VOLNAM          ; CHECK ALL CHARACTERS
000399                  PLA                     ; SINCE A MATCH IS FOUND
000400                  STA     VCBPTR          ;  SET UP INDEX TO VCB ENTRY
```



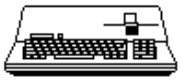
```
000401                  TAX
000402                  LDA     VCB+VCBSWAP,X       ;  BRANCH IF
000403                  BEQ     FOUNDVOL            ;  VOLUME NOT SWAPPED
000404                  JSR     SWAPIN              ;  IF USER REALLY WANTS IT, THEN BRING IN IF SWAPPED
000405                  BCC     FOUNDVOL            ;  BRANCH IF SUCCESS
000406                  LDA     #XIOERROR           ;  USER REFUSES TO MOUNT
000407                  RTS
000408  FOUNDVOL        CLC                         ; INDICATE VOLUME FOUND
000409                  RTS
000410  *
000411  NXTVCB          PLA                         ; GET CURRENT INDEX AGAIN.
000412                  CLC
000413                  ADC     #VCBSIZE            ; VCB ENTRY LENGTH.
000414                  BCC     FNDVOL1             ; BRANCH IF THER IS ANOTHER TO CHECK
000415                  RTS                         ; RETURN WITH CARRY SET TO SHOW FAILURE.
000416                  PAGE
000417  *
000418  *
000419  LOOKVOL         LDX     #12                 ; (1) COUNT+(12)DEVICE LIST
000420  LOOKVOL1        LDA     BLKDLST,X           ;  EXTRN
000421                  STA     SCRTCH,X            ;  MY CHANGEABLE COPY
000422                  DEX
000423                  BPL     LOOKVOL1            ;  WORK BACKWARDS SO
000424                  STA     TOTDEVS             ;  ENTRY ZERO IS TOTAL DEVICES LISTED
000425                  INX                         ;  MAKE XREG = ZERO
000426  LOKDEV1         INX
000427                  STX     SCRTCH
000428                  LDA     SCRTCH,X
000429                  CMP     D.DEV
000430                  BEQ     NXTDEV              ; DON'T LOOK AGAIN ON A DRIVE THAT HAS BEEN CHECKED
000431                  STA     DEVNUM              ; CHECK FOR DEVICE ALREADY LOGGED IN A VCB
000432                  JSR     DEVVCB              ; (CARRY CLEAR IF IT'S THERE)
000433                  BCC     LOKVOL1
000434                  LDA     #0                  ; FIND A FREE VCB TO LOG THIS GUY IN
000435  ENTVCB          TAX                         ; INDEX TO NEXT VCB ENTRY
000436                  LDA     VCB,X
000437                  BEQ     FREEVCB             ; FOUND A FREE SPOT.
000438                  TXA                         ; NOW INDEX TO NEXT, AND KEEP LOOKIN
000439                  CLC
000440                  ADC     #VCBSIZE            ; (EACH VCB ENTRY IS 32 BYTES)
000441                  BCC     ENTVCB              ; BRANCH IF MORE TO FIND
000442                  LDA     #0
000443  ENTVCB2         EQU     *                   ; SEE IF WE CAN REPLACE A DEVICE
000444                  TAX
000445                  LDA     VCB+VCBSTAT,X       ; VCB HAS FILES OPEN?
000446                  BEQ     FREEVCB             ; NO, USE IT!
000447                  TXA
000448                  CLC
000449                  ADC     #VCBSIZE            ; SEARCH NEXT VCB ENTRY
000450                  BCC     ENTVCB2
000451                  RTS                         ; FAILED TO FIND A FREE VCB ENTRY
000452  *
000453  CHKVLOG         LDY     #0                  ; MAKE SURE VOLUME WAS ACTUALLY LOGGED IN
000454                  LDA     (VCBPTR),Y
000455                  BNE     FOUNDVOL            ; AH, MADE IT...
000456                  LDA     #DUPVOL             ; WELL, NOT QUITE, THIS VOLUME CAN'T BE LOGGED
000457                  SEC
000458                  RTS
000459                  PAGE
000460  *
000461  FREEVCB         STX     VCBPTR              ; NOW THIS IS THE POINTER TO A FREE VCB
000462                  LDA     #2                  ; ROOT DIRECTORIES ALWAYS AT BLOCK 2
000463                  LDX     #0
000464                  BEQ     GETROOT             ; BRANCH ALWAYS
000465  LOKVOL1         LDY     #VCBSTAT            ; MAKE SURE NO FILES ARE ACTIVE ON
000466                  LDA     (VCBPTR),Y          ;  THE VOLUME BEFORE LOGGING IT IN.
000467                  BMI     SNSWIT              ;  BRANCH IF FILES ACTIVE
000468                  LDY     #VCBROOT+1          ; GET ADDRESS OF ROOT DIRECTORY
000469                  LDA     (VCBPTR),Y          ; HIGH FIRST.
000470                  TAX
000471                  DEY                         ; THEN LOW.
000472                  LDA     (VCBPTR),Y
000473  GETROOT         JSR     GETROT0
000474                  BCC     LOKVOL2             ; BRANCH IF SUCCESSFULLY READ.
000475                  LDA     #0                  ; OTHERWISE, TAKE THIS DEVICE OUT OF VCB
000476                  TAY
000477                  STA     (VCBPTR),Y          ; (VOLUME 'OFF LINE')
000478                  BEQ     NXTDEV              ; BRANCH ALWAYS
000479  *
000480  LOKVOL2         JSR     LOGVCB              ; GO UPDATE VCB TO INCLUDE CURRENT VOLUME INFO
000481                  BCS     NXTDEV              ;  IF NOT A SOS DISKETTE, SKIP TO NEXT DEVICE
```



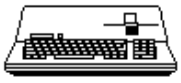
```
000482                  JSR     CHKROOT             ; GO COMPARE TO SEE IF WE FOUND WHAT WE'RE
000483                  BEQ     CHKVLOG             ;  LOOKING FOR...
000484  *
000485  NXTDEV          LDX     SCRTCH              ; LOOK AT OTHER DEVICES?
000486                  CPX     TOTDEVS
000487                  BCC     LOKDEV1             ; YES.
000488                  LDA     #VNFERR             ; REPORT VOLUME NOT FOUND.
000489                  RTS
000490  *
000491  SNSWIT          EQU     *                   ;  SENSE DSWITCH
000492                  LDY     #VCBDEV
000493                  LDA     (VCBPTR),Y
000494                  STA     DEVNUM              ;  MAKE SURE DEVICE NUMBER IS CURRENT
000495                  JSR     TWRPROT1            ;  USES DEVNUM
000496                  LDA     DSWGLOB             ;  DISK SWITCH GLOBAL
000497                  BEQ     NXTDEV              ;  BRANCH IF NO DISK SWITCH
000498                  JSR     VERFYVOL            ;  COMPARES VCBPTR VS. DEVNUM CONTENTS
000499                  BCC     NXTDEV              ;  BRANCH IF DISK HAS NOT BEEN SWITCHED
000500                  JSR     CHKROOT             ;  COMPARES PATHNML VS. GBUF
000501                  BNE     NXTDEV              ;  IGNORE IF NOT WHAT WE ARE LOOKING FOR
000502                  LDX     #0                  ;  LOOK FOR FREE
000503                  JSR     SNSWIT1
000504                  BCS     NXTDEV              ;  ANY ERRORS LOGGING IN THE NEW VOLUME
000505                  JMP     CHKVLOG             ;  MAKE SURE THE NEW VOLUME IS LOGGED
000506  SNSWIT1         LDA     VCB,X               ;  VCB ENTRY
000507                  BEQ     SNSWIT2             ;  BRANCH IF FOUND
000508                  TXA
000509                  CLC
000510                  ADC     #VCBSIZE            ;  LOOK AT NEXT VCB AREA
000511                  TAX
000512                  BCC     SNSWIT1
000513                  RTS                         ;  CAN'T BE LOGGED IN!
000514  SNSWIT2         LDA     #0
000515                  STA     DUPLFLAG            ;  TURN OFF DUPLICATE VOLUME FLAG
000516                  STX     VCBPTR
000517                  JSR     LOGVCB1             ;  PARTIALLY LOG IN THE NEW VOLUME
000518                  BCS     NONSOS              ;  CS MEANS NONSOS ERROR
000519                  LDA     DUPLFLAG            ;  WAS IT A DUPLICATE VOLUME?
000520                  BNE     SNSWIT6             ;  BRANCH IF YES
000521                  LDY     #VCBSWAP            ;  BY MAKING SWAP BYTE NON ZERO
000522                  LDA     #1
000523                  STA     (VCBPTR),Y          ;  SO SWAPOUT WON'T AFFECT
000524                  LDA     DEVNUM              ;  A REG PASSES DEVNUM TO SWAPOUT
000525                  JSR     SWAPOUT             ;  OLD ACTIVE MOUNT MUST BE SWAPPED
000526                  BCC     SNSWIT3
000527                  LDA     #XIOERROR           ;  USER REFUSED TO REPLACE OLD VOLUME
000528                  RTS
000529  SNSWIT3         LDY     #VCBSWAP            ;  NOW LOG IN THE NEW ALL THE WAY
000530                  LDA     #0
000531                  STA     (VCBPTR),Y
000532  SNSWIT4         JSR     VERFYVOL            ;  DON'T BOTHER TO ASK IF NEW VOLUME IS ALREADY MOUNTED
000533                  BCC     SNSWIT5             ;  BRANCH IF NEW VOLUME ON LINE
000534                  JSR     USRREQ              ;  ASK USER TO REMOUNT NEW VOLUME
000535                  BCC     SNSWIT4             ;  USER SAYS THEY DID: CHECK IT OUT
000536                  LDA     #VNFERR
000537  SNSWIT5         RTS
000538  SNSWIT6         LDA     #DUPVOL
000539                  SEC
000540                  RTS
000541                  PAGE
000542  *
000543  NONSOS          LDA     #NOTSOS             ; TELL EM IT'S NOT A SOS DISK (COULD BE PASCAL)
000544                  RTS                         ;  CARRY SHOULD ALREADY BE SET
000545  *
000546  *
000547  DEVVCB          LDA     #0                  ; SCAN VCB FOR DEVICE SPECIFIED IN 'DEVNUM'
000548  DVCB1           TAX                         ; FIRST TEST FOR VALID VCB.
000549                  LDA     VCB,X
000550                  BEQ     DVCB2
000551                  LDA     VCB+VCBSWAP,X       ;  SWAPPED VOLUMES DON'T COUNT
000552                  BNE     DVCB2               ;  AS LOGGED IN
000553                  LDA     VCB+VCBDEV,X        ; GET DEVICE NUMBER
000554                  CMP     DEVNUM              ; TEST AGAINST REQUESTED DEVICE
000555                  BEQ     FOUNDEV             ; YES, SET UP A POINTER TO IT
000556  DVCB2           TXA                         ; BUMP TO NEXT VCB
000557                  CLC
000558                  ADC     #VCBSIZE
000559                  BCC     DVCB1               ; BRANCH IF MORE TO LOOK AT.
000560                  RTS                         ; RETURN CARRY SET TO INDICATE NOT FOUND
000561  *
000562  TSTDUPVOL       LDX     VCBPTR              ; PRESERVE CURRENT ADDR OF FREE VCB
```



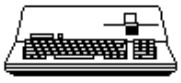
```
000563                  LDA     #0                      ; LOOK FOR A CURRENTLY LOGGED ON VOLUME OF THE SAME NAME.
000564  TSDUPV1         STA     VCBPTR
000565                  JSR     CMPVCB
000566                  BCS     TSDUPV2                 ; BRANCH IF NO MATCH.
000567                  LDY     #VCBSTAT                ; TEST FOR ANY OPEN FILES.
000568                  LDA     (VCBPTR),Y
000569                  BMI     FOUNDDUP                ; TELL THE SUCKER HE CAN'T LOOK AT THIS VOLUME!
000570                  LDA     #0                      ; TAKE DUPLICATE OFF LINE IF NO OPEN FILES.
000571                  TAY
000572                  STA     (VCBPTR),Y
000573                  BEQ     NODUPVOL                ; RETURN THAT ALL IS OK TO LOG IN NEW.
000574  TSDUPV2         LDA     VCBPTR
000575                  CLC
000576                  ADC     #VCBSIZE                ; BUMP TO NEXT ENTRY.
000577                  BCC     TSDUPV1
000578  NODUPVOL        EQU     *
000579  FOUNDEV         CLC
000580  FNDDUP1         STX     VCBPTR
000581                  RTS
000582  *
000583  FOUNDDUP        STA     DUPLFLAG                ; A DUPLICATE HAS BEEN DETECTED.
000584                  SEC                             ; INDICATE ERROR
000585                  LDA     VCBPTR                  ;  SAVE ADDRESS OF DUPLICATE
000586                  STA     VCBENTRY
000587                  BCS     FNDDUP1                 ; BRANCH ALWAYS TAKEN
000588                  PAGE
000589  *
000590  *
000591  LOGVCB          LDY     #VCBNML                 ; IS THIS A PREVIOUSLY LOGGED IN VOLUME
000592                  LDA     (VCBPTR),Y              ; (ACC=0?)
000593                  BEQ     LOGVCB1                 ; NO, GO AHEAD AND PREPARE VCB.
000594                  JSR     CMPVCB                  ; DOES VCB MATCH VOLUME READ?
000595                  BCC     VCBLOGD                 ; YES, DON'T DISTURB IT.
000596  LOGVCB1         LDA     #0                      ; ZERO OUT VCB ENTRY
000597                  LDY     #VCBSIZE-1
000598  ZERVCB          STA     (VCBPTR),Y
000599                  DEY
000600                  BPL     ZERVCB
000601                  JSR     TSTSOS                  ; MAKE SURE IT'S A SOS DISKETTE.
000602                  BCS     VCBLOGD                 ; IF NOT, RETURN CARRY SET.
000603                  JSR     TSTDUPVOL               ; FIND OUT IF A DUPLICATE WITH OPEN FILES ALREADY EXISTS
000604                  BCS     NOTLOG0
000605                  LDA     GBUF+4                  ; MOVE VOLUME NAME TO VCB
000606                  AND     #$F                     ; STRIP ROOT MARKER
000607                  TAY
000608                  PHA
000609  MOVOLNM         LDA     GBUF+4,Y
000610                  STA     (VCBPTR),Y
000611                  DEY
000612                  BNE     MOVOLNM
000613                  PLA                             ; GET LENGTH AGAIN
000614                  STA     (VCBPTR),Y              ; SAVE THAT TOO.
000615                  LDY     #VCBDEV                 ; SAVE DEVICE NUMBER ALSO.
000616                  LDA     DEVNUM
000617                  STA     (VCBPTR),Y
000618                  JSR     CLEARBMS                ;  MARKS THIS DEVICES OLD BITMAPS AS INVALID (A REG PASSED)
000619                  LDA     GBUF+VTBLK+4            ; AND TOTOL NUMBER OF BLOCKS ON THIS UNIT,
000620                  LDY     #VCBTBLK
000621                  STA     (VCBPTR),Y
000622                  LDA     GBUF+VTBLK+5
000623                  INY
000624                  STA     (VCBPTR),Y
000625                  LDY     #VCBROOT
000626                  LDA     BLOKNML                 ; AND ADDRESS OF ROOT DIRECTORY
000627                  STA     (VCBPTR),Y
000628                  INY
000629                  LDA     BLOKNMH
000630                  STA     (VCBPTR),Y
000631                  LDY     #VCBDMAP
000632                  LDA     GBUF+VBMAP+4            ; AND LASTLY, THE ADDRESS
000633                  STA     (VCBPTR),Y              ;  OF THE FIRST BITMAP
000634                  LDA     GBUF+VBMAP+5
000635                  INY
000636                  STA     (VCBPTR),Y
000637                  CLC                             ; INDICATE THAT IT WAS LOGGED IF POSIBLE.
000638  VCBLOGD         RTS
000639  NOTLOG0         JMP     NOTLOG1
000640                  PAGE
000641  CMPVCB          LDA     GBUF+4                  ; COMPARE VOLUME NAME IN VCB
000642                  AND     #$F
000643                  LDY     #VCBNML                 ;  WITH NAME IN DIRECTORY
```



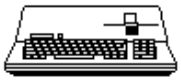
```
000644                  CMP         (VCBPTR),Y          ; ARE THEY SAME LENGTH
000645                  BNE         NOTSAME
000646                  TAY
000647  VCBCMP1         LDA         GBUF+4,Y
000648                  CMP         (VCBPTR),Y
000649                  BNE         NOTSAME
000650                  DEY
000651                  BNE         VCBCMP1
000652                  CLC                             ; INDICATE MATCH.
000653                  RTS
000654  *
000655  VERFYVOL        LDX         #0                  ; READ IN ROOT DIRECTORY HEADER.
000656                  LDA         #2
000657                  JSR         GETROT0
000658                  BCS         NOVRFY1             ; PASS BACK WHATEVER OTHER ERROR OCCURS.
000659                  JSR         CMPVCB              ; TEST ROOT WITH VOLUME NAME IN VCB.
000660                  BCC         NOVRFY              ;  BRANCH IF ROOT MATCHES VCB
000661                  LDA         #0                  ;  OTHERWISE, PASS BACK FOREIGN VOLUME ERROR (SOS OR UCSD)
000662  NOVRFY          RTS                             ; RETURN RESULTS IN CARRY.
000663  NOVRFY1         LDA         #VNFERR             ;  NOTHING IN DRIVE
000664                  RTS
000665  *
000666  GETROT0         STA         BLOKNML
000667                  STX         BLOKNMH             ; STORE ADDRESS AND READ IN ROOT
000668                  JSR         RDGBUF
000669                  BCC         RETROT2             ; BRANCH IF SUCCESSFULLY READ.
000670  NOTSAME         EQU         *
000671                  SEC                             ; INDICATE ERROR
000672  RETROT2         RTS
000673  *
000674  NOTLOG1         LDX         VCBPTR              ;  LOAD THE VCB ADDRESS
000675                  LDA         VCBENTRY            ;  OF THE DUPLICATE VOLUME
000676                  STA         VCBPTR
000677                  STX         VCBENTRY            ;  AND SAVE THE FREE VCB SPACE ADDR
000678                  LDY         #VCBDEV             ;  IS DUPLICATE ON SAME DEVICE?
000679                  LDA         DEVNUM
000680                  CMP         (VCBPTR),Y
000681                  BNE         NOTLOG2             ;  BRANCH IF NOT
000682                  JSR         SWAPIN              ;  SWAP IN IF NECESSARY
000683                  LDA         #0
000684                  STA         DUPLFLAG            ;  NO MORE DUPLICATE VOLUME STATUS
000685                  LDA         VCBPTR              ;  MAKE CHKROOT WORK IN A MOMENT
000686                  STA         PATHNML             ;  THIS IS INCREDIBLY GROSS
000687  ;  BUT IS A RESULT OF MAKING VOLUME A SPECIAL
000688  ;  CASE OF SEARCHING ALL DEVICES FOR
000689  ;  A KNOWN VOLUME
000690                  CLC
000691                  RTS
000692  NOTLOG2         LDA         VCBENTRY            ;  REACH HERE IF REAL DUPLICATE VOLUME
000693                  STA         VCBPTR              ;  RESOTRE FREE VCB PTR
000694                  CLC
000695                  RTS                             ;  DUPLICATE VOLUME PRETENDS TO BE NO ERROR
000696                  PAGE
000697  *
000698  TSFRBLK         LDY         #VCBTFRE+1
000699                  LDA         (VCBPTR),Y          ; FIND OUT IF ENOUGH FREE BLOCKS
000700                  DEY                             ; ARE AVAILABLE TO ACCOMODATE REQEST.
000701                  ORA         (VCBPTR),Y          ; BUT FIRST FIND OUT IF WE GOT A PROPER COUNT FOR THIS VOLUME.
000702                  BNE         CMPFREB             ; BRANCH IF COUNT IS NON-ZERO
000703                  DEY                             ; IF ZERO, THEN COUNT MUST BE TAKEN
000704                  LDA         (VCBPTR),Y          ;  GET HIGH TOTAL BLKS
000705                  TAX                             ;  SAVE IT
000706                  DEY                             ;  GET LOW
000707                  LDA         (VCBPTR),Y          ;  TOTAL BLKS
000708                  BNE         TSFR01
000709                  DEX                             ;  ADJUST FOR BITMAP BLOCK BOUNDARY
000710  TSFR01          TXA
000711                  LSR         A                   ; DIVIDE BY 16. THE RESULT IS THE NUMBER
000712                  LSR         A                   ;  OF BIT MAPS TO BE SEARCHED.
000713                  LSR         A
000714                  LSR         A
000715                  STA         BMCNT               ; SAVE IT.
000716                  LDA         #0                  ; START COUNT AT ZERO.
000717                  STA         SCRTCH
000718                  STA         SCRTCH+1
000719                  LDA         #$FF                ; MARK 'FIRST FREE' TEMP AS UNKNOWN
000720                  STA         NOFREE
000721                  LDY         #VCBDEV             ; MAKE SURE BIT MAP IS UP TO DATE
000722                  LDA         (VCBPTR),Y          ; GET DEVICE NUMBER
000723                  TAX                             ; PASS TO 'UPBMAP' IN X
000724                  JSR         UPBMAP              ; (NOTHING HAPPENS IF IT DON'T HAFTA.)
```



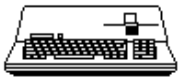
```
000725                  BCS         TFBERR              ; BRANCH IF WE GOT TROUBLE,
000726                  LDY         #VCBDMAP            ; GET ADDRESS OF FIRST BIT MAP.
000727                  LDA         (VCBPTR),Y
000728                  STA         BLOKNML
000729                  INY                             ; (FOR HIGH ADDRESS)
000730                  LDA         (VCBPTR),Y
000731                  STA         BLOKNMH
000732  BMAPRD          JSR         RDGBUF              ; USE G(ENERAL)BUFF(ER) FOR TEMPORARY
000733                  BCS         TFBERR              ;  SPACE TO COUNT FREE BLOCKS (BITS)
000734                  JSR         COUNT               ; GO COUNT EM
000735                  DEC         BMCNT               ; WAS THAT THE LAST BIT MAP?
000736                  BMI         CHGVCB              ; IF SO, GO CHANGE FCB TO AVOID DOING THIS AGAIN!
000737                  INC         BLOKNML             ; NOTE: THE ORGANIZATION OF THE BIT MAPS
000738                  BNE         BMAPRD              ;  ARE CONTIGUOUS FOR SOS VERSION 0
000739                  INC         BLOKNMH             ;  IF SOME OTHER ORGANIZATION IS IMPLEMENTED, THIS CODE
000740                  JMP         BMAPRD              ;  MUST BE CHANGED!
000741                  PAGE
000742  *
000743  CHGVCB          LDY         #VCBCMAP            ; MARK WHICH BLOCK HAD FIRST FREE SPACE
000744                  LDA         NOFREE
000745                  BMI         DSKFULL             ; BRANCH IF NO FREE SPACE WAS FOUND.
000746                  STA         (VCBPTR),Y
000747                  LDY         #VCBTFRE+1          ; UPDATE THE FREE COUNT.
000748                  LDA         SCRTCH+1            ; GET HIGH COUNT BYTE
000749                  STA         (VCBPTR),Y          ; UPDATE VOLUME CONTROL BLOCK.
000750                  DEY
000751                  LDA         SCRTCH
000752                  STA         (VCBPTR),Y          ; AND LOW BYTE TOO...
000753  CMPFREB         LDA         (VCBPTR),Y          ; COMPARE TOTAL AVAILABLE
000754                  SEC
000755                  SBC         REQL                ;  FREE BLOCKS ON THIS VOLUME.
000756                  INY
000757                  LDA         (VCBPTR),Y
000758                  SBC         REQH
000759                  BCC         DSKFULL
000760                  CLC
000761                  RTS
000762  DSKFULL         LDA         #OVRERR
000763                  SEC
000764  TFBERR          RTS
000765                  PAGE
000766  *
000767  COUNT           LDY         #0                  ; BEGIN AT THE BEGINNING.
000768  FRCONT          LDA         GBUF,Y              ; GET BIT PATTERN
000769                  BEQ         FRCNT1              ; DON'T BOTHER COUNTING NOTHIN'
000770                  JSR         CNTFREE
000771  FRCNT1          LDA         GBUF+$100,Y         ; DO BOTH PAGES WITH SAME LOOP
000772                  BEQ         FRCNT2
000773                  JSR         CNTFREE
000774  FRCNT2          INY
000775                  BNE         FRCONT              ; LOOP TILL ALL 512 BYTES COUNTED
000776                  BIT         NOFREE              ; HAS FIRST BLOCK WITH FREE SPACE BEEN FOUND YET?
000777                  BPL         FRCNT3              ; BRANCH IF IT HAS.
000778                  LDA         SCRTCH              ; TEST TO SEE IF ANY BLOCKS WERE COUNTED
000779                  ORA         SCRTCH+1
000780                  BEQ         FRCNT3              ; BRANCH IF NONE COUNTED.
000781                  LDY         #VCBTBLK+1
000782                  LDA         (VCBPTR),Y          ; SHOW THIS MAP IS FIRST WITH FREE SPACE
000783                  SEC                             ;  CORRECT FOR EXACT MULTIPLES OF $1000
000784                  SBC         #$01
000785                  LSR         A
000786                  LSR         A
000787                  LSR         A
000788                  LSR         A
000789                  SEC                             ; SUBTRACT COUNTDOWN FROM TOTAL BIT MAPS
000790                  SBC         BMCNT
000791                  STA         NOFREE
000792  FRCNT3          RTS
000793  *
000794  CNTFREE         ASL         A                   ; COUNT THE NUMBER OF BITS IN THIS BYTE.
000795                  BCC         CFREE1
000796                  INC         SCRTCH
000797                  BNE         CFREE1
000798                  INC         SCRTCH+1
000799  CFREE1          TAX
000800                  BNE         CNTFREE             ; LOOP UNTIL ALL BITS COUNTED.
000801                  RTS
000802
000803                  CHN         ALLOC,4,1
000804
000805  ********************************************************************************
```



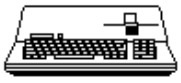
```
000806  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: FNDFIL
000807  ****************************************************************************
000808
000809

End of File -- Lines: 809 Characters: 31627
```



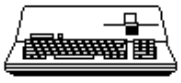
```
================================================================================
FILE: "SOS.INIT.SRC.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: INIT.SRC
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1 INITIALIZATION"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGINIT
000010  ZZORG           EQU        *
000011                  MSB        OFF
000012                  REP        100
000013  *          COPYRIGHT (C) APPLE COMPUTER INC.  1981
000014  *                      ALL RIGHTS RESERVED
000015                  REP        100
000016  *
000017  * SOS INIT MODULE (VERSION = 1.1O   )
000018  *                 (DATE    = 8/04/81)
000019  *
000020                  REP        100
000021  *
000022                  ENTRY      INT.INIT
000023                  ENTRY      EVQ.INIT
000024                  ENTRY      CLK.INIT
000025                  ENTRY      MMGR.INIT
000026                  ENTRY      BMGR.INIT
000027                  ENTRY      DMGR.INIT
000028                  ENTRY      CFMGR.INIT
000029                  ENTRY      BFM.INIT
000030  *
000031  *  EXTERNAL SUBROUTINES & DATA
000032  *
000033                  EXTRN      SXPAGE
000034                  EXTRN      SYSDEATH
000035  *
000036  *  INTERRUPT SYSTEM INITIALIZATION
000037  *
000038                  EXTRN      COLDSTRT
000039                  EXTRN      IRQ.RCVR
000040                  EXTRN      NMI.RCVR
000041                  EXTRN      NMIFLAG
000042                  EXTRN      SIRTABLE
000043                  EXTRN      SIRTBLSIZ
000044                  EXTRN      ZPGSTACK
000045                  EXTRN      ZPGSTART
000046  *
000047  *  EVENT QUEUE INITIALIZATION
000048  *
000049                  EXTRN      EV.QUEUE
000050                  EXTRN      EVQ.LEN
000051                  EXTRN      EVQ.CNT
000052                  EXTRN      EVQ.SIZ
000053                  EXTRN      EVQ.FREE
000054                  EXTRN      EVQ.LINK
000055  *
000056  *  CLOCK INITIALIZATION
000057  *
000058                  EXTRN      PCLOCK
000059  *
000060  *  CHARACTER FILE MANAGER INITIALIZATION
000061  *
000062                  EXTRN      CFCB.MAX
000063                  EXTRN      CFCB.DEV
000064  *
000065  *  DEVICE MANAGER INITIALIZATION
000066  *
000067                  EXTRN      DMGR
000068                  EXTRN      MAX.DNUM
000069  *
000070  *  BUFFER MANAGER INITIALIZATION
000071  *
000072                  EXTRN      BUF.CNT
000073                  EXTRN      PGCT.T
000074                  EXTRN      XBYTE.T
000075                  EXTRN      BUFREF
000076  *
```



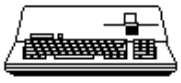
```
000077  *  MEMORY MANAGER INITIALIZATION
000078  *
000079                  EXTRN      ST.CNT
000080                  EXTRN      ST.ENTRY
000081                  EXTRN      ST.FREE
000082                  EXTRN      ST.FLINK
000083                  EXTRN      VRT.LIM
000084                  EXTRN      MEMSIZE
000085                  EXTRN      MEM2SML
000086  *
000087  *  BLOCK FILE MANAGER INITIALIZATION
000088  *
000089                  EXTRN      FCBZPP
000090                  EXTRN      PATHBUF
000091                  EXTRN      VCB
000092                  EXTRN      WORKSPC
000093                  EXTRN      PFIXPTR
000094                  EXTRN      FCBADDRH
000095                  EXTRN      BMAPAGE
000096                  EXTRN      BMBPAGE
000097                  EXTRN      BMAMADR
000098                  EXTRN      BMBMADR
000099                  EXTRN      BFMFCB1
000100                  EXTRN      BFMFCB2
000101  *
000102  *  CONSTANT DECLARATIONS
000103  *
000104  TRUE            EQU        $80
000105  FALSE           EQU        $00
000106  BITON6          EQU        $40
000107  BITON7          EQU        $80
000108  *
000109  *  SYSTEM CONTROL REGISTERS
000110  *
000111  E.REG           EQU        $FFDF               ;ENVIRONMENT REGISTER
000112  Z.REG           EQU        $FFD0               ;ZERO PAGE REGISTER
000113                  SBTL       "INTERRUPT SYSTEM INITIALIZATION"
000114  *
000115  *  6522 REGISTERS
000116  *
000117  D.DDRB          EQU        $FFD2
000118  D.DDRA          EQU        $FFD3
000119  D.ACR           EQU        $FFDB
000120  D.PCR           EQU        $FFDC
000121  D.IFR           EQU        $FFDD
000122  D.IER           EQU        $FFDE
000123  E.IORB          EQU        $FFE0
000124  E.DDRB          EQU        $FFE2
000125  E.DDRA          EQU        $FFE3
000126  E.ACR           EQU        $FFEB
000127  E.PCR           EQU        $FFEC
000128  E.IFR           EQU        $FFED
000129  E.IER           EQU        $FFEE
000130  ACIASTAT        EQU        $C0F1
000131  *
000132  *
000133                  REP        60
000134  *
000135  *  THIS SUBROUTINE INITIALIZES THE INTERRUPT SYSTEM.
000136  *  ALL HARDWARE INTERRUPTS ARE MASKED AND THE
000137  *  INTERRUPT ALLOCATION TABLE IS CLEARED.
000138  *
000139                  REP        60
000140  *
000141  *
000142  INT.INIT        EQU        *
000143                  SEI                            ;DISABLE INTERRUPTS
000144                  LDA        #>ZPGSTART          ;SET UP MIH
000145                  STA        ZPGSTACK            ;  ZERO PAGE STACK POINTER
000146  *
000147                  LDA        E.REG               ;SELECT $C000 I/O SPACE
000148                  PHA                            ;  AND SET 1 MHZ
000149                  ORA        #BITON7+BITON6
000150                  STA        E.REG
000151  *
000152                  STA        ACIASTAT            ;RESET ACIA
000153  *
000154                  LDA        #$FF                ;SET UP 6522 D
000155                  STA        D.DDRB
000156                  STA        D.DDRA
000157                  LDA        #$00
```



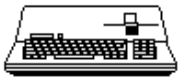
```
000158                  STA         D.ACR
000159                  LDA         #$76
000160                  STA         D.PCR
000161                  LDA         #$7F
000162                  STA         D.IFR
000163                  STA         D.IER
000164                  LDA         #$82
000165                  STA         D.IER
000166  *
000167                  LDA         #$3F                ;SET UP 6522 E
000168                  STA         E.DDRB
000169                  LDA         #$0F
000170                  STA         E.DDRA
000171                  LDA         #$00
000172                  STA         E.ACR
000173                  LDA         #$63
000174                  STA         E.PCR
000175                  LDA         #$7F
000176                  STA         E.IFR
000177                  STA         E.IER
000178  *
000179                  LDA         #$FF
000180                  STA         E.IORB              ;SOUND PORT
000181                  BIT         $C0D8               ;DISABLE GRAPHICS SCROLL
000182                  BIT         $C0DA               ;DISABLE CHARACTER DOWNLOAD
000183                  BIT         $C0DC               ;DISABLE ENSEL
000184                  BIT         $C0DE               ;SET ENSIO FOR INPUT
000185  *
000186                  PLA                             ;RESTORE E REGISTER
000187                  STA         E.REG
000188  *
000189                  LDA         #FALSE
000190                  STA         NMIFLAG             ;CLEAR NMI WAIT FLAG
000191                  LDY         #>SIRTBLSIZ-1
000192  INTI010         STA         SIRTABLE,Y          ;  ALLOCATION TABLE
000193                  DEY
000194                  BPL         INTI010
000195                  LDA         #TRUE
000196                  STA         SIRTABLE+$0A        ;LOCK DOWN ANY SLOT SIR
000197  *
000198                  LDX         #$05
000199  INTI020         LDA         RAMVECT,X           ;SET UP VECTORS
000200                  STA         $FFFA,X             ;  AT $FFFA - $FFFF
000201                  LDA         RAMJMPS,X           ;SET UP JMP INSTRUCTIONS
000202                  STA         $FFCA,X             ;  AT $FFCA - $FFCF
000203                  DEX
000204                  BPL         INTI020
000205                  RTS
000206  *
000207  RAMVECT         DW          NMI.RCVR
000208                  DW          COLDSTRT
000209                  DW          IRQ.RCVR
000210  RAMJMPS         JMP         NMI.RCVR
000211                  JMP         IRQ.RCVR
000212                  SBTL        "EVENT QUEUE INITIALIZATION"
000213                  REP         60
000214  *
000215  *  THIS SUBROUTINE INITIALIZES THE EVENT QUEUE.  ALL ENTRIES
000216  *  ARE CLEARED AND LINKED INTO THE FREE LIST.  THE ACTIVE
000217  *  LIST IS EMPTY.
000218  *
000219                  REP         60
000220  *
000221  *
000222  EVQ.INIT        EQU         *
000223  *
000224  *  CLEAR ALL ENTRIES
000225  *
000226                  LDY         #>EVQ.LEN
000227                  LDA         #0
000228  EVQI010         STA         EV.QUEUE-1,Y
000229                  DEY
000230                  BNE         EVQI010
000231  *
000232  *  SET UP FREE LIST
000233  *
000234                  LDX         #>EVQ.CNT-2
000235                  LDA         #>EVQ.SIZ
000236                  STA         EVQ.FREE
000237  EVQI020         TAY
000238                  CLC
```



```
000239                  ADC         #>EVQ.SIZ
000240                  STA         EVQ.LINK,Y
000241                  DEX
000242                  BNE         EVQI020
000243                  RTS
000244                  SBTL        "PSEUDO CLOCK INITIALIZATION"
000245                  REP         60
000246  *
000247  *  THIS SUBROUTINE INITIALIZES THE PSEUDO CLOCK.  IF THE
000248  *  RAM BEHIND THE "D" 6522 HAS THE PROPER CHECKSUM, IT
000249  *  IS USED TO INITIALIZE THE PSEUDO CLOCK.  OTHERWISE,
000250  *  THE PSEUDO CLOCK IS SET TO ZERO.
000251  *
000252  * (ADDED 23 OCT 81)
000253  * BOTH THE CLOCK AND PSEUDO CLOCK ARE
000254  * ARE NOW INITIALIZED
000255  *
000256                  REP         60
000257  *
000258  PCLK            EQU         $F0
000259  CKSUM           EQU         $F2
000260  CLKICR          EQU         $11                 ; CLOCK INTERRUPT CONTROL REG
000261  CLKSTBY         EQU         $16                 ; CLOCK STANDBY INTERRUPT
000262  CLOCK           EQU         $C070
000263  *
000264  CLK.INIT        EQU         *
000265                  LDA         #$D0
000266                  STA         PCLK                ;POINT (PCLK) TO 8F:FFD0
000267                  LDA         #$FF
000268                  STA         PCLK+1
000269                  LDA         #$8F
000270                  STA         SXPAGE+PCLK+1
000271                  LDA         #$A5
000272                  STA         CKSUM               ;INITIALIZE CHECKSUM
000273  *
000274                  LDY         #$00
000275  CLK010          LDA         (PCLK),Y            ;COPY SAVED CLOCK DATA
000276                  STA         PCLOCK,Y            ;  TO PSEUDO CLOCK
000277                  EOR         CKSUM
000278                  STA         CKSUM               ;UPDATE CHECKSUM
000279                  INY
000280                  CPY         #$0A
000281                  BCC         CLK010
000282  *
000283                  CMP         (PCLK),Y            ;TEST CHECKSUM
000284                  BEQ         CLK030
000285  *
000286                  LDA         #$00
000287  CLK020          DEY
000288                  STA         PCLOCK,Y            ;ZERO PSEUDO CLOCK
000289                  BNE         CLK020
000290  CLK030          LDA         E.REG
000291                  PHA
000292                  ORA         #$80                ; SET 1 MHZ
000293                  STA         E.REG
000294                  LDA         #$00
000295                  LDY         Z.REG
000296                  LDX         #CLKICR
000297                  STX         Z.REG
000298                  STA         CLOCK               ; DISABLE CLOCK INTERRUPTS
000299                  LDX         #CLKSTBY
000300                  STX         Z.REG
000301                  STA         CLOCK               ; DISABLE STANDBY INTERRUPT
000302                  STY         Z.REG
000303                  PLA
000304                  STA         E.REG
000305                  RTS
000306                  SBTL        "CHARACTER FILE MANAGER INITIALIZATION"
000307                  REP         60
000308  *
000309  * CHAR FILE MANAGER INITIALIZATION ROUTINE
000310  *
000311  * CFMGR.INIT INITIALIZES ALL ENTRIES IN THE CFCB TABLE TO
000312  * THE "FREE" STATE.
000313  *
000314                  REP         60
000315  *
000316  CFMGR.INIT      EQU         *
000317                  LDA         #$80
000318                  LDX         #CFCB.MAX-1
000319  CFINIT010       STA         CFCB.DEV,X
```

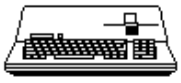


```
000320                  DEX
000321                  BPL         CFINIT010
000322                  RTS
000323                  SBTL        "DEVICE MANAGER INITIALIZATION"
000324                  REP         60
000325  *
000326  * DEVICE MANAGER INITIALIZATION ROUTINE
000327  *
000328  * INITIALIZES THE SYSTEM DEVICE TABLE (SDT) BY WALKING THE
000329  * DEVICE INFORMATION BLOCK (DIB) LINKS.  CALLED BY SYSLDR.
000330  *
000331                  REP         60
000332  *
000333  D.TPARMX        EQU         $C0
000334  REQCODE         EQU         D.TPARMX+$00
000335  DNUM            EQU         D.TPARMX+$01
000336  DNUM.TEMP       DS          1
000337  *
000338  *
000339  DMGR.INIT       EQU         *
000340                  LDX         MAX.DNUM
000341                  INC         MAX.DNUM            ; MAX.DNUM:=MAX DEV NUMBER IN SYSTEM+1
000342                  STX         DNUM.TEMP
000343  DMI110          LDA         #8                  ; INITIALIZE ALL DEVICES IN SYSTEM (D.INIT)
000344                  STA         REQCODE
000345                  LDA         DNUM.TEMP
000346                  STA         DNUM
000347                  JSR         DMGR
000348                  DEC         DNUM.TEMP
000349                  BNE         DMI110
000350                  RTS                             ; NORMAL EXIT
000351                  SBTL        "BUFFER MANAGER INITIALIZATION"
000352                  REP         60
000353  *
000354  * BMGR.INIT
000355  *
000356  * THIS ROUTINE INITIALIZES THE BUFFER TABLE'S ENTRIES TO "FREE".
000357  * CALLED DURING SYSTEM BOOT.
000358  *
000359                  REP         60
000360  *
000361  BMGR.INIT       EQU         *
000362                  LDA         #$FF                ; USED WHEN FINDING LOWEST BUFFER IN TBL (BUFCOMPACT)
000363                  STA         XBYTE.T
000364  *
000365                  LDX         #BUF.CNT-1
000366                  LDA         #$80
000367  BUFI010         STA         PGCT.T,X            ;SET ALL ENTRIES "FREE"
000368                  DEX
000369                  BNE         BUFI010
000370  *
000371                  STX         BUFREF              ;ZERO COUNT BYTE IN BUFFER REFERENCE TABLE
000372  *
000373                  CLC
000374                  RTS
000375                  SBTL        "MEMORY MANAGER INITIALIZATION"
000376                  REP         60
000377  *
000378  * MMGR.INIT
000379  *
000380  * THIS ROUTINE INITIALIZES THE MEMORY MANAGER'S SEGMENT TABLE
000381  * TO FREE ENTRIES, AND DETERMINES THE MEMORY SIZE OF THE
000382  * MACHINE (96K,128K,160K,192K,224K,256K,..,512K IN 32K STEPS).
000383  *
000384                  REP         60
000385  *
000386  MMGR.INIT       EQU         *
000387  *
000388  * INIT SEGMENT TABLE
000389  *
000390                  LDA         #0
000391                  STA         ST.ENTRY
000392                  LDA         #$81
000393                  STA         ST.FREE
000394  *
000395                  LDY         #ST.CNT-1
000396                  LDA         #$80                ; SET LAST LINK TO NULL
000397                  STA         ST.FLINK,Y
000398  MEMI010         TYA
000399                  ORA         #$80
000400                  DEY
```

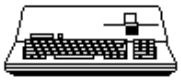


```
000401                  STA         ST.FLINK,Y
000402                  BNE         MEMI010
000403  *
000404  * COMPUTE VIRTUAL LIMIT FROM MEMORY SIZE
000405  * VRT.LIM := NUMBER OF PAGES IN BANK SWITCHED MEMORY - 1
000406  *         := (MEMSIZ-2)*64 - 1
000407  *         := (MEMSIZ-4)*64 + 127
000408  *
000409                  SEC
000410                  LDA         MEMSIZE
000411                  SBC         #4
000412                  BCC         MEMI.ERR
000413                  LSR         A
000414                  LSR         A
000415                  STA         VRT.LIM+1
000416                  LDA         #$FE
000417                  ROR         A
000418                  STA         VRT.LIM
000419                  CLC
000420                  RTS                             ; NORMAL EXIT
000421  *
000422  MEMI.ERR        LDA         #MEM2SML            ; FATAL ERR - MEM < 64K
000423                  JSR         SYSDEATH
000424                  PAGE
000425                  REP         60
000426  *
000427  *  BLOCK FILE MANAGER INITIALIZATION
000428  *
000429                  REP         60
000430  *
000431  SISTER          EQU         $1400               ;BFM XPAGE
000432  BFM.INIT        EQU         *
000433                  LDA         #BFMFCB1            ; ADDRESS OF PAGE 1 OF FCB
000434                  STA         >FCBZPP+1
000435                  LDA         #BFMFCB2            ; AND PAGE 2
000436                  STA         >FCBZPP+3
000437                  LDA         #0
000438                  STA         >FCBZPP             ; FCB PAGE ALIGNED
000439                  STA         >FCBZPP+2
000440                  STA         SISTER+FCBZPP+1     ; PREPARE PART OF EXTEND BYTE
000441                  STA         SISTER+FCBZPP+3
000442                  TAY                             ; MAKE ZERO INTO INDEX
000443  CLRBUFFS        EQU         *
000444                  STA         PATHBUF,Y           ; PATHNAME BUFFER PAGE
000445                  STA         VCB,Y               ; VOLUME CONTROL BLOCK PAGE
000446                  STA         (>FCBZPP),Y         ; BOTH FILE CONTROL BLOCK PAGES
000447                  STA         (>FCBZPP+2),Y
000448                  INY
000449                  BNE         CLRBUFFS
000450                  LDX         #$3F                ; SIZE OF MY ZERO PAGE STUFF
000451  CLRZWRK         STA         0,X                 ; ZERO PAGE ZEROED
000452                  STA         WORKSPC,X
000453                  DEX
000454                  BPL         CLRZWRK
000455                  LDA         #<PATHBUF
000456                  STA         PFIXPTR+1
000457                  LDA         #BFMFCB1
000458                  STA         FCBADDRH
000459                  LDA         #BMAPAGE            ; BIT MAP A PAGE NUMBER
000460                  STA         BMAMADR
000461                  LDA         #BMBPAGE            ; BIT MAP B PAGE NUMBER
000462                  STA         BMBMADR
000463                  CLC
000464                  RTS
000465  *
000466                  LST         ON
000467  ZZEND           EQU         *
000468  ZZLEN           EQU         ZZEND-ZZORG
000469                  IFNE        ZZLEN-LENINIT
000470                  FAIL        2,"SOSORG           FILE IS INCORRECT FOR INIT"
000471                  FIN
000472
000473  ********************************************************************
000474  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: INIT.SRC
000475  ********************************************************************
000476
```

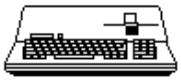
End of File -- Lines: 476 Characters: 11714



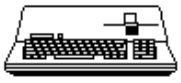
```
================================================================================
FILE: "SOS.IPL.SRC1.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC1
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1  INTRPTS. & PROC. LAUNCH"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGIPL
000010  ZZORG           EQU        *
000011                  MSB        OFF
000012                  REP        60
000013  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000014  *                    ALL RIGHTS RESERVED
000015                  REP        60
000016  *
000017  *  THIS MODULE IS RESPONSIBLE FOR FIELDING ALL INTERRUPTS
000018  *  AND RELAUNCHING THE INTERRUPTED CODE AFTER THE INTERRUPTS
000019  *  HAVE BEEN PROCESSED.  THE MAJOR FUNCTIONAL AREAS ARE:
000020  *
000021  *       GENERAL INTERRUPT RECEIVER
000022  *       NMI INTERRUPT RECEIVER
000023  *       DISPATCHER
000024  *       INTERRUPT ALLOCATION & DEALLOCATION
000025  *       EVENT QUEUE MANAGER
000026  *       TABLE INITIALIZATION
000027  *
000028                  REP        60
000029  *
000030  *  SUBROUTINE ENTRY POINTS
000031  *
000032                  ENTRY      IRQ.RCVR            ;GENERAL INTERRUPT RECEIVER
000033                  ENTRY      NMI.RCVR            ;NON-MASKABLE INTRPT RCVR
000034                  ENTRY      DISPATCH            ;DISPATCHER
000035                  ENTRY      ALLOCSIR            ;SIR ALLOCATION
000036                  ENTRY      DEALCSIR            ;SIR DEALLOCATION
000037                  ENTRY      SELC800             ;SELECT I/O EXPANSION ROM
000038                  ENTRY      NMIDSBL             ;DISABLE NMI
000039                  ENTRY      NMIENBL             ;ENABLE NMI
000040                  ENTRY      NMIDBUG             ;NMI DEBUG ENTRY
000041                  ENTRY      NMICONT             ;NMI DEBUG CONTINUATION
000042                  ENTRY      QUEEVENT            ;QUEUE AN EVENT
000043  *
000044  *  EXTERNAL SUBROUTINES & DATA
000045  *
000046                  EXTRN      SCMGR
000047                  EXTRN      CHKBUF
000048  *
000049  *  SYSTEM DEATH ERRORS
000050  *
000051                  EXTRN      SYSDEATH
000052                  EXTRN      BADBRK
000053                  EXTRN      BADINT1
000054                  EXTRN      BADINT2
000055                  EXTRN      NMIHANG
000056                  EXTRN      EVQOVFL
000057                  EXTRN      STKOVFL
000058  *
000059  *  LINKAGE DATA FOR INITIALIZATION ROUTINES
000060  *
000061                  ENTRY      EV.QUEUE
000062                  ENTRY      EVQ.CNT
000063                  ENTRY      EVQ.SIZ
000064                  ENTRY      EVQ.LEN
000065                  ENTRY      EVQ.FREE
000066                  ENTRY      EVQ.LINK
000067                  ENTRY      SIRTABLE
000068                  ENTRY      SIRTBLSIZ
000069                  ENTRY      ZPGSTACK
000070                  ENTRY      ZPGSTART
000071  *
000072  *  SYSGLOB DATA
000073  *
000074                  EXTRN      SERR
000075                  EXTRN      CEVPRI              ;CALLER'S EVENT PRIORITY
000076                  EXTRN      SYSBANK             ;SYSTEM BANK
```



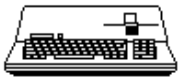
```
000077                  EXTRN      KYBDNMI
000078                  EXTRN      NMISPSV
000079                  EXTRN      NMIFLAG             ;NMI PENDING FLAG
000080                  EXTRN      SCRNMODE            ;CURRENT SCREEN MODE
000081                  EXTRN      SIRTEMP             ;FOR ALLOCSIR & DEALCSIR
000082                  EXTRN      SIRARGSIZ
000083                  EXTRN      IRQCNTR             ;FLASE IRQ COUNTER
000084                  EXTRN      NMICNTR             ;TWO BYTE COUNTER
000085                  EXTRN      QEVTEMP
000086                  EXTRN      QEV.THIS
000087                  EXTRN      QEV.LAST
000088                  EXTRN      BACKMASK
000089  *
000090  *  CONSTANT DECLARATIONS
000091  *
000092  FALSE           EQU        $00
000093  BITON0          EQU        $01
000094  BITON1          EQU        $02
000095  BITON2          EQU        $04
000096  BITON4          EQU        $10
000097  BITON5          EQU        $20
000098  BITON6          EQU        $40
000099  BITON7          EQU        $80
000100  BITOFF3         EQU        $F7
000101  BITOFF4         EQU        $EF
000102  BITOFF5         EQU        $DF
000103  BITOFF6         EQU        $BF
000104  BITOFF7         EQU        $7F
000105  BACKBIT         EQU        $20                 ; BACKUP BIT MASK
000106  *
000107  *  SYSTEM CONTROL REGISTERS
000108  *
000109  B.REG           EQU        $FFEF               ;BANK REGISTER
000110  E.REG           EQU        $FFDF               ;ENVIRONMENT REGISTER
000111  Z.REG           EQU        $FFD0               ;ZERO PAGE REGISTER
000112  *
000113  *  6522 REGISTERS
000114  *
000115  D.IFR           EQU        $FFDD
000116  D.IER           EQU        $FFDE
000117  E.IORB          EQU        $FFE0
000118  E.IFR           EQU        $FFED
000119  E.IER           EQU        $FFEE
000120  E.IORA          EQU        $FFEF
000121                  PAGE
000122  *
000123  *  REGISTER PRESERVATION EQUATES
000124  *  FOR USE DURING INTERRUPT PROCESSING
000125  *
000126  A.SAVE          EQU        $103
000127  S.SAVE          EQU        $104
000128  SP.SAVE         EQU        $1FF
000129  E.SAVE          EQU        $1FE
000130  Z.SAVE          EQU        $1FD
000131  B.SAVE          EQU        $1FC
000132  EXPNSLOT        DFB        $00                 ;CURRENT I/O EXPANSION SLOT
000133  *
000134  *  STATUS LOCATIONS FOR INTERRUPT POLLING
000135  *
000136  ACIASTAT        EQU        $C0F1
000137  ANYSLOT         DFB        BITON1
000138  SLOT1           EQU        $C065
000139  SLOT2           EQU        $C064
000140  SLOT3           DFB        BITON5
000141  SLOT4           DFB        BITON4
000142  *
000143  *  INTERRUPT ZERO PAGE STORAGE & EQUATES
000144  *
000145  SIRARGS         EQU        $F9                 ;AND $FA
000146  QEVARGS         EQU        $FB                 ;AND $FC
000147  IRQADDR         EQU        $FD                 ;AND $FE
000148  ZPGSP           EQU        $FF
000149  ZPGSTART        EQU        $F8
000150  ZPGSTOP         EQU        $28
000151  ZPGSPACE        EQU        $20
000152  ZPGSTACK        DFB        ZPGSTART
000153  *
000154  *  SYSTEM INTERNAL RESOURCE
000155  *  TABLE STORAGE AND EQUATES
000156  *
000157  SIRTBLSIZ       EQU        $18
```



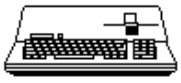
```
000158  SIRTABLE        DS          SIRTBLSIZ
000159  SIRADR.L        DS          SIRTBLSIZ
000160  NMIADR.L        DS          1                   ;MUST PRECEED SIRADR.H
000161  SIRADR.H        DS          SIRTBLSIZ
000162  SIRADR.B        DS          SIRTBLSIZ
000163  *
000164  *  EVENT QUEUE STORAGE AND EQUATES
000165  *
000166  EVQ.SIZ         EQU         6                   ;ENTRY SIZE
000167  EVQ.CNT         EQU         $07                 ;ENTRY COUNT
000168  EVQ.LEN         EQU         $2A                 ;(EVQ.SIZ*EVQ.CNT)
000169  EV.QUEUE        DS          EVQ.LEN
000170  EVQ.FREE        EQU         EV.QUEUE+2          ;FIRST FREE ENTRY INDEX
000171  EVQ.LINK        EQU         EV.QUEUE+0          ;NEXT ACTIVE ENTRY INDEX
000172  EVQ.PRI         EQU         EV.QUEUE+1          ;EVENT PRIORITY
000173  EVQ.ID          EQU         EV.QUEUE+2          ;EVENT IDENTIFICATION
000174  EVQ.ADRL        EQU         EV.QUEUE+3          ;EVENT ADDRESS:  LOW BYTE
000175  EVQ.ADRH        EQU         EV.QUEUE+4          ;EVENT ADDRESS:  HIGH BYTE
000176  EVQ.BANK        EQU         EV.QUEUE+5          ;EVENT ADDRESS:  BANK
000177                  SBTL        "GENERAL INTERRUPT RECEIVER"
000178                  REP         60
000179  *
000180  *  THIS IS THE GENERAL INTERRUPT RECEIVER.  WHEN AN
000181  *  INTERRUPT OCCURS, THE CPU PASSES CONTROL TO THE GIR
000182  *  THROUGH THE IRQ VECTOR.  THE GIR IS RESPONSIBLE FOR
000183  *  SAVING THE CURRENT ENVIRONMENT, SETTING UP THE SOS
000184  *  ENVIRONMENT, AND CALLING THE APPROPRIATE CODE MODULE.
000185  *  IF THE INTERRUPT WAS CAUSED BY A BRK, THE GIR CALLS
000186  *  THE SYSTEM CALL MANAGER.  OTHERWISE, THE GIR POLLS THE
000187  *  I/O DEVICES AND CALLS THE APPROPRIATE MASTER INTERRUPT
000188  *  HANDLER.  WHEN THE SCM OR MIH RETURNS, THE GIR PASSES
000189  *  CONTROL TO THE DISPATCHER.
000190  *
000191                  REP         60
000192  *
000193  IRQ.RCVR        EQU         *
000194  *
000195  *  SAVE CPU REGISTERS A, X, & Y ON CURRENT STACK
000196  *
000197                  PHA
000198                  TXA
000199                  PHA
000200                  TYA
000201                  PHA
000202  *
000203  *  CHECK FOR STACK OVERFLOW AND
000204  *  SAVE INTERRUPTED STATUS IN Y REGISTER.
000205  *
000206                  TSX
000207                  CPX         #$FA
000208                  BCC         GIR005
000209                  LDA         #>STKOVFL
000210                  JSR         SYSDEATH
000211  GIR005          LDY         S.SAVE,X
000212  *
000213  *  SET UP INTERRUPT ENVIRONMENT:
000214  *    BINARY ARITHMETIC, 2 MHZ, I/O ENABLED,
000215  *    RAM WRITE ENABLED, PRIMARY STACK,
000216  *    AND $F000 RAM SELECTED.  PRESERVE
000217  *    USER STATE OF SCREEN AND RESET LOCK.
000218  *
000219                  CLD
000220                  LDA         E.REG
000221                  TAX
000222                  AND         #BITON5+BITON4
000223                  ORA         #BITON6+BITON2
000224                  STA         E.REG
000225  *
000226  *  IF NOT ALREADY ON PRIMARY STACK, SAVE USER'S STACK
000227  *  POINTER AND SET UP SOS STACK POINTER.
000228  *
000229                  TXA
000230                  AND         #BITON2
000231                  BNE         GIR010
000232                  TXA
000233                  TSX
000234                  STX         SP.SAVE
000235                  LDX         #>E.SAVE
000236                  TXS
000237                  TAX
000238  *
```



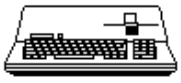
```
000239  *  SAVE E, Z, B, & I/O EXPANSION SLOT ON SOS STACK
000240  *  IF BRK THEN CALL SCMGR ELSE POLL I/O DEVICES
000241  *
000242  GIR010          TXA
000243                  PHA
000244                  LDA         Z.REG
000245                  PHA
000246                  LDA         B.REG
000247                  PHA
000248                  LDA         EXPNSLOT
000249                  PHA
000250                  BIT         $CFFF
000251                  BIT         $C020               ;RESET I/O SPACE
000252                  LDA         #$00
000253                  STA         EXPNSLOT
000254                  TYA
000255                  AND         #BITON4
000256                  BEQ         POLL.IO
000257  *
000258  *  CALL SYSTEM CALL MANAGER; ON RETURN, PUT ERROR CODE IN
000259  *  USER'S A REGISTER AND SET RETURN STATUS, THEN DISPATCH.
000260  *
000261                  TSX                             ;CHECK FOR
000262                  CPX         #>B.SAVE-2          ;  REENTRANT
000263                  BEQ         GIR020              ;  SYSTEM CALL
000264                  LDA         #>BADBRK
000265                  JSR         SYSDEATH
000266  GIR020          LDA         E.REG               ;SELECT $C000 RAM
000267                  AND         #BITOFF6
000268                  STA         E.REG
000269                  CLI                             ;ENABLE INTERRUPTS
000270                  JSR         SCMGR               ;CALL THE SYSTEM CALL MGR
000271                  LDA         #BACKBIT            ; GET THE MASK
000272                  STA         BACKMASK            ; SET IT IN SYSGLOB
000273                  JSR         CHKBUF
000274                  SEI
000275                  LDX         SP.SAVE
000276                  LDA         Z.SAVE
000277                  EOR         #BITON0             ;SET ZERO PAGE TO
000278                  STA         Z.REG               ;  CALLER'S STACK
000279                  LDA         SERR
000280                  STA         >A.SAVE,X
000281                  PHP
000282                  LDA         >S.SAVE,X
000283                  AND         #$7D
000284                  STA         >S.SAVE,X
000285                  PLA
000286                  AND         #$82
000287                  ORA         >S.SAVE,X
000288                  STA         >S.SAVE,X
000289                  JMP         DISPATCH
000290                  PAGE
000291  *
000292  *  SET INTERRUPT ZERO PAGE AND SOS BANK
000293  *    THEN POLL I/O DEVICES
000294  *
000295  POLL.IO         BIT         E.IORA              ;VERIFY THAT 'IRQ IS LOW
000296                  BPL         PIO006
000297                  INC         IRQCNTR             ;BUMP FALSE IRQ COUNTER
000298                  BNE         PIO004
000299                  INC         IRQCNTR+1
000300  PIO004          JMP         DISPATCH
000301  PIO006          LDA         #0                  ;SET INTERRUPT ZERO PAGE
000302                  STA         Z.REG
000303                  LDA         E.REG
000304                  ORA         #BITON7             ;FORCE 1 MHZ FOR
000305                  STA         E.REG               ;  READING ACIA STATUS
000306                  AND         #BITOFF7
000307                  LDX         #$01
000308                  LDY         ACIASTAT            ;ANY INTERRUPT ON ACIA?
000309                  STA         E.REG
000310                  BMI         PIO070
000311                  LDA         E.IFR               ;ANY INTERRUPT ON E-6522?
000312                  BPL         PIO020              ;  NO
000313                  AND         E.IER
000314                  LDY         #7
000315                  LDX         #$02
000316  PIO010          LSR         A                   ;CHECK FLAG BITS
000317                  BCS         PIO070
000318                  INX
000319                  DEY
```



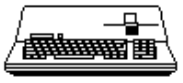
```
000320                  BNE         PIO010
000321                  BEQ         PIO035
000322  PIO020          LDA         D.IFR               ;ANY INTERRUPT ON D-6522?
000323                  BPL         PIO035
000324                  AND         D.IER
000325                  BIT         ANYSLOT             ;ANY SLOT INTERRUPT?
000326                  BNE         PIO040              ;  YES
000327                  LDY         #7
000328                  LDX         #$09
000329  PIO030          LSR         A                   ;CHECK FLAG BITS
000330                  BCS         PIO070
000331                  INX
000332                  DEY
000333                  BNE         PIO030
000334  PIO035          LDX         #$10                ;INTERRUPT NOT FOUND
000335                  BNE         PIO050
000336  PIO040          LDX         #$11
000337                  BIT         SLOT1               ;SLOT 1?
000338                  BPL         PIO070
000339                  INX
000340                  BIT         SLOT2               ;SLOT 2?
000341                  BPL         PIO070
000342                  LDA         E.IORA
000343                  INX
000344                  BIT         SLOT3               ;SLOT 3?
000345                  BEQ         PIO070
000346                  INX
000347                  BIT         SLOT4               ;SLOT 4?
000348                  BEQ         PIO070
000349                  LDX         #$0A
000350  *
000351  *  BAD INTERRUPT -- SYSTEM DEATH
000352  *
000353  PIO050          LDA         #>BADINT1           ;INTERRUPT NOT FOUND
000354                  JSR         SYSDEATH
000355  PIO060          LDA         #>BADINT2           ;BAD ZERO PAGE ALLOCATION
000356                  JSR         SYSDEATH
000357  *
000358  *  INTERRUPTING DEVICE FOUND
000359  *    ALLOCATE ZERO PAGE AND CALL MASTER INTERRUPT HANDLER
000360  *
000361  *  NOTE:
000362  *    SINCE READING THE ACIA'S STATUS REGISTER RESETS THE
000363  *    DSR AND DCD BITS, THE STATUS READ BY THE POLLING
000364  *    ROUTINE MUST BE PASSED TO THE INTERRUPT HANDLER;
000365  *    THE Y REGISTER HAS BEEN SELECTED FOR THIS PURPOSE.
000366  *    THE CURRENT IMPLEMENTATION DOES NOT USE Y IN CALLING
000367  *    THE INTERRUPT HANDLER.  IF SUBSEQUENT REVISIONS
000368  *    NEED TO USE Y, THE STATUS MUST BE PRESERVED AND
000369  *    RESTORED BEFORE CALLING THE INTERRUPT HANDLER.
000370  *
000371  CALLMIH         JMP         (IRQADDR)
000372  *
000373  PIO070          LDA         SIRTABLE,X          ;INTERRUPT ALLOCATED?
000374                  BPL         PIO050              ;  NO
000375                  LDA         SIRADR.L,X          ;GET INTERRUPT ADDRESS
000376                  STA         IRQADDR
000377                  ORA         SIRADR.H,X          ;CHECK FOR ADDRESS = $00
000378                  BEQ         PIO050              ;  BAD ADDRESS
000379                  LDA         SIRADR.H,X
000380                  STA         IRQADDR+1
000381                  LDA         SIRADR.B,X
000382                  STA         B.REG
000383                  LDA         ZPGSTACK            ;ALLOCATE MIH ZERO PAGE
000384                  CMP         #ZPGSTOP+ZPGSPACE
000385                  BCC         PIO060              ;TOO MANY NESTED INTERRUPTS
000386                  SBC         #ZPGSPACE
000387                  STA         ZPGSTACK
000388                  STA         ZPGSP
000389                  TAX
000390                  JSR         CALLMIH             ;CALL INTERRUPT HANDLER
000391                  SEI
000392                  LDA         #$00
000393                  STA         Z.REG
000394                  CLC
000395                  LDA         ZPGSTACK            ;DEALLOCATE MIH ZERO PAGE
000396                  ADC         #ZPGSPACE
000397                  STA         ZPGSTACK
000398                  STA         ZPGSP
000399                  LDA         #BITON1
000400                  STA         D.IFR               ;CLEAR ANY SLOT INTERRUPT
```



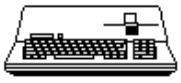
```
000401                  JMP         DISPATCH
000402                  SBTL        "NON-MASKABLE INTERRUPT RECEIVER"
000403                  REP         60
000404  *
000405  *  THIS IS THE NON-MASKABLE INTERRUPT RECEIVER.  WHEN AN
000406  *  NMI OCCURS, THE CPU PASSES CONTROL TO THE NMI RECEIVER
000407  *  THROUGH THE NMI VECTOR.  THE OPERATION OF THE NMI
000408  *  RECEIVER IS ESSENTIALLY THE SAME AS THE GIR EXCEPT
000409  *  THAT IT IS NOT CONCERNED WITH BRK, AND THE ONLY VALID
000410  *  SOURCE OF AN NMI IS THE KEYBOARD OR THE I/O DEVICE THAT
000411  *  HAS ALLOCATED THE NMI RESOURCE.
000412  *
000413                  REP         60
000414  *
000415  *
000416  NMI.RCVR        EQU         *
000417  *
000418  *  SAVE CPU REGISTERS A, X, & Y ON CURRENT STACK
000419  *
000420                  PHA
000421                  TXA
000422                  PHA
000423                  TYA
000424                  PHA
000425  *
000426  *  CHECK FOR STACK OVERFLOW
000427  *
000428                  TSX
000429                  CPX         #$FA
000430                  BCC         NMI005
000431                  LDA         #>STKOVFL
000432                  JSR         SYSDEATH
000433  *
000434  *  SET UP INTERRUPT ENVIRONMENT:
000435  *    BINARY ARITHMETIC, 2 MHZ, I/O ENABLED,
000436  *    RAM WRITE ENABLED, PRIMARY STACK,
000437  *    AND $F000 RAM SELECTED.  PRESERVE
000438  *    USER STATE OF SCREEN AND RESET LOCK.
000439  *
000440  NMI005          CLD
000441                  LDA         E.REG
000442                  TAX
000443                  AND         #BITON5+BITON4
000444                  ORA         #BITON6+BITON2
000445                  STA         E.REG
000446  *
000447  *  IF NOT ALREADY ON PRIMARY STACK, SAVE USER'S
000448  *  STACK POINTER AND SET UP SOS STACK POINTER.
000449  *
000450                  TXA
000451                  AND         #BITON2
000452                  BNE         NMI010
000453                  TXA
000454                  TSX
000455                  STX         SP.SAVE
000456                  LDX         #>E.SAVE
000457                  TXS
000458                  TAX
000459  *
000460  *  SAVE SYSTEM CONTROL REGISTERS E, Z, & B ON SOS STACK
000461  *
000462  NMI010          TXA
000463                  PHA
000464                  LDA         Z.REG
000465                  PHA
000466                  LDA         B.REG
000467                  PHA
000468                  LDA         EXPNSLOT
000469                  PHA
000470                  BIT         $CFFF
000471                  BIT         $C020               ;RESET I/O SPACE
000472                  LDA         #$00
000473                  STA         EXPNSLOT
000474  *
000475  *  SET INTERRUPT ZERO PAGE
000476  *
000477                  LDA         #0
000478                  STA         Z.REG
000479  *
000480  *  SEE IF NMI IS FROM KEYBOARD OR I/O DEVICE
000481  *
```



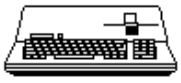
```
000482                  LDA         E.IORB
000483                  BMI         NMI030
000484  *
000485  *  NMI IS FROM I/O DEVICE
000486  *
000487                  LDA         SIRTABLE            ;NMI ALLOCATED?
000488                  BPL         NMI020
000489                  JSR         CALLNMI
000490                  SEI
000491                  JMP         DISPATCH
000492  CALLNMI         LDA         SIRADR.L
000493                  STA         NMIADR.L
000494                  LDA         SIRADR.B
000495                  STA         B.REG
000496                  JMP         (NMIADR.L)
000497  *
000498  *  BAD INTERRUPT -- SYSTEM DEATH
000499  *
000500  NMI020          LDA         #>BADINT1           ;NMI NOT ALLOCATED
000501                  JSR         SYSDEATH
000502  *
000503  *  NMI IS FROM THE KEYBOARD
000504  *
000505  NMI030          LDA         SYSBANK
000506                  STA         B.REG
000507                  JSR         KYBDNMI
000508                  SEI
000509                  JMP         DISPATCH
000510                  SBTL        "DISPATCHER"
000511                  REP         60
000512  *
000513  *  THIS IS THE DISPATCHER.  UPON COMPLETION, ALL SOS CALLS
000514  *  AND INTERRUPT HANDLERS RETURN CONTROL TO THE DISPATCHER.
000515  *  ITS PURPOSE IS TO SET UP THE APPROPRIATE ENVIRONMENT AND
000516  *  PASS CONTROL TO WHATEVER CODE SHOULD RUN NEXT.
000517  *
000518  *  WHEN SOS IS INTERRUPTED, CONTROL ALWAYS RETURNS TO THE
000519  *  INTERRUPTED CODE.  HOWEVER, WHEN THE USER IS INTERRUPTED,
000520  *  BY EITHER A SOS CALL OR AN INTERRUPT, THE DISPATCHER
000521  *  MUST CHECK THE EVENT QUEUE.  IF THERE IS AN ACTIVE EVENT
000522  *  WITH A PRIORITY HIGHER THAN THE CURRENT EVENT FENCE,
000523  *  CONTROL IS PASSED TO THE EVENT CODE.  OTHERWISE, CONTROL
000524  *  RETURNS TO THE INTERRUPTED CODE.
000525  *
000526                  REP         60
000527  *
000528  DISPATCH        EQU         *
000529  *
000530  *  DISABLE INTERRUPTS AND RESTORE
000531  *  SYSTEM CONTROL REGISTERS B & Z
000532  *
000533                  SEI
000534                  LDA         E.REG
000535                  ORA         #BITON6             ;ENABLE I/O
000536                  STA         E.REG
000537                  PLA
000538                  JSR         SELC800             ;RESTORE I/O SPACE
000539                  PLA
000540                  STA         B.REG
000541                  PLA
000542                  STA         Z.REG
000543  *
000544  *  CHECK SAVED ENVIRONMENT REGISTER
000545  *  IF RETURNING TO PRIMARY STACK
000546  *    THEN RESTORE E REG AND RELAUNCH SOS
000547  *    ELSE RESET STACK POINTER & RESTORE E REG
000548  *
000549                  PLA
000550                  ORA         #BITON5             ;SET SCREEN STATE TO
000551                  BIT         SCRNMODE            ;  CURRENT SCREEN MODE
000552                  BMI         DSP005
000553                  AND         #BITOFF5
000554  DSP005          TAY
000555                  AND         #BITON2
000556                  BEQ         DSP010
000557                  STY         E.REG
000558                  BNE         DSP030
000559  DSP010          PLA
000560                  TAX
000561                  TXS
000562                  STY         E.REG
```



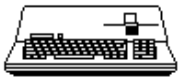
```
000563  *
000564  *  CHECK FOR ACTIVE EVENT WITH PRIORITY > FENCE
000565  *
000566  DSP020          LDA        CEVPRI
000567                  LDX        EVQ.LINK
000568                  CMP        EVQ.PRI,X
000569                  BCS        DSP030
000570  *
000571  *  PROCESS ACTIVE EVENT TRAP
000572  *  SAVE E, Z, B, & CALLER'S PRIORITY ON STACK THEN CALL
000573  *  EVENT.  UPON RETURN, RESTORE PRIORITY, B, Z, & E THEN
000574  *  CHECK FOR MORE EVENTS.
000575  *
000576                  LDA        E.REG
000577                  PHA
000578                  LDA        Z.REG
000579                  PHA
000580                  LDA        B.REG
000581                  PHA
000582                  LDA        CEVPRI
000583                  PHA
000584                  JSR        DO.EVENT
000585                  SEI
000586                  PLA
000587                  STA        CEVPRI
000588                  PLA
000589                  STA        B.REG
000590                  PLA
000591                  STA        Z.REG
000592                  PLA
000593                  ORA        #BITON5              ;SET SCREEN STATE TO
000594                  BIT        SCRNMODE             ;  CURRENT SCREEN MODE
000595                  BMI        DSP025
000596                  AND        #BITOFF5
000597  DSP025          STA        E.REG
000598                  JMP        DSP020
000599  *
000600  *  RESTORE CPU REGISTERS Y, X, & A AND LAUNCH
000601  *
000602  DSP030          PLA
000603                  TAY
000604                  PLA
000605                  TAX
000606                  PLA
000607                  RTI
000608                  PAGE
000609                  REP        60
000610  *
000611  *  THIS SUBROUTINE CALLS THE HIGHEST PRIORITY ACTIVE EVENT.
000612  *  FIRST, IT DELINKS THE FIRST ENTRY ON THE ACTIVE LIST AND
000613  *  LINKS IT TO THE FREE LIST.  THEN, IT SETS UP THE BANK,
000614  *  ADDRESS, ID, & STATUS AND CALLS THE EVENT VIA AN RTI.
000615  *
000616                  REP        60
000617  *
000618  DO.EVENT        EQU        *
000619  *
000620  *  WRITE ENABLE RAM
000621  *
000622                  LDY        E.REG
000623                  TYA
000624                  AND        #BITOFF3
000625                  STA        E.REG
000626  *
000627  *  DELINK ENTRY FROM ACTIVE LIST AND RELINK IT TO FREE LIST
000628  *
000629                  LDX        EVQ.LINK
000630                  LDA        EVQ.LINK,X
000631                  STA        EVQ.LINK
000632                  LDA        EVQ.FREE
000633                  STA        EVQ.LINK,X
000634                  STX        EVQ.FREE
000635  *
000636  *  SET FENCE TO EVENT PRIORITY THEN RESTORE E REG
000637  *
000638                  LDA        EVQ.PRI,X
000639                  STA        CEVPRI
000640                  STY        E.REG
000641  *
000642  *  SET UP B, Z, E, ADDRESS, ID, & STATUS
000643  *
```



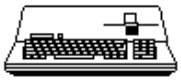
```
000644                  LDA         EVQ.BANK,X
000645                  STA         B.REG
000646                  LDA         EVQ.ADRH,X
000647                  PHA
000648                  LDA         EVQ.ADRL,X
000649                  PHA
000650                  LDY         EVQ.ID,X
000651                  PHP
000652                  PLA
000653                  AND         #$82
000654                  PHA
000655                  TYA
000656                  RTI
000657
000658                  CHN         IPL.SRC2
000659
000660  ********************************************************************************
000661  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC1
000662  ********************************************************************************
000663
000664

End of File -- Lines: 664 Characters: 17425
```



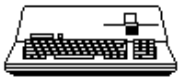
```
================================================================================
FILE: "SOS.IPL.SRC2.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC2
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SYSTEM INTERNAL RESOURCES"
000007                  REP        60
000008  *
000009  *  SYSTEM INTERNAL RESOURCE NUMBERS
000010  *
000011  *
000012  *  SIR  RESOURCE
000013  *
000014  *   0   SOUND PORT / I/O NMI
000015  *   1   ACIA
000016  *   2   E.CA2 -- KEYBOARD
000017  *   3   E.CA1 -- CLOCK
000018  *   4   E.SR
000019  *   5   E.CB2 -- VBL +
000020  *   6   E.CB1 -- VBL -
000021  *   7   E.T2
000022  *   8   E.T1
000023  *   9   D.CA2 -- CSP INPUT FLAG / INPUT SWITCH 1
000024  *   A   D.CA1 -- ANY SLOT (RESERVED FOR SOS)
000025  *   B   D.SR  -- CSP DATA REGISTER
000026  *   C   D.CB2 -- CSP DATA I/O / ENSIO
000027  *   D   D.CB1 -- CSP CLOCK / ENSEL / A/D SELECT / INPUT SW3
000028  *   E   D.T2
000029  *   F   D.T1
000030  *  10   DISK STEPPER / GRAPHICS SCROLL / CHARACTER DOWNLOAD
000031  *  11   SLOT 1
000032  *  12   SLOT 2
000033  *  13   SLOT 3
000034  *  14   SLOT 4
000035  *  15   (UNASSIGNED)
000036  *  16   (UNASSIGNED)
000037  *  17   (UNASSIGNED)
000038  *
000039                  REP        60
000040                  SBTL       "RESOURCE ALLOCATION & DEALLOCATION"
000041                  REP        60
000042  *
000043  *  RESOURCE ALLOCATION AND DEALLOCATION
000044  *
000045  *  SIRS ARE ALLOCATED AND DEALLOCATED BY THE SUBROUTINES
000046  *  'ALLOCSIR' AND 'DEALCSIR'.  THE RESOURCE PARAMETERS ARE
000047  *  PASSED IN A TABLE THAT CONTAINS ONE FIVE-BYTE ENTRY FOR
000048  *  EACH SIR THAT IS TO BE ALLOCATED OR DEALLOCATED.  THE
000049  *  TABLE ENTRY FORMAT IS SHOWN BELOW:
000050  *
000051  *           0       1       2       3       4
000052  *       +-------+-------+-------+-------+-------+
000053  *       | SIR # |  ID   | ADR.L | ADR.H | ADR.B |
000054  *       +-------+-------+-------+-------+-------+
000055  *
000056  *  SIR # -- SYSTEM INTERNAL RESOURCE NUMBER
000057  *  ID    -- IDENTIFICATION BYTE
000058  *           SUPPLIED BY ALLOCSIR, CHECKED BY DEALCSIR
000059  *  ADR   -- INTERRUPT ADDRESS (LOW, HIGH, BANK)
000060  *           ZERO IF NO INTERRUPT HANDLER
000061  *
000062  *
000063  *  ALLOCSIR -- ALLOCATE SYSTEM INTERNAL RESOURCES
000064  *
000065  *    PARAMETERS:
000066  *      A:  NUMBER OF BYTES IN TABLE
000067  *      X:  TABLE ADDRESS (LOW BYTE)
000068  *      Y:  TABLE ADDRESS (HIGH BYTE)
000069  *
000070  *    NORMAL EXIT -- SIRS ALLOCATED
000071  *      CARRY:  CLEAR
000072  *      A, X, Y:  UNDEFINED
000073  *
000074  *    ERROR EXIT -- SIRS NOT ALLOCATED
000075  *      CARRY:  SET
000076  *      X:  SIR NUMBER
```



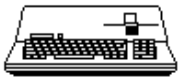
```
000077  *      A, Y:  UNDEFINED
000078  *
000079  *
000080  *  DEALCSIR -- DEALLOCATE SYSTEM INTERNAL RESOURCES
000081  *
000082  *    PARAMETERS:
000083  *      A:  NUMBER OF BYTES IN TABLE
000084  *      X:  TABLE ADDRESS (LOW BYTE)
000085  *      Y:  TABLE ADDRESS (HIGH BYTE)
000086  *
000087  *    NORMAL EXIT -- SIRS DEALLOCATED
000088  *      CARRY:  CLEAR
000089  *      A, X, Y:  UNDEFINED
000090  *
000091  *    ERROR EXIT -- SIRS NOT DEALLOCATED
000092  *      CARRY:  SET
000093  *      X:  SIR NUMBER
000094  *      A, Y:  UNDEFINED
000095  *
000096                  REP         60
000097                  PAGE
000098  *
000099  IDBYTE          DFB         $81
000100  *
000101  ALLOCSIR        EQU         *
000102                  CLC
000103                  PHP
000104                  SEI
000105                  STA         SIRARGSIZ           ;SAVE TABLE SIZE
000106                  LDA         E.REG
000107                  STA         SIRTEMP
000108                  ORA         #BITON2             ;FORCE PRIMARY STACK
000109                  AND         #BITOFF3            ;  AND WRITE ENABLE
000110                  STA         E.REG
000111                  LDA         SIRTEMP
000112                  PHA
000113                  LDA         Z.REG
000114                  PHA
000115                  LDA         #$00
000116                  STA         Z.REG               ;SET ZERO PAGE := $00
000117                  STX         SIRARGS
000118                  STY         SIRARGS+1           ;SET POINTER TO TABLE
000119  *
000120                  LDY         #$00
000121  ASIR010         LDA         (SIRARGS),Y         ;GET SIR NUMBER
000122                  CMP         #SIRTBLSIZ
000123                  TAX
000124                  BCS         ASIR020
000125                  LDA         SIRTABLE,X          ;CHECK ALLOCATION
000126                  BMI         ASIR020
000127                  LDA         IDBYTE
000128                  STA         SIRTABLE,X          ;ALLOCATE SIR
000129                  INY
000130                  STA         (SIRARGS),Y         ;RETURN ID BYTE
000131                  INY
000132                  LDA         (SIRARGS),Y
000133                  STA         SIRADR.L,X          ;SAVE INTERRUPT ADDRESS
000134                  INY
000135                  LDA         (SIRARGS),Y
000136                  STA         SIRADR.H,X
000137                  INY
000138                  LDA         (SIRARGS),Y
000139                  STA         SIRADR.B,X
000140                  INY
000141                  CPY         SIRARGSIZ
000142                  BCC         ASIR010
000143  *
000144                  CLC
000145                  INC         IDBYTE              ;BUMP ID BYTE
000146                  BMI         SIREXIT
000147                  LDA         #$81
000148                  STA         IDBYTE
000149                  BMI         SIREXIT
000150  *
000151  ASIR020         STX         SIRTEMP             ;SAVE BAD SIR NUMBER
000152  ASIR030         SEC
000153                  TYA
000154                  SBC         #5
000155                  TAY
000156                  BCC         ASIR040
000157                  LDA         (SIRARGS),Y         ;GET SIR NUMBER
```



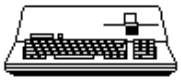
```
000158                  TAX
000159                  LDA       #FALSE
000160                  STA       SIRTABLE,X          ;RELEASE ALLOCATED SIRS
000161                  BEQ       ASIR030
000162  *
000163  ASIR040         LDX       SIRTEMP             ;RETURN BAD SIR
000164                  SEC
000165  *
000166  *
000167  *
000168  SIREXIT         PLA
000169                  STA       Z.REG               ;RESTORE Z REGISTER
000170                  PLA
000171                  STA       E.REG               ;RESTORE E REGISTER
000172                  BCC       SIREXIT1
000173                  PLA
000174                  ORA       #BITON0
000175                  PHA
000176  SIREXIT1        PLP
000177                  RTS
000178  *
000179  *
000180  *
000181  DEALCSIR        EQU       *
000182                  CLC
000183                  PHP
000184                  SEI
000185                  STA       SIRARGSIZ           ;SAVE TABLE SIZE
000186                  LDA       E.REG
000187                  STA       SIRTEMP
000188                  ORA       #BITON2             ;FORCE PRIMARY STACK
000189                  AND       #BITOFF3            ;  AND WRITE ENABLE
000190                  STA       E.REG
000191                  LDA       SIRTEMP
000192                  PHA
000193                  LDA       Z.REG
000194                  PHA
000195                  LDA       #$00
000196                  STA       Z.REG               ;SET ZERO PAGE := $00
000197                  STX       SIRARGS
000198                  STY       SIRARGS+1           ;SET POINTER TO TABLE
000199  *
000200                  LDY       #$00
000201  DSIR010         LDA       (SIRARGS),Y         ;GET SIR NUMBER
000202                  TAX
000203                  CPX       #SIRTBLSIZ
000204                  BCS       DSIR030
000205                  INY
000206                  LDA       SIRTABLE,X
000207                  BPL       DSIR030             ;VERIFY ALLOCATION
000208                  CMP       (SIRARGS),Y
000209                  BNE       DSIR030
000210                  INY
000211                  INY
000212                  INY
000213                  INY
000214                  CPY       SIRARGSIZ
000215                  BCC       DSIR010
000216  *
000217                  LDY       SIRARGSIZ
000218  DSIR020         SEC
000219                  TYA
000220                  SBC       #5
000221                  TAY
000222                  BCC       SIREXIT
000223                  LDA       (SIRARGS),Y         ;GET SIR NUMBER
000224                  TAX
000225                  LDA       #FALSE
000226                  STA       SIRTABLE,X
000227                  BEQ       DSIR020
000228  *
000229  DSIR030         SEC
000230                  BCS       SIREXIT
000231                  SBTL      "SELECT I/O EXPANSION ROM"
000232                  REP       60
000233  *
000234  *  SUBROUTINE 'SELC800' IS CALLED TO SELECT THE C800 I/O EX-
000235  *  PANSION ADDRESS SPACE FOR A PERIPHERAL SLOT.  ON ENTRY,
000236  *  THE SLOT NUMBER IS PASSED IN THE ACCUMULATOR.  IF NO
000237  *  ERROR OCCURS, CARRY IS CLEARED; OTHERWISE, CARRY IS SET
000238  *  AND THE PREVIOUS SLOT REMAINS SELECTED.
```



```
000239  *
000240  *  PARAMETERS:
000241  *    A:  SLOT NUMBER
000242  *
000243  *  NORMAL EXIT -- NEW SLOT SELECTED
000244  *    CARRY:  CLEAR
000245  *    A:  UNDEFINED
000246  *    X, Y:  UNCHANGED
000247  *
000248  *  ERROR EXIT -- SLOT NOT CHANGED
000249  *    CARRY:  SET
000250  *    A, X, Y:  UNCHANGED
000251  *
000252  *  WARNING !!!
000253  *    'SELC800' USES SELF-MODIFYING CODE!
000254  *
000255                  REP        60
000256  *
000257  SELC800         EQU        *
000258                  CMP        #$05                ;CHECK SLOT NUMBER
000259                  BCS        SC8EXIT             ;  INVALID
000260                  PHP
000261                  SEI
000262                  STA        EXPNSLOT
000263                  ORA        #$C0                ;MAKE SLOT INTO $CN00
000264                  STA        CNADDR+2            ;  AND MODIFY BIT ADDRESS
000265                  BIT        $C020
000266                  BIT        $CFFF               ;DESELECT PREVIOUS SLOT
000267  CNADDR          BIT        $C0FF               ;  AND SELECT CURRENT SLOT
000268                  PLP
000269  SC8EXIT         RTS
000270                  SBTL       "NMI DISABLE / ENABLE"
000271                  REP        60
000272  *
000273  *  THE SUBROUTINES NMIDSBL AND NMIENBL ARE CALLED TO
000274  *  DISABLE AND ENABLE NMI, RESPECTIVELY.  THERE ARE NO
000275  *  INPUT PARAMETERS.  ON EXIT, THE REGISTERS ARE UN-
000276  *  DEFINED.  NMIDSBL CLEARS THE CARRY FLAG IF NMI WAS
000277  *  SUCCESSFULLY DISABLED; OTHERWISE, CARRY IS SET.
000278  *
000279                  REP        60
000280  *
000281  NMIDSBL         EQU        *
000282                  LDX        E.REG
000283                  BIT        NMIFLAG
000284                  BPL        NDS020
000285                  TXA
000286                  ORA        #BITON7
000287                  STA        E.REG               ;SET 1MHZ
000288                  LDA        #$00
000289                  STA        NMICNTR
000290                  STA        NMICNTR+1
000291  NDS010          BIT        NMIFLAG             ;NMI PENDING?
000292                  BPL        NDS020              ;  NO
000293                  INC        NMICNTR             ;BUMP NMI COUNTER
000294                  BNE        NDS010              ;  AND RECHECK NMI FLAG
000295                  INC        NMICNTR+1
000296                  BNE        NDS010
000297                  LDA        #>NMIHANG           ;CAN'T LOCK NMI
000298                  JSR        SYSDEATH
000299  NDS020          TXA                            ;GET E.REG
000300                  AND        #BITOFF4            ;DISABLE NMI
000301                  STA        E.REG
000302                  RTS
000303  *
000304  *
000305  *
000306  NMIENBL         EQU        *
000307                  LDA        E.REG
000308                  ORA        #BITON4             ;ENABLE NMI
000309                  STA        E.REG
000310                  RTS
000311                  SBTL       "KEYBOARD NMI HANDLER"
000312                  REP        60
000313  *
000314  *  BY DEFAULT, KEYBOARD NMI IS IGNORED.  THE USER MAY
000315  *  PROCESS NMI BY CHANGING THE ADDRESS IN SYSTEM GLOBAL.
000316  *
000317                  REP        60
000318  *
000319  NMIDBUG         EQU        *
```

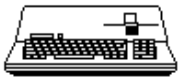


```
000320                  TSX                             ;SAVE THE STACK POINTER
000321                  STX         NMISPSV
000322                  LDA         #$03                ;SELECT MONITOR'S ZERO PAGE
000323                  STA         Z.REG
000324                  LDA         E.REG
000325                  ORA         #$03                ;SELECT MONITOR ROM
000326                  STA         E.REG
000327                  JSR         $F901               ;CALL THE MONITOR
000328  *
000329  NMICONT         EQU         *
000330                  LDA         E.REG
000331                  ORA         #BITON2             ;FORCE PRIMARY STACK
000332                  STA         E.REG
000333                  LDX         NMISPSV
000334                  TXS                             ;RESTORE STACK POINTER
000335                  RTS
000336                  SBTL        "EVENT QUEUE MANAGER"
000337                  REP         60
000338  *
000339  *  THE EVENT QUEUE IS USED TO HOLD THE PARAMETERS OF EVENTS
000340  *  THAT HAVE BEEN DETECTED BUT NOT YET RECOGNIZED.  EVENT
000341  *  QUEUE ENTRIES ARE ORGANIZED INTO TWO LINKED LISTS; A FREE
000342  *  LIST AND AN ACTIVE LIST.  EACH ENTRY IS SIX BYTES LONG,
000343  *  WITH THE FIRST BYTE (BYTE 0) USED AS A LINK.  THE LINK
000344  *  BYTE CONTAINS THE TABLE INDEX OF THE NEXT ENTRY IN THE
000345  *  LIST.  BECAUSE OF THE INDEXING METHOD, THE EVENT QUEUE
000346  *  MUST NOT EXCEED 256 BYTES.
000347  *
000348  *  ENTRY ZERO IS A SPECIAL ENTRY.  BYTE 0 IS THE INDEX OF
000349  *  THE FIRST ACTIVE ENTRY; BYTE 1 CONTAINS A ZERO, ALLOWING
000350  *  ENTRY 0 TO BE USED AS THE ACTIVE EVENT LIST TERMINATER;
000351  *  BYTE 2 CONTAINS THE INDEX OF THE FIRST FREE ENTRY; AND
000352  *  BYTES 4 THROUGH 6 ARE UNUSED.
000353  *
000354  *  THE FREE LIST IS LINKED LIFO.  THE ONLY VALID BYTE IN A
000355  *  FREE ENTRY IS THE LINK BYTE; THE REMAINING BYTES ARE
000356  *  UNDEFINED.  THE FREE LIST IS TERMINATED BY A LINK BYTE
000357  *  CONTAINING A ZERO.
000358  *
000359  *  THE ACTIVE LIST IS LINKED IN DECREASING PRIORITY ORDER
000360  *  WITH ENTRIES OF EQUAL PRIORITY LINKED FIFO.  BYTES 1
000361  *  THROUGH 5 CONTAIN THE EVENT PRIORITY, EVENT ID, LOW BYTE
000362  *  OF THE EVENT ADDRESS, HIGH BYTE OF THE EVENT ADDRESS, AND
000363  *  THE ADDRESS BANK.  THE ACTIVE LIST IS TERMINATED BY AN
000364  *  ENTRY WITH AN EVENT PRIORITY OF ZERO.
000365  *
000366                  REP         60
000367                  PAGE
000368                  REP         60
000369  *
000370  *  SUBROUTINE 'QUEEVENT' IS USED TO ENTER AN EVENT INTO THE
000371  *  EVENT QUEUE.  ACTIVE EVENTS ARE LINKED IN DECREASING
000372  *  PRIORITY ORDER WITH EVENTS OF EQUAL PRIORITY LINKED FIFO.
000373  *  EVENTS ARE REMOVED FROM THE QUEUE AS THEY ARE RECOGNIZED
000374  *  BY THE DISPATCHER.
000375  *
000376  *  PARAMETERS:
000377  *    X:  EVENT PARAMETER ADDRESS (LOW BYTE)
000378  *    Y:  EVENT PARAMETER ADDRESS (HIGH BYTE)
000379  *
000380  *    EVENT       0       1       2       3       4
000381  *    PARMS:  +-------+-------+-------+-------+-------+
000382  *            |  PRI  |   ID  | ADR.L | ADR.H | ADR.B |
000383  *            +-------+-------+-------+-------+-------+
000384  *            PRI:  EVENT PRIORITY
000385  *            ID:   EVENT ID BYTE
000386  *            ADR:  EVENT ADDRESS (LOW, HIGH, BANK)
000387  *
000388  *  EXIT CONDITIONS:
000389  *    CARRY:  CLEAR
000390  *    A, X, Y:  UNDEFINED
000391  *
000392                  REP         60
000393  *
000394  QUEEVENT        EQU         *
000395                  CLC
000396                  PHP
000397                  SEI
000398                  LDA         E.REG
000399                  STA         QEVTEMP
000400                  ORA         #BITON2             ;FORCE PRIMARY STACK
```

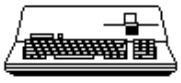


```
000401                  AND         #BITOFF3            ;  AND WRITE ENABLE
000402                  STA         E.REG
000403                  LDA         QEVTEMP
000404                  PHA
000405                  LDA         Z.REG
000406                  PHA
000407                  LDA         #0
000408                  STA         Z.REG               ;SET ZERO PAGE := 0
000409  *
000410                  STX         QEVARGS
000411                  STY         QEVARGS+1           ;SET ARGUMENT POINTER
000412                  LDY         #0
000413                  LDA         (QEVARGS),Y         ;GET PRIORITY
000414                  BEQ         Q.EXIT              ;  IGNORE IF ZERO
000415  *
000416                  LDX         EVQ.FREE
000417                  BEQ         Q.FULL
000418                  STX         QEV.THIS            ;GET FIRST FREE ENTRY
000419                  LDA         EVQ.LINK,X          ;  AND DELINK IT
000420                  STA         EVQ.FREE
000421  *
000422                  LDY         #EVQ.SIZ-2
000423  QEV010          LDA         (QEVARGS),Y         ;COPY ARGUMENTS
000424                  STA         EVQ.BANK,X          ;  INTO NEW ENTRY
000425                  DEX
000426                  DEY
000427                  BPL         QEV010
000428  *
000429                  LDX         QEV.THIS
000430                  LDY         #0
000431  QEV020          STY         QEV.LAST
000432                  LDA         EVQ.LINK,Y
000433                  TAY
000434                  LDA         EVQ.PRI,Y           ;SCAN EVENT QUEUE
000435                  CMP         EVQ.PRI,X           ;  FOR PROPER POSITION
000436                  BCS         QEV020
000437  *
000438                  TYA
000439                  STA         EVQ.LINK,X          ;RELINK EVENT INTO QUEUE
000440                  TXA
000441                  LDY         QEV.LAST
000442                  STA         EVQ.LINK,Y
000443  *
000444  Q.EXIT          PLA
000445                  STA         Z.REG               ;RESTORE Z REGISTER
000446                  PLA
000447                  STA         E.REG               ;RESTORE E REGISTER
000448                  PLP
000449                  RTS
000450  *
000451  Q.FULL          LDA         #>EVQOVFL           ;EVENT QUEUE OVERFLOW
000452                  JSR         SYSDEATH
000453                  LST         ON
000454
000455  ZZEND           EQU         *
000456  ZZLEN           EQU         ZZEND-ZZORG
000457                  IFNE        ZZLEN-LENIPL
000458                  FAIL        2,"SOSORG           FILE IS INCORRECT FOR IPL"
000459                  FIN
000460
000461  ********************************************************************************
000462  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC2
000463  ********************************************************************************
000464
000465
```

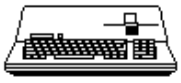
End of File -- Lines: 465 Characters: 12364



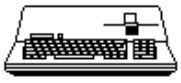
```
================================================================================
FILE: "SOS.LC.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: LC
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  IFNE       ZZLEN-LEN????
000007                  FAIL       2,"SOSORG            FILE IS INCORRECT FOR ??????"
000008                  FIN
000009
000010  ********************************************************************
000011  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: LC
000012  ********************************************************************
000013

End of File -- Lines: 13 Characters: 549
```



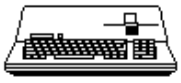
```
================================================================================================
FILE: "SOS.LCHK.TEXT"
================================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: LCHK
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  INCLUDE    SOSORG,6,1,254
000007                  ORG        ???????
000008  ---------------------------------
000009                  IFNE       ZZLEN-LEN????
000010                  FAIL       2,"SOSORG            FILE IS INCORRECT FOR ??????"
000011                  FIN
000012
000013  ***************************************************************************
000014  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: LCHK
000015  ***************************************************************************
000016

End of File -- Lines: 16 Characters: 643
```



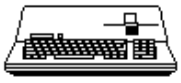
```
================================================================================
FILE: "SOS.MEMMGR.A.SRC.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.A.SRC
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1  MEMORY MANAGER"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGMEMMG
000010  ZZORG           EQU        *
000011                  MSB        OFF
000012                  REP        60
000013  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000014  *                     ALL RIGHTS RESERVED
000015                  REP        60
000016  *
000017  * MEMORY MANAGER (VERSION = 1.1O   )
000018  *                (DATE    = 8/04/81)
000019  *
000020  * THIS MODULE CONTAINS ALL OF THE MEMORY MANAGEMENT SYSTEM
000021  * CALLS SUPPORTED BY THE SARA OPERATING SYSTEM.  IT IS
000022  * ALSO CALLED BY THE BUFFER MANAGER.
000023  *
000024                  REP        60
000025  *
000026                  ENTRY      MMGR
000027  *
000028                  ENTRY      ST.CNT
000029                  ENTRY      ST.ENTRY
000030                  ENTRY      ST.FREE
000031                  ENTRY      ST.FLINK
000032                  ENTRY      VRT.LIM
000033  *
000034                  EXTRN      SYSERR
000035                  EXTRN      BADSCNUM
000036                  EXTRN      BADBKPG
000037                  EXTRN      SEGRQDN
000038                  EXTRN      SEGTBLFULL
000039                  EXTRN      BADSEGNUM
000040                  EXTRN      SEGNOTFND
000041                  EXTRN      BADSRCHMODE
000042                  EXTRN      BADCHGMODE
000043                  EXTRN      BADPGCNT
000044                  PAGE
000045                  REP        60
000046  *
000047  * SEGMENT TABLE
000048  * (NOTE: ENTRY 0 IS NOT USED)
000049  *
000050                  REP        60
000051  *
000052  ST.FREE         DS         1                   ; PTR TO FIRST FREE SEG TABLE ENTRY
000053  ST.ENTRY        DS         1                   ; PTR TO HIGHEST ALLOC SEG TABLE ENTRY
000054  ST.SIZ          EQU        7
000055  ST.CNT          EQU        32
000056  ST.TBL          DS         ST.SIZ*ST.CNT
000057  ST.BLINK        EQU        ST.TBL              ; BACK LINK TO PREV ALLOC SEG ENTRY
000058  ST.FLINK        EQU        ST.BLINK+ST.CNT     ; FORWARD LINK     "
000059  ST.BASEL        EQU        ST.FLINK+ST.CNT     ; BASE BANK/PAGE
000060  ST.BASEH        EQU        ST.BASEL+ST.CNT
000061  ST.LIML         EQU        ST.BASEH+ST.CNT     ; LIMIT BANK/PAGE
000062  ST.LIMH         EQU        ST.LIML+ST.CNT
000063  ST.ID           EQU        ST.LIMH+ST.CNT      ; SEG ID
000064                  PAGE
000065                  REP        60
000066  *
000067  * DATA DECLARATIONS
000068  *
000069                  REP        60
000070  *
000071  ZPAGE           EQU        $40                 ; BEGINNING OF ZPAGE TEMP SPACE FOR MEMORY MANAGER
000072  VRT.BASE        EQU        $0                  ; INTERNAL BK/PG PTR TO LOWEST VIRT PAGE
000073  VRT.LIM         EQU        ZPAGE+$0            ; &$1, INTERNAL BK/PG PTR TO HIGHEST VIRT PAGE
000074  PHY1BASE        EQU        $0780               ; BANK "F",PAGE "0"
000075  PHY1LIM         EQU        $079F               ; BANK "F",PAGE "1F"
000076  PHY2BASE        EQU        $0820               ; BANK "10",PAGE "A0"
```



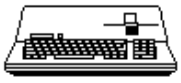
```
000077  PHY2LIM         EQU        $087F               ; BANK "10",PAGE "FF"
000078  *
000079  * REQUEST.SEG DATA DECLARATIONS
000080  *
000081  M.TPARMX        EQU        $60                 ; BEGINNING ADDRESS OF MMGR SOS CALL PARMS
000082  M.RQCODE        EQU        M.TPARMX
000083  RQ.BASE         EQU        M.TPARMX+1          ; BASE.BANK/PAGE
000084  RQ.LIM          EQU        M.TPARMX+3          ; LIMIT.BANK/PAGE
000085  RQ.ID           EQU        M.TPARMX+5
000086  RQ.NUM          EQU        M.TPARMX+6
000087  *
000088  RQ.REGION       EQU        ZPAGE+$2            ;VRT(0),PHY0(1),PHY1(2)
000089  *
000090  * FIND.SEG DATA DECLARATIONS
000091  *
000092  SRCHMODE        EQU        M.TPARMX+1          ; SEARCH MODE (0,1,2)
000093  F.ID            EQU        M.TPARMX+2          ; SEG ID
000094  F.PGCT          EQU        M.TPARMX+3          ; PAGE COUNT (LO
000095  FX.PGCT         EQU        ZPAGE+$3            ; &$4, INTERNAL PAGE COUNT
000096  F.BASE          EQU        M.TPARMX+5          ; BASE.BANK/PAGE
000097  F.LIM           EQU        M.TPARMX+7          ; LIMIT.BANK/PAGE
000098  F.NUM           EQU        M.TPARMX+9          ; SEG NUM
000099  F.ERR           EQU        ZPAGE+$5            ; ERROR FLAG
000100  TRUE            EQU        $80
000101  FALSE           EQU        $0
000102  CFS.PGCT        EQU        ZPAGE+$6            ; &7, CURRENT FREE SEGMENT'S PAGE COUNT
000103  CFS.BASE        EQU        ZPAGE+$8            ; &9,        "             BASE.BANK/PAGE
000104  CFS.LIM         EQU        ZPAGE+$A            ; &$B,       "             LIMIT.BANK/PAGE
000105  CFS.BLINK       EQU        ZPAGE+$C            ;          "             BACK LINK
000106  CFS.BASE0       EQU        ZPAGE+$D            ; &$E,     "             BASE (SMODE=0)
000107  CFS.BASE1       EQU        ZPAGE+$F            ; &$10,     "             BASE (SMODE=1)
000108  CFS.NEXT        EQU        ZPAGE+$11           ;          "             NEXT ENTRY
000109  CFS.PREV        EQU        ZPAGE+$12           ;          "             PREV ENTRY
000110  CFS.PTR         EQU        ZPAGE+$13           ; &$14     "             POINTER TO NXT FREE PG
000111  BFS.PGCT        EQU        ZPAGE+$15           ; &$16, BIGGEST FREE SEGMENT'S PAGE COUNT
000112  BFS.BASE        EQU        ZPAGE+$17           ; &$18     "             BASE.BANK/PAGE
000113  BFS.LIM         EQU        ZPAGE+$19           ; &$1A     "             LIMIT.BANK/PAGE
000114  BFS.BLINK       EQU        ZPAGE+$1B           ;          "             BACK LINK
000115  *
000116  * CHANGE.SEG DATA DECLARATIONS
000117  *
000118  CHG.NUM         EQU        M.TPARMX+1          ; SEGNUM PARM
000119  CHG.MODE        EQU        M.TPARMX+2          ; CHANGE MODE PARM
000120  CHG.PGCT        EQU        M.TPARMX+3          ; PAGE COUNT PARM
000121  CHG.PGCTX       EQU        ZPAGE+$1C           ; &$1D, INTERNAL STORE FOR PGCT
000122  CHG.NEW         EQU        ZPAGE+$1E           ; &$1F, BANK/PAGE OF SEG'S NEW LIMIT OR BASE
000123  *
000124  * GET.SEG.INFO DATA DECLARATIONS
000125  *
000126  GSI.NUM         EQU        M.TPARMX+1
000127  GSI.BASE        EQU        M.TPARMX+2
000128  GSI.LIM         EQU        M.TPARMX+4
000129  GSI.PGCT        EQU        M.TPARMX+6
000130  GSI.ID          EQU        M.TPARMX+8
000131  *
000132  * GET.SEG.NUM DATA DECLARATIONS
000133  *
000134  GSN.BKPG        EQU        M.TPARMX+1
000135  GSN.NUM         EQU        M.TPARMX+3
000136  *
000137  * RELEASE.SEG DATA DECLARATIONS
000138  *
000139  RLS.NUM         EQU        M.TPARMX+1          ; SEG NUM
000140  *
000141  * REGION - DATA DECLARATIONS
000142  *
000143  RGN.BKPG        DS         2                   ; TEMP CONTAINER FOR BANK/PAGE
000144                  PAGE
000145                  REP        60
000146  *
000147  * MMGR
000148  *
000149  * THIS ROUTINE IS THE MAIN ENTRANCE TO THE MEMORY MANAGER
000150  * MODULE.  IT FUNCTIONS AS A SWITCH, BASED UPON THE RECEIVED
000151  * REQUEST CODE, TO TRANSFER CONTROL TO THE ROUTINE THAT
000152  * HANDLES THE SPECIFIC SYSTEM CALL.
000153  *
000154                  REP        60
000155  *
000156  MMGR            EQU        *
000157                  LDA        M.RQCODE
```



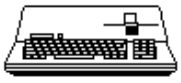
```
000158                  BEQ         MMGR010             ; "REQ.SEG"
000159                  CMP         #1
000160                  BEQ         MMGR020             ; "FIND.SEG"
000161                  CMP         #2
000162                  BEQ         MMGR030             ; "CHANGE.SEG"
000163                  CMP         #3
000164                  BEQ         MMGR040             ; "GET.SEG.INFO"
000165                  CMP         #4
000166                  BEQ         MMGR050             ; "GET.SEG.NUM"
000167                  CMP         #5
000168                  BEQ         MMGR060             ; "RELEASE.SEG"
000169  *
000170                  LDA         #BADSCNUM
000171                  JSR         SYSERR
000172  *
000173  MMGR010         JMP         REQ.SEG
000174  MMGR020         JMP         FIND.SEG
000175  MMGR030         JMP         CHG.SEG
000176  MMGR040         JMP         GET.SEG.INFO
000177  MMGR050         JMP         GET.SEG.NUM
000178  MMGR060         JMP         RELEASE.SEG
000179                  PAGE
000180                  REP         60
000181  *
000182  * REQUEST.SEG(IN.BASE.BANKPAGE,LIMIT.BANKPAGE,SEGID; OUT.SEGNUM)
000183  *
000184                  REP         60
000185  *
000186  REQ.SEG         EQU         *
000187  *
000188  * CONVERT CALLER'S BASE.BANK/PAGE TO INTERNAL FMT
000189  *
000190                  LDX         RQ.BASE
000191                  LDY         RQ.BASE+1
000192                  JSR         CNVRT.IBP
000193                  BCC         RQ005
000194  *
000195  RQ.ERR          RTS                             ; ERR EXIT - INVALID BANK/PAGE
000196  *
000197  RQ005           STX         RQ.BASE
000198                  STY         RQ.BASE+1
000199                  STA         RQ.REGION
000200  *
000201  * CONVERT CALLER'S LIMIT.BANK/PAGE TO INTERNAL FMT
000202  *
000203                  LDX         RQ.LIM
000204                  LDY         RQ.LIM+1
000205                  JSR         CNVRT.IBP
000206                  BCS         RQ.ERR              ; ERR - INVALID BANK/PAGE
000207                  STX         RQ.LIM
000208                  STY         RQ.LIM+1
000209  *
000210  * IF BASE AND LIMIT ARE IN DIFFERENT REGIONS THEN ERR
000211  *
000212                  CMP         RQ.REGION
000213                  BNE         RQ.ERR1             ; ERR - INVALID BANK/PAGE PAIR
000214  * IF CALLER'S BASE > LIMIT THEN ERR
000215  *
000216                  LDA         RQ.LIM
000217                  CMP         RQ.BASE
000218                  LDA         RQ.LIM+1
000219                  SBC         RQ.BASE+1
000220                  BCC         RQ.ERR1             ; ERR - INVALID BANK/PAGE PAIR
000221  *
000222  * PREV SEGNUM:=NULL; NEXT SEGNUM:=FIRST ENTRY
000223  *
000224                  LDX         #0
000225                  LDY         ST.ENTRY            ; NOTE: PREV/NEXT CARRIED IN X & Y REGISTERS
000226  *
000227  * IF NO SEGS IN SEG TABLE THEN ALLOCATE REQUESTED SEG
000228  *
000229                  BEQ         RQ030
000230  *
000231  * IF FIRST SEG IN SEG TABLE BELOW REQUESTED SEG
000232  * THEN ALLOCATE SEG
000233  *
000234                  LDA         ST.LIML,Y
000235                  CMP         RQ.BASE
000236                  LDA         ST.LIMH,Y
000237                  SBC         RQ.BASE+1
000238                  BCC         RQ030
```



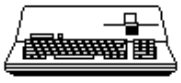
```
000239  *
000240  * ADVANCE TO NEXT SEG ENTRY
000241  *
000242  RQ010           TYA
000243                  TAX
000244                  LDA         ST.FLINK,Y
000245                  TAY
000246  *
000247  * IF THERE IS NO NEXT SEG ENTRY
000248  *   IF REQUESTED SEG IS BELOW PREV SEG
000249  *      THEN ALLOCATE REQ SEG
000250  *      ELSE ERR
000251  *
000252                  BNE         RQ020
000253                  LDA         RQ.LIM
000254                  CMP         ST.BASEL,X
000255                  LDA         RQ.LIM+1
000256                  SBC         ST.BASEH,X
000257                  BCC         RQ030
000258  *
000259                  BCS         RQ.ERR2             ; ERR - SEGMENT REQUEST DENIED
000260  *
000261  * IF REQUESTED LIMIT >= PREV SEG'S BASE THEN ERR
000262  *
000263  RQ020           LDA         RQ.LIM
000264                  CMP         ST.BASEL,X
000265                  LDA         RQ.LIM+1
000266                  SBC         ST.BASEH,X
000267                  BCS         RQ.ERR2             ; ERR - SEGMENT REQUEST DENIED
000268  *
000269  * IF REQUESTED BASE > NEXT SEG'S LIMIT
000270  *    THEN ALLOCATE REQUESTED SEGMENT
000271  *
000272                  LDA         ST.LIML,Y
000273                  CMP         RQ.BASE
000274                  LDA         ST.LIMH,Y
000275                  SBC         RQ.BASE+1
000276                  BCS         RQ010               ; NO, ADVANCE TO NEXT SEGMENT
000277  *
000278  RQ030           TXA                             ; ALLOCATE REQUESTED SEGMENT
000279                  JSR         GET.FREE
000280                  BCS         RQ.ERR3             ; ERR - SEG TABLE FULL
000281  *
000282  * ENTER BASE,LIMIT AND ID IN NEW SEG ENTRY
000283  *
000284                  TAX
000285                  LDA         RQ.BASE
000286                  STA         ST.BASEL,X
000287                  LDA         RQ.BASE+1
000288                  STA         ST.BASEH,X
000289  *
000290                  LDA         RQ.LIM
000291                  STA         ST.LIML,X
000292                  LDA         RQ.LIM+1
000293                  STA         ST.LIMH,X
000294  *
000295                  LDA         RQ.ID
000296                  STA         ST.ID,X
000297  *
000298  * RETURN NEW SEG NUM TO CALLER AND RETURN
000299  *
000300                  LDY         #0
000301                  TXA
000302                  STA         (RQ.NUM),Y
000303  *
000304                  CLC
000305                  RTS                             ; NORMAL EXIT
000306  *
000307  RQ.ERR1         LDA         #BADBKPG
000308                  JSR         SYSERR              ; ERR EXIT
000309  RQ.ERR2         LDA         #SEGRQDN
000310                  JSR         SYSERR              ; ERR EXIT
000311  *
000312  RQ.ERR3         LDA         #SEGTBLFULL
000313                  JSR         SYSERR              ; ERR EXIT
000314                  PAGE
000315                  REP         60
000316  *
000317  * FIND.SEG(IN.SRCHMODE,SEGID; INOUT.PAGECT;
000318  *          OUT.BASE.BKPG,LIMIT.BKPG,SEGNUM)
000319  *
```



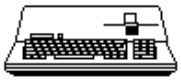
```
000320                  REP         60
000321  *
000322  FIND.SEG        EQU         *
000323  *
000324  * RETRIEVE PAGE COUNT PARAMETER AND CLEAR ERR FLAG
000325  *
000326                  LDY         #0
000327                  LDA         (F.PGCT),Y
000328                  STA         FX.PGCT
000329                  INY
000330                  LDA         (F.PGCT),Y
000331                  STA         FX.PGCT+1
000332  *
000333                  BNE         FIND001
000334                  LDA         FX.PGCT
000335                  BNE         FIND001
000336                  LDA         #BADPGCNT           ; ERR, PAGECT=0, EXIT
000337                  JSR         SYSERR
000338  *
000339  FIND001         LDA         #FALSE
000340                  STA         F.ERR
000341  *
000342  * IF SEARCH MODE>2 THEN ERR
000343  *
000344                  LDA         SRCHMODE
000345                  CMP         #3
000346                  BCC         FIND005
000347                  LDA         #BADSRCHMODE
000348                  JSR         SYSERR              ; ERR EXIT
000349  *
000350  * INITIALIZE NEXT FREE SEGMENT SUBROUTINE,
000351  * AND BIGGEST FREE SEGMENT PAGE COUNT
000352  *
000353  FIND005         JSR         NXTFRSEG.I
000354                  LDA         #0
000355                  STA         BFS.PGCT
000356                  STA         BFS.PGCT+1
000357  *
000358  * GET NEXT FREE SEGMENT
000359  *
000360  FIND010         JSR         NXTFRSEG
000361                  BCC         FIND015             ; PROCESS FREE SEGMENT
000362  *
000363  * NO MORE FREE SEGMENTS LEFT
000364  * RETURN BIGGEST FREE SEGMENT FOUND
000365  * ALONG WITH ERR
000366  *
000367                  LDA         #TRUE
000368                  STA         F.ERR
000369                  LDX         #0                  ; SEG#:=0
000370                  JMP         FIND070
000371  *
000372  * FREE SEGMENT FOUND.
000373  *   IF FREE SEGMENT > BIGGEST FREE SEGMENT THEN BFS:=CFS
000374  *
000375  FIND015         LDA         BFS.PGCT
000376                  CMP         CFS.PGCT
000377                  LDA         BFS.PGCT+1
000378                  SBC         CFS.PGCT+1
000379                  BCS         FIND030
000380  *
000381                  LDX         #6
000382  FIND020         LDA         CFS.PGCT,X
000383                  STA         BFS.PGCT,X
000384                  DEX
000385                  BPL         FIND020
000386  *
000387  * IF BFS.PGCT<F.PGCT THEN GET NEXT FREE SEGMENT
000388  *
000389  FIND030         LDA         BFS.PGCT
000390                  CMP         FX.PGCT
000391                  LDA         BFS.PGCT+1
000392                  SBC         FX.PGCT+1
000393                  BCC         FIND010
000394  *
000395  * BFS.BASE:=BFS.LIM-FX.PGCT+1
000396  * BFS.PGCT:=FX.PGCT
000397  *
000398                  LDA         BFS.LIM
000399                  SBC         FX.PGCT
000400                  STA         BFS.BASE
```



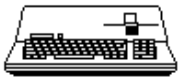
```
000401                  LDA         BFS.LIM+1
000402                  SBC         FX.PGCT+1
000403                  STA         BFS.BASE+1
000404                  INC         BFS.BASE
000405                  BNE         FIND050
000406                  INC         BFS.BASE+1
000407  *
000408  FIND050         LDA         FX.PGCT
000409                  STA         BFS.PGCT
000410                  LDA         FX.PGCT+1
000411                  STA         BFS.PGCT+1
000412  *
000413  * DELINK ENTRY FROM FREE LIST, AND LINK
000414  * IT INTO SEGMENT LIST
000415  *
000416                  LDA         BFS.BLINK
000417                  JSR         GET.FREE
000418                  BCC         FIND060
000419                  RTS                             ; ERR - SEG TABLE FULL
000420  *
000421  * ST.ID(NEW):=F.ID
000422  * ST.BASE(NEW):=BFS.BASE
000423  * ST.LIM(NEW):=BFS.LIM
000424  *
000425  FIND060         TAX
000426                  LDA         F.ID
000427                  STA         ST.ID,X
000428  *
000429                  LDA         BFS.BASE
000430                  STA         ST.BASEL,X
000431                  LDA         BFS.BASE+1
000432                  STA         ST.BASEH,X
000433  *
000434                  LDA         BFS.LIM
000435                  STA         ST.LIML,X
000436                  LDA         BFS.LIM+1
000437                  STA         ST.LIMH,X
000438  *
000439  * RETURN SEGNUM, PAGE COUNT, BASE BANK/PAGE, AND LIMIT BANK/PAGE
000440  * TO CALLER
000441  FIND070         LDY         #0
000442                  TXA
000443                  STA         (F.NUM),Y
000444  *
000445                  LDA         BFS.PGCT
000446                  STA         (F.PGCT),Y
000447                  INY
000448                  LDA         BFS.PGCT+1
000449                  STA         (F.PGCT),Y
000450  *
000451                  LDX         BFS.BASE
000452                  LDY         BFS.BASE+1
000453                  JSR         CNVRT.XBP
000454                  TYA
000455                  LDY         #1
000456                  STA         (F.BASE),Y
000457                  DEY
000458                  TXA
000459                  STA         (F.BASE),Y
000460  *
000461                  LDX         BFS.LIM
000462                  LDY         BFS.LIM+1
000463                  JSR         CNVRT.XBP
000464                  TYA
000465                  LDY         #1
000466                  STA         (F.LIM),Y
000467                  DEY
000468                  TXA
000469                  STA         (F.LIM),Y
000470  *
000471                  LDA         F.ERR               ; IF ERR FLAG TRUE THEN REPORT IT.
000472                  BNE         FIND.ERR
000473  *
000474                  CLC
000475                  RTS                             ; NORMAL EXIT
000476  *
000477  FIND.ERR        LDA         #SEGRQDN
000478                  JSR         SYSERR              ; ERR EXIT
000479
000480                  CHN         MEMMGR.B.SRC
000481
```



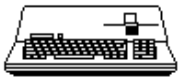
```
000482  ********************************************************************
000483  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.A.SRC
000484  ********************************************************************
000485
000486

End of File -- Lines: 486 Characters: 12629
```



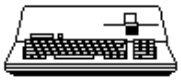
```
================================================================================
FILE: "SOS.MEMMGR.B.SRC.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.B.SRC
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP         60
000008  *
000009  * NEXT FREE SEGMENT - INITIALIZATION
000010  *
000011  * INPUT:  SEGMENT TABLE
000012  * OUTPUT: CFS.PTR "1ST FREE BANK/PAGE IN VIRTUAL MEMORY
000013  *         CFS.PREV "PREVIOUS SEGMENT EXAMINED"
000014  *         CFS.NEXT "SEGMENT FOLLOWING CFS.PREV"
000015  * ERROR:  NONE (IF NO FREE BK/PG FOUND, THEN CFS.PTR="FFFF")
000016  *
000017                  REP         60
000018  *
000019  NXTFRSEG.I      EQU         *
000020  *
000021  * CFS.PTR := VRT.LIM
000022  * CFS.PREV := 0
000023  * CFS.NEXT := ST.ENTRY
000024  *
000025                  LDA         >VRT.LIM
000026                  STA         CFS.PTR
000027                  LDA         >VRT.LIM+1
000028                  STA         CFS.PTR+1
000029  *
000030                  LDA         #0
000031                  STA         CFS.PREV
000032  *
000033                  LDX         ST.ENTRY
000034                  STX         CFS.NEXT
000035  *
000036  * L0:  IF CFS.NEXT=0 THEN DONE
000037  *
000038  FRSGI010        BEQ         FRSGI.EXIT
000039  *
000040  * IF ST.LIM(CFS.NEXT)<=VRT.LIM THEN GOTO L1
000041  *
000042                  LDA         >VRT.LIM
000043                  CMP         ST.LIML,X
000044                  LDA         >VRT.LIM+1
000045                  SBC         ST.LIMH,X
000046                  BCS         FRSGI020
000047  *
000048  * CFS.PREV:=CFS.NEXT
000049  * CFS.NEXT:=ST.FLINK(CFS.NEXT)
000050  * GOTO L0
000051  *
000052                  STX         CFS.PREV
000053                  LDA         ST.FLINK,X
000054                  TAX
000055                  STX         CFS.NEXT
000056                  JMP         FRSGI010
000057  *
000058  * L1:  IF ST.LIM(CFS.NEXT)<VRT.LIM THEN DONE
000059  *
000060  FRSGI020        LDA         ST.LIML,X
000061                  CMP         >VRT.LIM
000062                  LDA         ST.LIMH,X
000063                  SBC         >VRT.LIM+1
000064                  BCC         FRSGI.EXIT
000065  *
000066  *
000067                  JSR         NXTFRPG
000068  *
000069  FRSGI.EXIT      RTS                             ; NORMAL EXIT
000070                  PAGE
000071                  REP         60
000072  *
000073  * NEXT FREE SEGMENT
000074  *
000075  * INPUT:  SEG TABLE
000076  * OUTPUT: CFS.BLINK
```



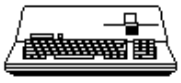
```
000077  *         CFS.BASE
000078  *         CFS.LIMIT
000079  *         CFS.PGCT
000080  * OWN:    CFS.PREV
000081  *         CFS.NEXT
000082  *         CFS.PTR
000083  *
000084  * BUILDS A CANDIDATE FREE SEGMENT, WHOSE LIMIT BANK/PAGE =
000085  * THE CURRENT FREE PAGE (CFS.PTR).
000086  *
000087                  REP        60
000088  *
000089  NXTFRSEG        EQU        *
000090  *
000091  * IF CFS.PTR="FFFF" THEN EXIT
000092  *
000093                  LDA        CFS.PTR+1
000094                  BPL        FRSG010
000095  *
000096                  SEC
000097                  RTS                           ; EXIT - NO MORE FREE SEGMENTS LEFT
000098  *
000099  * CFS.BLINK:=CFS.PREV
000100  * CFS.LIM:=CFS.PTR
000101  *
000102  FRSG010         LDA        CFS.PREV
000103                  STA        CFS.BLINK
000104  *
000105                  LDA        CFS.PTR
000106                  STA        CFS.LIM
000107                  LDA        CFS.PTR+1
000108                  STA        CFS.LIM+1
000109  *
000110  * IF CFS.NEXT=0 THEN CFS.BASE:=0
000111  *    ELSE CFS.BASE:=ST.LIM(CFS.NEXT)+1
000112  *
000113                  LDA        CFS.NEXT
000114                  BNE        FRSG020
000115                  LDA        #0
000116                  STA        CFS.BASE
000117                  STA        CFS.BASE+1
000118                  BEQ        FRSG030
000119  *
000120  FRSG020         LDX        CFS.NEXT
000121                  CLC
000122                  LDA        ST.LIML,X
000123                  ADC        #1
000124                  STA        CFS.BASE
000125                  LDA        ST.LIMH,X
000126                  ADC        #0
000127                  STA        CFS.BASE+1
000128  *
000129  * CFS.BASE0:=CFS.LIM AND $FF80
000130  *
000131  FRSG030         LDY        CFS.LIM+1
000132                  STY        CFS.BASE0+1
000133                  LDA        CFS.LIM
000134                  AND        #$80
000135                  STA        CFS.BASE0
000136  *
000137  * CFS.BASE1:=CFS.BASE0-32K
000138  *
000139                  SEC
000140                  SBC        #$80
000141                  STA        CFS.BASE1
000142                  TYA
000143                  SBC        #0
000144                  STA        CFS.BASE1+1
000145                  BCS        FRSG035
000146                  LDA        #0
000147                  STA        CFS.BASE1
000148                  STA        CFS.BASE1+1
000149  *
000150  * IF CFS.BASE>=CFS.BASE0 THEN GOTO L1
000151  *
000152  FRSG035         LDA        CFS.BASE
000153                  CMP        CFS.BASE0
000154                  LDA        CFS.BASE+1
000155                  SBC        CFS.BASE0+1
000156                  BCS        FRSG050
000157  *
```



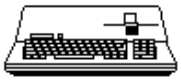
```
000158  * IF SEARCH MODE=0 THEN CFS.BASE:=CFS.BASE0
000159  * GOTO L1
000160  *
000161                  LDA         SRCHMODE
000162                  BNE         FRSG040
000163                  LDA         CFS.BASE0
000164                  STA         CFS.BASE
000165                  LDA         CFS.BASE0+1
000166                  STA         CFS.BASE+1
000167                  JMP         FRSG050
000168  *
000169  * IF CFS.BASE<CFS.BASE1 AND SEARCH MODE=1
000170  *    THEN CFS.BASE:=CFS.BASE1
000171  *
000172  FRSG040         LDA         CFS.BASE
000173                  CMP         CFS.BASE1
000174                  LDA         CFS.BASE+1
000175                  SBC         CFS.BASE1+1
000176                  BCS         FRSG050
000177  *
000178                  LDA         SRCHMODE
000179                  CMP         #1
000180                  BNE         FRSG050
000181  *
000182                  LDA         CFS.BASE1
000183                  STA         CFS.BASE
000184                  LDA         CFS.BASE1+1
000185                  STA         CFS.BASE+1
000186  *
000187  * L1:  CFS.PGCT:=CFS.LIM-CFS.BASE+1
000188  *
000189  FRSG050         SEC
000190                  LDA         CFS.LIM
000191                  SBC         CFS.BASE
000192                  STA         CFS.PGCT
000193                  LDA         CFS.LIM+1
000194                  SBC         CFS.BASE+1
000195                  STA         CFS.PGCT+1
000196                  INC         CFS.PGCT
000197                  BNE         FRSG052
000198                  INC         CFS.PGCT+1
000199  *
000200  * ADVANCE FREE PAGE POINTER TO NEXT FREE PAGE
000201  *
000202  * IF SEARCH MODE<>1 THEN L2:
000203  *
000204  FRSG052         LDA         SRCHMODE
000205                  CMP         #1
000206                  BNE         FRSG060
000207  *
000208  * IF CFS.BASE < CFS.BASE0 THEN CFS.PTR:=CFS.BASE0-1
000209  *
000210                  LDA         CFS.BASE
000211                  CMP         CFS.BASE0
000212                  LDA         CFS.BASE+1
000213                  SBC         CFS.BASE0+1
000214                  BCS         FRSG060
000215  *
000216                  LDY         CFS.BASE0+1
000217                  LDX         CFS.BASE0
000218                  BNE         FRSG055
000219                  DEY
000220  FRSG055         DEX
000221                  STX         CFS.PTR
000222                  STY         CFS.PTR+1
000223  *
000224                  JMP         FRSG070             ; AND EXIT
000225  * L2: CFS.PTR:=CFS.BASE-1
000226  *
000227  FRSG060         SEC
000228                  LDA         CFS.BASE
000229                  SBC         #1
000230                  STA         CFS.PTR
000231                  LDA         CFS.BASE+1
000232                  SBC         #0
000233                  STA         CFS.PTR+1
000234  *
000235  * IF CFS.PTR="FFFF" OR CFS.NEXT=0 THEN EXIT
000236  *
000237                  BCC         FRSG070
000238                  LDA         CFS.NEXT
```



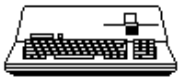
```
000239                  BEQ         FRSG070
000240  *
000241  * IF CFS.PTR > ST.LIM(CFS.NEXT) THEN EXIT
000242  *
000243                  LDX         CFS.NEXT
000244                  LDA         ST.LIML,X
000245                  CMP         CFS.PTR
000246                  LDA         ST.LIMH,X
000247                  SBC         CFS.PTR+1
000248                  BCC         FRSG070
000249  *
000250  * OTHERWISE, ADVANCE CFS PTR TO NEXT FREE PAGE BELOW NEXT
000251  * SEGMENT IN SEGMENT LIST
000252  *
000253                  JSR         NXTFRPG
000254  *
000255  FRSG070         CLC
000256                  RTS                             ; EXIT - FREE SEGMENT FOUND
000257                  PAGE
000258                  REP         60
000259  *
000260  * NEXT FREE PAGE
000261  *
000262  * "WALKS" THE FREE PAGE PTR (CFS.PTR) TO THE NEXT FREE PAGE
000263  * IMMEDIATELY BELOW THE CURRENT FREE SEGMENT.
000264  *
000265                  REP         60
000266  *
000267  NXTFRPG         EQU         *
000268  *
000269  * L0: CFS.PTR:=ST.BASE(CFS.NEXT)-1
000270  *     IF CFS.PTR="FFFF" THEN DONE
000271  *
000272                  LDX         CFS.NEXT
000273                  SEC
000274                  LDA         ST.BASEL,X
000275                  SBC         #1
000276                  STA         CFS.PTR
000277                  LDA         ST.BASEH,X
000278                  SBC         #0
000279                  STA         CFS.PTR+1
000280                  BCC         NFRPG.EXIT
000281  *
000282  * CFS.PREV:=CFS.NEXT
000283  * CFS.NEXT:=ST.FLINK(CFS.NEXT)
000284  *
000285                  STX         CFS.PREV
000286                  LDA         ST.FLINK,X
000287                  TAX
000288                  STX         CFS.NEXT
000289  *
000290  * IF CFS.NEXT=0 OR ST.LIM(CFS.NEXT)<CFS.PTR
000291  *    THEN DONE
000292  *    ELSE GOTO L0
000293  *
000294                  BEQ         NFRPG.EXIT
000295                  LDA         ST.LIML,X
000296                  CMP         CFS.PTR
000297                  LDA         ST.LIMH,X
000298                  SBC         CFS.PTR+1
000299                  BCS         NXTFRPG
000300  *
000301  NFRPG.EXIT      RTS                             ; NORMAL EXIT
000302                  PAGE
000303                  REP         60
000304  *
000305  * CHANGE.SEG(IN.SEGNUM,CHG.MODE; INOUT.PAGECT) SYSTEM CALL
000306  *
000307                  REP         60
000308  *
000309  CHG.SEG         EQU         *
000310  *
000311  * MOVE CALLER'S PAGE COUNT TO INTERNAL BUFFER
000312  *
000313                  LDY         #0
000314                  LDA         (CHG.PGCT),Y
000315                  STA         CHG.PGCTX
000316                  INY
000317                  LDA         (CHG.PGCT),Y
000318                  STA         CHG.PGCTX+1
000319  *
```



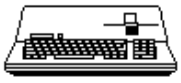
```
000320  * IF SEG# OUT OF RANGE OR ST.FLINK(SEG#)=FREE THEN ERR
000321  *
000322                  LDX         CHG.NUM
000323                  BEQ         CHGS.ERR
000324                  CPX         #ST.CNT
000325                  BCS         CHGS.ERR
000326                  LDA         ST.FLINK,X
000327                  BPL         CHGS005
000328  *
000329  CHGS.ERR        LDA         #BADSEGNUM
000330                  JSR         SYSERR              ; ERR EXIT
000331                  REP         35
000332  * CASE OF CHANGE MODE
000333                  REP         35
000334  CHGS005         LDY         CHG.MODE
000335                  CPY         #1
000336                  BCC         CHGS010
000337                  BEQ         CHGS020
000338                  CPY         #3
000339                  BCC         CHGS030
000340                  BEQ         CHGS040
000341  *
000342                  LDA         #BADCHGMODE
000343                  JSR         SYSERR              ; ERR EXIT
000344                  PAGE
000345                  REP         35
000346  * CHANGE MODE = 0(BASE UP)
000347                  REP         35
000348  * CHG.NEW:=ST.BASE(SEG#)+PGCT
000349  *
000350  CHGS010         CLC
000351                  LDA         ST.BASEL,X
000352                  ADC         CHG.PGCTX
000353                  STA         CHG.NEW
000354                  LDA         ST.BASEH,X
000355                  ADC         CHG.PGCTX+1
000356                  STA         CHG.NEW+1
000357  *
000358                  BCS         CHGS014             ; OVERFLOW, PEG IT
000359  *
000360  * IF CHG.NEW <= ST.LIM(SEG#) THEN EXIT
000361  *
000362                  LDA         ST.LIML,X
000363                  CMP         CHG.NEW
000364                  LDA         ST.LIMH,X
000365                  SBC         CHG.NEW+1
000366                  BCS         CHGS016
000367  *
000368  * OTHERWISE, CHG.NEW:=ST.LIM(SEG#)
000369  *
000370  CHGS014         LDA         ST.LIML,X
000371                  STA         CHG.NEW
000372                  LDA         ST.LIMH,X
000373                  STA         CHG.NEW+1
000374  *
000375  CHGS016         JMP         CHGS.EXIT
000376                  REP         35
000377  * CHANGE MODE = 1(BASE DOWN)
000378                  REP         35
000379  * CHG.NEW:=ST.BASE(SEG#)-PGCT
000380  *
000381  CHGS020         SEC
000382                  LDA         ST.BASEL,X
000383                  SBC         CHG.PGCTX
000384                  STA         CHG.NEW
000385                  LDA         ST.BASEH,X
000386                  SBC         CHG.PGCTX+1
000387                  STA         CHG.NEW+1
000388                  BCS         CHGS050
000389                  BCC         CHGS052             ; OVERFLOW, PEG IT
000390                  REP         35
000391  * CHANGE MODE = 2(LIMIT UP)
000392                  REP         35
000393  * CHG.NEW:=ST.LIM(SEG#)+PGCT
000394  *
000395  CHGS030         CLC
000396                  LDA         ST.LIML,X
000397                  ADC         CHG.PGCTX
000398                  STA         CHG.NEW
000399                  LDA         ST.LIMH,X
000400                  ADC         CHG.PGCTX+1
```



```
000401                  STA         CHG.NEW+1
000402                  BCC         CHGS050
000403                  BCS         CHGS052             ; OVERFLOW, PEG IT
000404                  REP         35
000405  * CHANGE MODE = 3(LIMIT DOWN)
000406                  REP         35
000407  * CHG.NEW:=ST.LIM(SEG#)-PGCT
000408  *
000409  CHGS040         SEC
000410                  LDA         ST.LIML,X
000411                  SBC         CHG.PGCTX
000412                  STA         CHG.NEW
000413                  LDA         ST.LIMH,X
000414                  SBC         CHG.PGCTX+1
000415                  STA         CHG.NEW+1
000416                  BCC         CHGS044             ; OVERFLOW, PEG IT
000417  *
000418  * IF CHG.NEW >= ST.BASE(SEG#) THEN EXIT
000419  *
000420                  LDA         CHG.NEW
000421                  CMP         ST.BASEL,X
000422                  LDA         CHG.NEW+1
000423                  SBC         ST.BASEH,X
000424                  BCS         CHGS046
000425  *
000426  * OTHERWISE CHG.NEW:=ST.BASE(SEG#)
000427  *
000428  CHGS044         LDA         ST.BASEL,X
000429                  STA         CHG.NEW
000430                  LDA         ST.BASEH,X
000431                  STA         CHG.NEW+1
000432  *
000433  CHGS046         JMP         CHGS.EXIT
000434  *
000435  * DETERMINE NEW BANK/PAGE'S REGION,
000436  * IF NEW BANK/PAGE IS INVALID THEN
000437  * SET TO BASE OR LIMIT (CASE CHANGE MODE)
000438  *
000439  CHGS050         LDX         CHG.NEW
000440                  LDY         CHG.NEW+1
000441                  JSR         REGION
000442                  BCS         CHGS052
000443                  BNE         CHGS052
000444                  BEQ         CHGS100
000445  CHGS052         LDA         CHG.MODE
000446                  CMP         #1
000447                  BNE         CHGS054
000448                  LDX         #>VRT.BASE
000449                  LDY         #<VRT.BASE
000450                  JMP         CHGS056
000451  CHGS054         LDX         >VRT.LIM
000452                  LDY         >VRT.LIM+1
000453  CHGS056         STX         CHG.NEW
000454                  STY         CHG.NEW+1
000455                  PAGE
000456  *
000457  * COMPUTE BANK/PAGE OF ADJACENT SEGMENT, IF ANY
000458  *   CASE CHANGE MODE
000459  *
000460  CHGS100         LDX         CHG.NUM
000461                  LDA         CHG.MODE
000462                  CMP         #1
000463                  BNE         CHGS200
000464  *   "1" IF ST.FLINK(SEG#)=0 THEN EXIT
000465                  LDA         ST.FLINK,X
000466                  BEQ         CHGS.EXIT
000467  *       X,Y:=ST.LIM(ST.FLINK(SEG#))+1
000468                  TAY
000469                  LDA         ST.LIML,Y
000470                  TAX
000471                  LDA         ST.LIMH,Y
000472                  TAY
000473                  INX
000474                  BNE         CHGS110
000475                  INY
000476  *       IF CHG.NEW < X,Y THEN CHG.NEW:=X,Y
000477  CHGS110         CPY         CHG.NEW+1
000478                  BCC         CHGS.EXIT
000479                  BEQ         CHGS120
000480                  BCS         CHGS300
000481  CHGS120         CPX         CHG.NEW
```

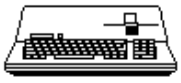


```
000482                  BCC         CHGS.EXIT
000483                  BCS         CHGS300
000484  *   "2" IF ST.BLINK(SEG#)=0 THEN EXIT
000485  CHGS200         LDA         ST.BLINK,X
000486                  BEQ         CHGS.EXIT
000487  *       X,Y:= ST.BASE(ST.BLINK(SEG#))-1
000488                  TAY
000489                  LDA         ST.BASEL,Y
000490                  TAX
000491                  LDA         ST.BASEH,Y
000492                  TAY
000493                  TXA
000494                  BNE         CHGS210
000495                  DEY
000496  CHGS210         DEX
000497  *       IF CHG.NEW > X,Y THEN CHG.NEW:=X,Y
000498                  CPY         CHG.NEW+1
000499                  BCC         CHGS300
000500                  BEQ         CHGS220
000501                  BCS         CHGS.EXIT
000502  CHGS220         CPX         CHG.NEW
000503                  BCS         CHGS.EXIT
000504  *
000505  CHGS300         STX         CHG.NEW
000506                  STY         CHG.NEW+1
000507                  PAGE
000508                  REP         35
000509  *
000510  * COMPUTE DELTA PAGE COUNT AND RETURN IT TO CALLER
000511  * (CASE OF CHG.MODE)
000512  *
000513                  REP         35
000514  CHGS.EXIT       LDX         CHG.NUM
000515                  LDY         #0
000516                  LDA         CHG.MODE
000517                  CMP         #1
000518                  BCC         CHGS500
000519                  BEQ         CHGS510
000520                  CMP         #3
000521                  BCC         CHGS520
000522                  BEQ         CHGS530
000523  *
000524  * "0" -- PAGECOUNT:=NEW-BASE
000525  *
000526  CHGS500         SEC
000527                  LDA         CHG.NEW
000528                  SBC         ST.BASEL,X
000529                  STA         (CHG.PGCT),Y
000530                  LDA         CHG.NEW+1
000531                  SBC         ST.BASEH,X
000532                  JMP         CHGS600
000533  *
000534  * "1" -- PAGECOUNT:=BASE-NEW
000535  *
000536  CHGS510         SEC
000537                  LDA         ST.BASEL,X
000538                  SBC         CHG.NEW
000539                  STA         (CHG.PGCT),Y
000540                  LDA         ST.BASEH,X
000541                  SBC         CHG.NEW+1
000542                  JMP         CHGS600
000543  *
000544  * "2" -- PAGECOUNT:=NEW-LIM
000545  *
000546  CHGS520         SEC
000547                  LDA         CHG.NEW
000548                  SBC         ST.LIML,X
000549                  STA         (CHG.PGCT),Y
000550                  LDA         CHG.NEW+1
000551                  SBC         ST.LIMH,X
000552                  JMP         CHGS600
000553  *
000554  * "3" -- PAGECOUNT:=LIM-NEW
000555  *
000556  CHGS530         SEC
000557                  LDA         ST.LIML,X
000558                  SBC         CHG.NEW
000559                  STA         (CHG.PGCT),Y
000560                  LDA         ST.LIMH,X
000561                  SBC         CHG.NEW+1
000562  *
```

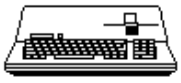


```
000563  CHGS600         INY
000564                  STA         (CHG.PGCT),Y
000565  *
000566  * IF NEW PAGE COUNT < REQUESTED PAGECOUNT THEN ERR
000567  *
000568                  TAX
000569                  DEY
000570                  LDA         (CHG.PGCT),Y
000571                  CMP         CHG.PGCTX
000572                  TXA
000573                  SBC         CHG.PGCTX+1
000574                  BCS         CHGS610
000575                  LDA         #SEGRQDN
000576                  JSR         SYSERR              ; ERR EXIT
000577  *
000578  * OTHERWISE, ENTER CHG.NEW IN SEGMENT TABLE AND EXIT
000579  *
000580  CHGS610         LDX         CHG.NUM
000581                  LDA         CHG.MODE
000582                  CMP         #2
000583                  LDA         CHG.NEW
000584                  LDY         CHG.NEW+1
000585                  BCS         CHGS620
000586  *
000587                  STA         ST.BASEL,X
000588                  TYA
000589                  STA         ST.BASEH,X
000590                  CLC
000591                  RTS                             ; NORMAL EXIT
000592  *
000593  *
000594  CHGS620         STA         ST.LIML,X
000595                  TYA
000596                  STA         ST.LIMH,X
000597                  CLC
000598                  RTS                             ; NORMAL EXIT
000599
000600                  CHN         MEMMGR.C.SRC
000601
000602  ********************************************************************************
000603  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.B.SRC
000604  ********************************************************************************
000605
000606
```

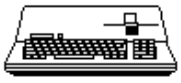
End of File -- Lines: 606 Characters: 12658



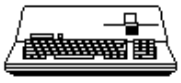
```
================================================================================================
FILE: "SOS.MEMMGR.C.SRC.TEXT"
================================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.C.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP         60
000008  *
000009  * GET.SEG.INFO(IN.SEGNUM; OUT.BASE.BKPG,LIMIT.BKPG,PGCT,SEGID)
000010  *
000011                  REP         60
000012  *
000013  GET.SEG.INFO    EQU         *
000014  *
000015  * IF SEG# OUT OF BOUNDS OR ST.FLINK(SEG#)=ST.FREE THEN ERR
000016  *
000017                  LDX         GSI.NUM
000018                  BEQ         GSI.ERR             ; ERR - INVALID SEGNUM
000019                  CPX         #ST.CNT
000020                  BCS         GSI.ERR             ; ERR - INVALID SEGNUM
000021                  LDA         ST.FLINK,X
000022                  BMI         GSI.ERR             ; ERR - INVALID SEGNUM
000023  *
000024  * RETURN BASE.BKPG TO CALLER
000025  *
000026                  LDY         ST.BASEH,X
000027                  LDA         ST.BASEL,X
000028                  TAX
000029                  JSR         CNVRT.XBP
000030                  TYA
000031                  LDY         #1
000032                  STA         (GSI.BASE),Y
000033                  DEY
000034                  TXA
000035                  STA         (GSI.BASE),Y
000036  *
000037  * RETURN LIMIT.BKPG TO CALLER
000038  *
000039                  LDX         GSI.NUM
000040                  LDY         ST.LIMH,X
000041                  LDA         ST.LIML,X
000042                  TAX
000043                  JSR         CNVRT.XBP
000044                  TYA
000045                  LDY         #1
000046                  STA         (GSI.LIM),Y
000047                  DEY
000048                  TXA
000049                  STA         (GSI.LIM),Y
000050  *
000051  * RETURN SEGID TO CALLER
000052  *
000053                  LDX         GSI.NUM
000054                  LDA         ST.ID,X
000055                  STA         (GSI.ID),Y
000056  *
000057  * COMPUTE PAGE COUNT
000058  *
000059                  SEC
000060                  LDA         ST.LIML,X
000061                  SBC         ST.BASEL,X
000062                  TAY
000063                  LDA         ST.LIMH,X
000064                  SBC         ST.BASEH,X
000065                  TAX
000066                  INY
000067                  BNE         GSI010
000068                  INX
000069  *
000070  * RETURN PAGE COUNT TO CALLER
000071  *
000072  GSI010          TYA
000073                  LDY         #0
000074                  STA         (GSI.PGCT),Y
000075                  INY
000076                  TXA
```



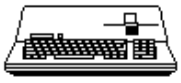
```
000077                  STA         (GSI.PGCT),Y
000078  *
000079                  CLC
000080                  RTS                             ; NORMAL EXIT
000081  *
000082  GSI.ERR         LDA         #BADSEGNUM
000083                  JSR         SYSERR              ; ERR EXIT
000084                  PAGE
000085                  REP         60
000086  *
000087  * GET.SEG.NUM(IN.BANKPAGE; OUT.SEGNUM) SYSTEM CALL
000088  *
000089  *
000090                  REP         60
000091  *
000092  GET.SEG.NUM     EQU         *
000093  *
000094  * CONVERT BANKPAGE TO INTERNAL FORMAT
000095  *
000096                  LDX         GSN.BKPG
000097                  LDY         GSN.BKPG+1
000098                  JSR         CNVRT.IBP
000099                  BCS         GSN.ERR             ; ERR - INVALID BANK PAGE
000100                  STX         GSN.BKPG
000101                  STY         GSN.BKPG+1
000102  *
000103  * QUIT IF NO ENTRIES IN SEG TABLE
000104  *
000105                  LDA         ST.ENTRY
000106                  BEQ         GSN.ERR1            ; ERR - SEG NOT FOUND
000107  *
000108  * L1: IF BANKPAGE>ST.LIM(SEG#) THEN ERR
000109  *
000110  GSN010          TAX
000111                  LDA         ST.LIML,X
000112                  CMP         GSN.BKPG
000113                  LDA         ST.LIMH,X
000114                  SBC         GSN.BKPG+1
000115                  BCC         GSN.ERR1            ; ERR - SEG NOT FOUND
000116  *
000117  * IF BANKPAGE>=ST.BASE(SEG#) THEN FOUND!
000118  *
000119                  LDA         GSN.BKPG
000120                  CMP         ST.BASEL,X
000121                  LDA         GSN.BKPG+1
000122                  SBC         ST.BASEH,X
000123                  BCS         GSN020
000124  *
000125  * SEG#:=ST.FLINK(SEG#); GOTO L1
000126  *
000127                  LDA         ST.FLINK,X
000128                  BEQ         GSN.ERR1            ; ERR - SEG NOT FOUND
000129                  JMP         GSN010
000130  *
000131  * RETURN SEG# TO CALLER
000132  *
000133  GSN020          LDY         #0
000134                  TXA
000135                  STA         (GSN.NUM),Y
000136                  CLC
000137                  RTS                             ; NORMAL EXIT
000138  *
000139  GSN.ERR         RTS                             ; ERROR EXIT
000140  *
000141  GSN.ERR1        LDA         #SEGNOTFND
000142                  JSR         SYSERR              ; ERROR EXIT
000143                  PAGE
000144                  REP         60
000145  *
000146  * RELEASE.SEG(IN.SEGNUM) SYSTEM CALL
000147  *
000148                  REP         60
000149  *
000150  RELEASE.SEG     EQU         *
000151  *
000152  * IF ST.FLINK(SEG#)=ST.FREE THEN ERR
000153  *
000154                  LDX         RLS.NUM
000155                  BEQ         RLS.ALL             ; RELEASE.SEG(SEG#=0)
000156                  CPX         #ST.CNT
000157                  BCS         RLS.ERR             ; ERR - SEG# TOO LARGE
```



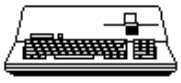
```
000158                  LDA         ST.FLINK,X
000159                  BMI         RLS.ERR             ; ERR - INVALID SEGNUM
000160                  BPL         REL.SEG             ; RELEASE.SEG(SEG#>0)
000161                  REP         35
000162  *
000163  * RELEASE ALL
000164  *
000165                  REP         35
000166  RLS.ALL         LDX         ST.ENTRY
000167                  BEQ         RLS0.EXIT
000168                  STX         RLS.NUM
000169  *
000170  RLS0.LOOP       LDA         ST.ID,X
000171                  CMP         #$10                ; CARRY SET/CLEARED HERE
000172  *
000173                  LDA         ST.FLINK,X
000174                  PHA
000175                  BCC         RLS006              ; IF ID=SYS SEG THEN SKIP
000176                  JSR         REL.SEG             ; RELEASE ONE SEGMENT
000177  RLS006          PLA
000178                  BEQ         RLS0.EXIT
000179                  STA         RLS.NUM
000180                  TAX
000181                  BNE         RLS0.LOOP           ; ALWAYS TAKEN
000182  *
000183  RLS0.EXIT       CLC
000184                  RTS                             ; NORMAL EXIT ; ALL NON SYSTEM SEGMENTS RELEASED.
000185                  REP         35
000186  *
000187  * REL SEG
000188  *
000189                  REP         35
000190  * Y:=ST.FLINK(SEG#)
000191  * X:=ST.BLINK(SEG#)
000192  *
000193  REL.SEG         TAY
000194                  LDA         ST.BLINK,X
000195                  TAX
000196  *
000197  * IF X<>0 THEN ST.FLINK(X):=Y
000198  *         ELSE ST.ENTRY:=Y
000199  *
000200                  BEQ         RLS010
000201                  TYA
000202                  STA         ST.FLINK,X
000203                  JMP         RLS020
000204  RLS010          STY         ST.ENTRY
000205  *
000206  * IF Y<>0 THEN ST.BLINK(Y):=X
000207  *
000208                  TYA
000209  RLS020          BEQ         RLS030
000210                  TXA
000211                  STA         ST.BLINK,Y
000212  *
000213  * ST.FLINK(SEG#):=ST.FREE
000214  * ST.FREE:=SEG# AND #$80
000215  *
000216  RLS030          LDA         ST.FREE
000217                  LDX         RLS.NUM
000218                  STA         ST.FLINK,X
000219                  TXA
000220                  ORA         #$80
000221                  STA         ST.FREE
000222  *
000223                  CLC
000224                  RTS                             ; NORMAL EXIT
000225  *
000226  RLS.ERR         LDA         #BADSEGNUM
000227                  JSR         SYSERR              ; ERR EXIT
000228                  PAGE
000229                  REP         60
000230  *
000231  * CONVERT INTERNAL BANK PAGE
000232  *
000233  * INPUT:  EXTERNAL BANK (X)
000234  *            "     PAGE (Y)
000235  * OUTPUT: INTERNAL BKPG LOW  (X)
000236  *            "     BKPG HIGH (Y)
000237  *         REGION (A) 0=>VIRT BANK
000238  *                    1=>PHY BANK (0-$2000)
```



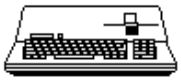
```
000239  *                    2=>   "     ($A000-$FFFF)
000240  * ERROR:  CARRY SET ("INVALID BANK PAGE")
000241  *
000242                  REP         60
000243  *
000244  CNVRT.IBP       EQU         *
000245  *
000246  * CONVERT FROM EXTERNAL TO INTERNAL FORMAT
000247  *
000248  * CASE OF BANK:  ADD PAGE BIAS
000249  *
000250                  TYA
000251                  CPX         #$F
000252                  BEQ         CNVI010
000253                  BCS         CNVI020
000254  *
000255                  CMP         #$20                ; BANK < "F"
000256                  BCC         CNVI.ERR1
000257                  CMP         #$A0
000258                  BCS         CNVI.ERR1
000259                  SEC
000260                  SBC         #$20
000261                  JMP         CNVI030
000262  *
000263  CNVI010         CMP         #$20                ; BANK = "F"
000264                  BCS         CNVI.ERR1
000265                  CLC
000266                  ADC         #$80
000267                  JMP         CNVI030
000268  *
000269  CNVI020         CPX         #$10                ; BANK = "10"
000270                  BNE         CNVI.ERR1
000271                  CMP         #$A0
000272                  BCC         CNVI.ERR1
000273                  SEC
000274                  SBC         #$80
000275  *
000276  CNVI030         TAY                             ; SHIFT BANK RIGHT ONE BIT
000277                  TXA                             ; INTO HIGH BIT OF PAGE BYTE.
000278                  LSR         A
000279                  TAX
000280                  TYA
000281                  BCC         CNVI040
000282                  ORA         #$80
000283  *
000284  * EXCHANGE X & Y
000285  *
000286  CNVI040         PHA
000287                  TXA
000288                  TAY
000289                  PLA
000290                  TAX
000291  *
000292  * COMPUTE REGION (VIRT=0,PHY1=1,PHY2=2)
000293  *
000294                  JSR         REGION              ; REGION RETURNED IN A REG.
000295                  BCS         CNVI.ERR1           ; ERR - INVALID BANK PAGE
000296  *
000297                  RTS                             ; NORMAL EXIT
000298  *
000299  CNVI.ERR1       LDA         #BADBKPG
000300                  JSR         SYSERR
000301                  PAGE
000302                  REP         60
000303  *
000304  * CONVERT EXTERNAL BANK PAGE
000305  *
000306  * INPUT:  INTERNAL BKPG LOW  (X)
000307  *             "         HIGH (Y)
000308  * OUTPUT: EXTERNAL BANK (X)
000309  *             "    PAGE (Y)
000310  * ERROR:  NO ERROR CHECKING DONE.  ASSUMES THAT INTERNAL #S
000311  * ARE VALID.
000312  *
000313                  REP         60
000314  *
000315  CNVRT.XBP       EQU         *
000316  *
000317  * CONVERT FROM INTERNAL TO EXTERNAL FORMAT
000318  *
000319                  TXA
```



```
000320                  ASL         A
000321                  TXA
000322                  AND         #$7F
000323                  TAX
000324                  TYA
000325                  ROL         A
000326                  TAY
000327  *
000328  * CASE OF BANK: ADD PAGE BIAS
000329  *
000330                  TXA
000331                  CPY         #$F
000332                  BEQ         CNVX020             ; BANK = "F"
000333                  BCS         CNVX010
000334  *
000335                  CLC                             ; BANK < "F"
000336                  ADC         #$20
000337                  JMP         CNVX020
000338  *
000339  CNVX010         CLC                             ; BANK = "10"
000340                  ADC         #$80
000341  *
000342  * EXCHANGE X & Y
000343  *
000344  CNVX020         PHA
000345                  TYA
000346                  TAX
000347                  PLA
000348                  TAY
000349                  RTS                             ; NORMAL EXIT
000350                  PAGE
000351                  REP         60
000352  *
000353  * REGION
000354  *
000355  * INPUT:  INTERNAL BKPG LOW  (X)
000356  *             "         HIGH (Y)
000357  * OUTPUT: REGION (A)
000358  *         INTERNAL BKPG LOW  (X) UNCHANGED
000359  *             "         HIGH (Y)    "
000360  * ERROR:  CARRY SET ("INVALID BANK/PAGE")
000361  *
000362                  REP         60
000363  *
000364  REGION          EQU         *
000365                  STX         RGN.BKPG
000366                  STY         RGN.BKPG+1
000367  *
000368  * IF BANKPAGE>PHY2LIM THEN ERR
000369  *
000370                  LDA         #>PHY2LIM
000371                  CMP         RGN.BKPG
000372                  LDA         #<PHY2LIM
000373                  SBC         RGN.BKPG+1
000374                  BCC         RGN.ERR             ; ERR - INVALID BANK PAGE
000375  *
000376  * IF BANKPAGE>=PHY2BASE THEN REGION:=2
000377  *
000378                  LDA         RGN.BKPG
000379                  CMP         #>PHY2BASE
000380                  LDA         RGN.BKPG+1
000381                  SBC         #<PHY2BASE
000382                  BCC         RGN010
000383                  LDA         #2
000384                  BNE         RGN040
000385  *
000386  * IF BANKPAGE>PHY1LIMIT THEN ERR
000387  *
000388  RGN010          LDA         #>PHY1LIM
000389                  CMP         RGN.BKPG
000390                  LDA         #<PHY1LIM
000391                  SBC         RGN.BKPG+1
000392                  BCC         RGN.ERR             ; ERR - INVALID BANK PAGE
000393  *
000394  * IF BANKPAGE>=PHY1BASE THEN REGION:=1
000395  *
000396                  LDA         RGN.BKPG
000397                  CMP         #>PHY1BASE
000398                  LDA         RGN.BKPG+1
000399                  SBC         #<PHY1BASE
000400                  BCC         RGN020
```

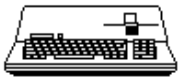


```
000401                  LDA         #1
000402                  BNE         RGN040
000403  *
000404  * IF BANKPAGE>VIRTUAL LIMIT THEN ERR
000405  *
000406  RGN020          LDA         >VRT.LIM
000407                  CMP         RGN.BKPG
000408                  LDA         >VRT.LIM+1
000409                  SBC         RGN.BKPG+1
000410                  BCC         RGN.ERR
000411                  LDA         #0
000412  *
000413  RGN040          CLC                             ; "N" FLAG ALWAYS REFLECTS REGION VAL IN A REG!
000414                  RTS                             ; NORMAL EXIT
000415  *
000416  RGN.ERR         SEC                             ; INVALID BANK PAGE
000417                  RTS
000418                  PAGE
000419                  REP         60
000420  *
000421  * GET FREE
000422  *
000423  * INPUT:  PREVIOUS SEG # (A)
000424  * OUTPUT: NEW SEG #      (A)
000425  * ERROR:  CARRY SET ("SEG TBL FULL")
000426  *
000427                  REP         60
000428  *
000429  GET.FREE        EQU         *
000430  *
000431  * SAVE PREV SEG # IN X
000432  * NOTE: PREV SEG # CARRIED IN X
000433  *       NEW SEG # CARRIED IN Y
000434  *
000435                  TAX
000436  *
000437  * IF NO FREE ENTRIES THEN ERR
000438  *
000439                  LDA         ST.FREE
000440                  CMP         #$80
000441                  BEQ         GTFR.ERR
000442  *
000443  * TURN OFF FREE FLAG (BIT7) AND DELINK FROM FREE LIST
000444  *
000445                  AND         #$7F
000446                  TAY
000447                  LDA         ST.FLINK,Y
000448                  STA         ST.FREE
000449  *
000450  * IF PREV SEG # IS NULL THEN LINK NEW ENTRY TO START
000451  * OF SEGMENT LIST
000452  *
000453                  CPX         #0
000454                  BNE         GTFR010
000455                  LDA         ST.ENTRY
000456                  STA         ST.FLINK,Y
000457                  LDA         #0
000458                  STA         ST.BLINK,Y
000459                  STY         ST.ENTRY
000460                  JMP         GTFR020
000461  *
000462  * OTHERWISE LINK NEW ENTRY TO PREV SEG #
000463  *
000464  GTFR010         LDA         ST.FLINK,X
000465                  STA         ST.FLINK,Y
000466                  TXA
000467                  STA         ST.BLINK,Y
000468                  TYA
000469                  STA         ST.FLINK,X
000470  *
000471  * IF ST.FLINK(NEW)<>NULL THEN
000472  *    ST.BLINK(ST.FLINK(NEW)):=NEWSEG #
000473  GTFR020         LDA         ST.FLINK,Y
000474                  BEQ         GTFR030
000475                  LDA         ST.FLINK,Y
000476                  TAX
000477                  TYA
000478                  STA         ST.BLINK,X
000479  *
000480  * RETURN WITH NEW SEG #
000481  *
```

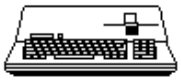


```
000482  GTFR030         TYA
000483                  CLC
000484                  RTS                             ; NORMAL EXIT
000485  *
000486  GTFR.ERR        LDA         #SEGTBLFULL
000487                  JSR         SYSERR
000488  *
000489                  LST         ON
000490  ZZEND           EQU         *
000491  ZZLEN           EQU         ZZEND-ZZORG
000492                  IFNE        ZZLEN-LENMEMMG
000493                  FAIL        2,"SOSORG           FILE IS INCORRECT FOR MEMMGR"
000494                  FIN
000495
000496  ********************************************************************************
000497  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.C.SRC
000498  ********************************************************************************
000499
000500
```

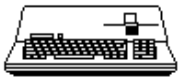
End of File -- Lines: 500 Characters: 10157



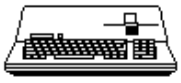
```
================================================================================================
FILE: "SOS.OPRMSG.SRC.TEXT"
================================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: OPRMSG.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1  OPERATOR MESSAGE/REPLY"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGOMSG
000010  ZZORG           EQU        *
000011                  MSB        OFF
000012                  REP        60
000013  *
000014  *          COPYRIGHT (C) APPLE COMPUTER INC. 1981
000015  *                    ALL RIGHTS RESERVED
000016  *
000017                  REP        60
000018  *
000019  *  THIS MODULE CONTAINS THE BLOCK FILE MANAGERS'S OPERATOR
000020  *  INTERFACE.  IT DISPLAYS A MESSAGE IN A FOUR LINE BY
000021  *  FOURTY COLUMN WINDOW, THEN WAITS FOR THE USER TO TOGGLE
000022  *  THE ALPHA-LOCK KEY BEFORE RETURNING.
000023  *
000024  *  THE VERTICAL BLANKING FLAGS AND COMPOSITE BLANKING
000025  *  TIMER ARE USED TO MAINTAIN THE DISPLAY.  MEMORY PAGE
000026  *  $02 IS USED FOR TEMPORARY STORAGE.  ON EXIT, ALL
000027  *  RESOURCES ARE RESTORED TO THEIR PREVIOUS STATES.
000028  *
000029  *  ENTRY POINT:  OPMSGRPLY
000030  *
000031  *  PARAMETERS:  X -- MESSAGE ADDRESS (LOW BYTE)
000032  *               Y -- MESSAGE ADDRESS (HIGH BYTE)
000033  *     (THE MESSAGE MUST RESIDE IN THE CURRENT BANK)
000034  *
000035  *  RESULT:  A -- RESPONSE KEYSTROKE
000036  *           X, Y -- UNDEFINED
000037  *
000038                  REP        60
000039  *
000040  *
000041                  ENTRY      OPMSGRPLY
000042  *
000043                  EXTRN      SCRNMODE
000044                  PAGE
000045  *
000046  *  HARDWARE EQUATES
000047  *
000048  Z.REG           EQU        $FFD0
000049  E.REG           EQU        $FFDF
000050  *
000051  KBPORT          EQU        $C008
000052  *
000053  BELL            EQU        $C040
000054  *
000055  VM0             EQU        $C050
000056  VM1             EQU        $C052
000057  VM2             EQU        $C054
000058  VM3             EQU        $C056
000059  *
000060  E.T2            EQU        $FFE8
000061  E.ACR           EQU        $FFEB
000062  E.PCR           EQU        $FFEC
000063  E.IFR           EQU        $FFED
000064  E.IER           EQU        $FFEE
000065  *
000066  *  ZERO PAGE DECLARATIONS
000067  *
000068                  DSECT
000069  ZPBASE          EQU        $200
000070                  ORG        $0000              ;ZERO PAGE DECLARATIONS
000071  MSGPTR          DS         2                  ;MESSAGE POINTER
000072  MSGIDX          DS         1
000073  *
000074  SCRNIDX         DS         1
000075  SCRNPTR         DS         2
000076  DATAPTR         DS         2
```



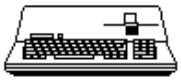
```
000077  DATABUF         DS          160
000078  *
000079  SV.ZREG         DS          1
000080  SV.EREG         DS          1
000081  SV.SMODE        DS          1
000082  SV.EACR         DS          1
000083  SV.EPCR         DS          1
000084  SV.EIER         DS          1
000085  *
000086  FLAG            DS          1
000087                  DEND
000088                  PAGE
000089  OPMSGRPLY       EQU         *
000090  *
000091  *
000092  *  SAVE CURRENT VALUES AND SET UP ZERO PAGE,
000093  *  ENVIRONMENT, SCREEN MODE, AND E.6522 REGISTERS.
000094  *
000095                  PHP
000096                  SEI
000097                  LDA         Z.REG
000098                  STA         ZPBASE+SV.ZREG      ;SAVE ZERO PAGE
000099                  LDA         #<ZPBASE
000100                  STA         Z.REG
000101                  STX         MSGPTR              ;SAVE MESSAGE ADDRESS
000102                  STY         MSGPTR+1
000103                  LDA         E.REG
000104                  STA         SV.EREG             ;SAVE ENVIRONMENT
000105                  AND         #$5F
000106                  ORA         #$40
000107                  STA         E.REG               ;SCREEN OFF, I/O SPACE ON
000108                  LDA         SCRNMODE
000109                  STA         SV.SMODE            ;SAVE SCREEN MODE
000110                  LDA         #$00
000111                  STA         SCRNMODE
000112                  BIT         VM0                 ;SET 40 COLUMN
000113                  BIT         VM1                 ;  BLACK & WHITE TEXT
000114                  BIT         VM2
000115                  BIT         VM3
000116                  LDX         E.ACR
000117                  TXA
000118                  AND         #$20
000119                  STA         SV.EACR             ;SAVE AUXILIARY CONTROL REG
000120                  TXA
000121                  ORA         #$20
000122                  STA         E.ACR               ;SET UP BL TIMER
000123                  LDX         E.PCR
000124                  TXA
000125                  AND         #$F0
000126                  STA         SV.EPCR             ;SAVE PERIPHERAL CONTROL REG
000127                  TXA
000128                  AND         #$0F
000129                  ORA         #$60
000130                  STA         E.PCR               ;SET UP VBL FLAGS
000131                  LDA         E.IER
000132                  AND         #$38
000133                  STA         E.IER               ;MASK VBL & BL INTERRUPTS
000134                  STA         SV.EIER             ;SAVE INTERRUPT MASKS
000135                  PLP
000136  *
000137  *
000138  *  SAVE SCREEN DATA AND CLEAR MESSAGE WINDOW
000139  *
000140                  LDX         #3
000141  OPR010          JSR         SETPTRS
000142                  LDY         #39
000143  OPR020          LDA         (SCRNPTR),Y         ;SAVE SCREEN DATA
000144                  STA         (DATAPTR),Y
000145                  LDA         #$A0
000146                  STA         (SCRNPTR),Y         ;BLANK SCREEN
000147                  DEY
000148                  BPL         OPR020
000149                  DEX
000150                  BPL         OPR010
000151  *
000152  *
000153  *  MOVE MESSAGE TO WINDOW
000154  *
000155                  BIT         BELL
000156                  LDX         #$00
000157                  STX         MSGIDX
```



```
000158  OPR100          JSR        SETPTRS
000159                  LDY        #$00
000160                  STY        SCRNIDX
000161  OPR110          LDY        MSGIDX
000162                  INC        MSGIDX
000163                  LDA        (MSGPTR),Y          ;SET UP MESSAGE
000164                  BEQ        OPR110
000165                  BMI        OPR200
000166                  CMP        #$0D
000167                  BEQ        OPR120
000168                  LDY        SCRNIDX
000169                  INC        SCRNIDX
000170                  ORA        #$80
000171                  STA        (SCRNPTR),Y
000172                  CPY        #39
000173                  BCC        OPR110
000174  OPR120          INX
000175                  CPX        #4
000176                  BCC        OPR100
000177  *
000178  *
000179  *  DISPLAY MESSAGE UNTIL ALPHA-LOCK KEY TOGGLES
000180  *
000181  OPR200          LDY        #2
000182                  LDA        KBPORT
000183                  AND        #$08
000184                  STA        FLAG
000185  OPR210          JSR        VIDEO
000186                  LDA        KBPORT
000187                  AND        #$08
000188                  CMP        FLAG
000189                  BEQ        OPR210
000190                  STA        FLAG
000191                  DEY
000192                  BNE        OPR210
000193  *
000194  *
000195  *  RESTORE PREVIOUS CONTENTS OF WINDOW
000196  *
000197                  LDX        #3
000198  OPR400          JSR        SETPTRS
000199                  LDY        #39
000200  OPR410          LDA        (DATAPTR),Y
000201                  STA        (SCRNPTR),Y
000202                  DEY
000203                  BPL        OPR410
000204                  DEX
000205                  BPL        OPR400
000206  *
000207  *
000208  *  RESTORE E.6522, SCREEN MODE, ENVIRONMENT, & ZERO PAGE
000209  *  THEN RETURN TO CALLER
000210  *
000211                  PHP
000212                  SEI
000213                  LDA        E.ACR
000214                  AND        #$DF
000215                  ORA        SV.EACR             ;RESTORE AUXILIARY CONTROL REG
000216                  STA        E.ACR
000217                  LDA        E.PCR
000218                  AND        #$0F
000219                  ORA        SV.EPCR             ;RESTORE PERIPHERAL CONTROL REG
000220                  STA        E.PCR
000221                  LDA        SV.EIER             ;RESTORE INTERRUPT ENABLE REG
000222                  ORA        #$80
000223                  STA        E.IER
000224                  LDA        SV.SMODE            ;RESTORE SCREEN MODE
000225                  STA        SCRNMODE
000226                  LSR        A
000227                  BCC        OPR500
000228                  BIT        VM0+1               ;RESTORE VIDEO MODE
000229  OPR500          LSR        A
000230                  BCC        OPR510
000231                  BIT        VM1+1
000232  OPR510          LSR        A
000233                  BCC        OPR520
000234                  BIT        VM2+1
000235  OPR520          BIT        SCRNMODE
000236                  BVC        OPR530
000237                  BIT        VM3+1
000238  OPR530          LDA        SV.EREG             ;RESTORE ENVIRONMENT
```

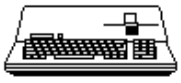


```
000239                  STA         E.REG
000240                  LDA         SV.ZREG             ;RESTORE ZERO PAGE
000241                  STA         Z.REG
000242                  PLP
000243                  RTS
000244                  PAGE
000245                  REP         60
000246  *
000247  *  SUBROUTINE VIDEO
000248  *
000249  *  THIS SUBROUTINE POLLS THE VERTICAL-BLANKING AND
000250  *  COMPOSITE-BLANKING-TIMER FLAGS AND TURNS THE SCREEN
000251  *  OFF AND ON SO THAT ONLY THE MESSAGE WINDOW WILL BE
000252  *  DISPLAYED.
000253  *
000254  *  THE E.6522 MUST BE INITIALIZED SO THAT E.CB2 FLAGS THE
000255  *  POSITIVE EDGE OF VBL AND E.T2 COUNTS BL PULSES.  THE
000256  *  INTERRUPTS MUST BE MASKED AND THE PROPER COUNT MUST
000257  *  ALREADY BE STORED IN THE LOW ORDER BYTE OF E.T2.
000258  *
000259  *  ENTRY:  VIDEO
000260  *
000261  *  PARAMETERS:  INTERRUPT SYSTEM DISABLED
000262  *
000263  *  EXIT:  A -- UNDEFINED
000264  *         X, Y -- PRESERVED
000265  *
000266                  REP         60
000267  *
000268  VIDEO           EQU         *
000269                  LDA         E.IFR
000270                  AND         #$28                ;GET VBL & BL FLAGS
000271                  BEQ         VID030
000272                  STA         E.IFR               ;CLEAR FLAGS
000273                  AND         #$20                ;WHICH FLAG?
000274                  BNE         VID010              ;  BL
000275  *
000276                  LDA         #$1F
000277                  STA         E.T2                ;SET UP BL TIMER
000278                  LDA         #$00
000279                  STA         E.T2+1
000280                  LDA         E.REG
000281                  ORA         #$20                ;SET UP FOR SCREEN ON
000282                  SEC
000283                  BCS         VID020
000284  *
000285  VID010          LDA         E.REG
000286                  AND         #$DF                ;SET UP FOR SCREEN OFF
000287                  CLC
000288  *
000289  VID020          STA         E.REG
000290                  LDA         #$00
000291                  ROR         A
000292                  STA         SCRNMODE
000293  VID030          RTS
000294                  PAGE
000295                  REP         60
000296  *
000297  *  SUBROUTINE SETPTRS
000298  *
000299  *  THIS SUBROUTINE SETS UP THE POINTERS TO THE MESSAGE
000300  *  WINDOW AND DATA SAVE AREA.
000301  *
000302  *  ENTRY:  SETPTRS
000303  *
000304  *  PARAMETERS:  X -- LINE NUMBER [0..3]
000305  *
000306  *  EXIT:  A -- UNDEFINED
000307  *         X, Y -- PRESERVED
000308  *
000309                  REP         60
000310  *
000311  SETPTRS         EQU         *
000312                  TXA
000313                  LSR         A
000314                  ORA         #$04
000315                  STA         SCRNPTR+1
000316                  LDA         #$00
000317                  ROR         A
000318                  STA         SCRNPTR
000319                  LDA         #<DATABUF
```

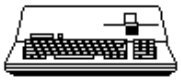


```
000320                  STA         DATAPTR+1
000321                  LDA         DBUFADR,X
000322                  STA         DATAPTR
000323                  RTS
000324  *
000325  DBUFADR         EQU         *
000326                  DFB         >0*40+DATABUF
000327                  DFB         >1*40+DATABUF
000328                  DFB         >2*40+DATABUF
000329                  DFB         >3*40+DATABUF
000330                  LST         ON
000331
000332  ZZEND           EQU         *
000333  ZZLEN           EQU         ZZEND-ZZORG
000334                  IFNE        ZZLEN-LENOMSG
000335                  FAIL        2,"SOSORG            FILE IS INCORRECT FOR OPRMSG"
000336                  FIN
000337
000338  ********************************************************************************
000339  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: OPRMSG.SRC
000340  ********************************************************************************
000341
000342
```

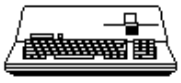
End of File -- Lines: 342 Characters: 8079



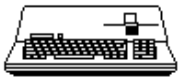
```
================================================================================================
FILE: "SOS.PATH.TEXT"
================================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: PATH
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007  *
000008  *
000009  *
000010  BFMGR           LDX         COMMAND             ; WHAT CALL?
000011  *
000012  *
000013  *
000014                  LDA         DISPTCH,X           ; TRANSLATE INTO COMMAND ADDRESS
000015                  ASL         A                   ; (BIT 7 INDICATES IT'S GOT A PATHNAME TO PREPROCESS)
000016                  STA         CMDTEMP
000017                  AND         #$3F                ; (BIT 6 IS REFNUM PREPROCESS, 5 IS FOR TIME, SO STRIP EM.)
000018                  TAX
000019                  LDA         CMDTABLE,X          ; MOVE ADDRESS FOR INDIRECT JUMP.
000020                  STA         CMDADR
000021                  LDA         CMDTABLE+1,X        ; (HIGH BYTE)
000022                  STA         CMDADR+1
000023                  LDA         #<VCB
000024                  STA         VCBPTR+1            ; INSURE DEFAULT HI ADDRESS TO VCB BEFORE CALLS
000025                  LDA         #BKBITVAL           ; INIT "BACKUP BIT FLAG"
000026                  STA         BKBITFLG            ; TO SAY "FILE MODIFIED"
000027                  LDY         #MAXTEMPS           ; ZERO OUT SISTER PAGE FOR TEMPS
000028                  LDA         #0
000029                  STA         SERR                ; MAKE GLOBAL ERROR SAY "NONE"
000030                  STA         DSWGLOB             ; "DISK SWITCH GLOBAL"
000031                  STA         DUPLFLAG            ; "DUPLICATE VOLUME ON LINE"
000032                  STA         CFLAG               ; SET "CREATE" TO NO
000033                  STA         BLOKSAVE
000034                  STA         BLOKSAVE+1          ; SET PARENT DIRECTORY TO NULL
000035  CLRSIS          STA         SISTEMPS,Y
000036                  DEY
000037                  BPL         CLRSIS              ; CARRY IS UNDISTURBED BY THIS LOOP
000038                  BCC         NOPATH
000039                  JSR         SETPATH             ; GO PROCESS PATHNAME BEFORE CALLING COMMAND
000040                  BCS         ERRORSYS            ; BRANCH IF BAD NAME.
000041  NOPATH          ASL         CMDTEMP             ; TEST FOR REFNUM PREPROCESSING
000042                  BCC         NOPREREF
000043                  JSR         FINDFCB             ; GO SET UP POINTERS TO FCB AND VCB OF THIS FILE.
000044                  BCS         ERRORSYS            ; BRANCH IF ANY ERRORS ARE ENCOUNTERED.
000045  NOPREREF        ASL         CMDTEMP             ; LASTLY CHECK FOR NECESSITY OF TIME STAMP.
000046                  BCC         TSWVRFY
000047                  LDX         #DATELO             ; PASS Z PAGE ADDRESS OF WHERE TO RETURN DATE/TIME
000048                  JSR         DATETIME            ; (NO ERROR POSIBLE)
000049  TSWVRFY         LDX         COMMAND             ; TEST FOR NECESSITY OF VOLUME VERIFICATION
000050                  LDA         #PREPATH+PREREF+PRETIME ; TO ENSURE VCB IS SET
000051                  AND         DISPTCH,X
000052                  BEQ         EXECUTE
000053                  LDY         #VCBSTAT
000054                  LDA         (VCBPTR),Y
000055                  AND         #DSWITCH            ; WAS THE VOLUME PREVIOUSLY SWITCHED?
000056                  BEQ         EXECUTE
000057                  DEY                             ; GET DEVICE NUMBER
000058                  LDA         (VCBPTR),Y
000059                  STA         DEVNUM
000060  DVERIFY         JSR         VERFYVOL            ; SEE IF PROPER VOLUME NOW ON LINE
000061                  BCC         CLRDSWT             ; BRANCH IF YES
000062                  JSR         USRREQ              ; OTHERWISE REQUEST IT BE PUT ON LINE
000063                  BCC         DVERIFY             ; USER SEZ S/HE DID: CHECK IT OUT
000064                  LDA         #VNFERR             ; VOLUME NOT FOUND IF USER REFUSES
000065                  BNE         ERRORSYS            ; REPORT ERROR (BRANCH ALWAYS)
000066  CLRDSWT         LDY         #VCBSTAT            ; GET VOLUME
000067                  LDA         (VCBPTR),Y          ; STATUS
000068                  AND         #$FF-DSWITCH        ; TURN OFF DISK SWITCH
000069                  STA         (VCBPTR),Y          ; SO WE WON'T VERIFY NEXT TIME
000070  EXECUTE         JSR         GOCMD               ; EXECUTE COMMAND
000071                  BCC         GOODOP              ; BRANCH IF SUCCESSFUL
000072                  CMP         #XDISKSW            ; DISK SWITCH?
000073                  BNE         ERRORSYS            ; NO, REPORT SOME OTHER
000074                  LDY         #VCBSTAT            ; MARK VCB WITH SWITCH
000075                  LDA         (VCBPTR),Y
000076                  AND         #$FF-DSWITCH        ; TO ENSURE VOLUME VERIFIED
```



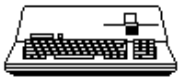
```
000077                  BPL         ERRCMD              ; NO FILES OPEN SO DSWITCH CANT APPLY
000078                  ORA         #DSWITCH
000079  ERRCMD          STA         (VCBPTR),Y
000080                  JMP         BFMGR               ; TRY THE COMMAND AGAIN
000081  *
000082  ERRORSYS        JSR         SYSERR
000083  GOODOP          RTS                             ; GOOD RETURN
000084  *
000085  GOCMD           JMP         (CMDADR)
000086  *
000087                  PAGE
000088  *
000089  CMDTABLE        EQU         *
000090                  DW          CREATE
000091                  DW          DESTROY
000092                  DW          RENAME
000093                  DW          SETINFO
000094                  DW          GETINFO
000095                  DW          VOLUME
000096                  DW          SETPREFX
000097                  DW          GETPREFX
000098                  DW          OPEN
000099                  DW          NEWLINE
000100                  DW          READ
000101                  DW          WRITE
000102                  DW          CLOSE
000103                  DW          FLUSH
000104                  DW          SETMARK
000105                  DW          GETMARK
000106                  DW          SETEOF
000107                  DW          GETEOF
000108  *
000109  DISPTCH         EQU         *
000110                  DFB         PREPATH+PRETIME+0   ; CREATE
000111                  DFB         PREPATH+PRETIME+1   ; DESTROY
000112                  DFB         PREPATH+PRETIME+2   ; RENAME
000113                  DFB         PREPATH+PRETIME+3   ; SETINFO
000114                  DFB         PREPATH+4           ; GETINFO
000115                  DFB         5                   ; VOLUME
000116                  DFB         6                   ; SETPREFIX, PATHNAME MOVED TO PREFIX BUFFER
000117                  DFB         7                   ; GETPREFIX
000118                  DFB         PREPATH+8           ; OPEN
000119                  DFB         PREREF+$9           ; NEWLINE
000120                  DFB         PREREF+$A           ; READ
000121                  DFB         PREREF+$B           ; WRITE
000122                  DFB         PRETIME+$C          ; CLOSE
000123                  DFB         PRETIME+$D          ; FLUSH, REFNUM MAY BE ZERO TO FLUSH ALL.
000124                  DFB         PREREF+$E           ; SETMARK
000125                  DFB         PREREF+$F           ; GETMARK
000126                  DFB         PREREF+$10          ; SET EOF
000127                  DFB         PREREF+$11          ; GET EOF
000128  *
000129                  PAGE
000130  *
000131  SETPATH         LDA         C.PATH              ; FOR A REGULAR PATHNAME,
000132                  STA         TPATH               ; SET UP TEMP POINTER TO PROCESS
000133                  LDA         C.PATH+1            ; PATHNAME AND CHECK FOR SYNTAX ERRORS
000134                  STA         TPATH+1
000135                  LDA         SISPATH
000136                  STA         SISTPATH            ; (LEAVE CALL PARAMETERS ALONE!)
000137  * DROP INTO 'SYNPATH'
000138  *
000139  SYNPATH         LDA         #>PATHBUF           ; SET UP DEFAULT ADDRESS FOR
000140                  STA         PATHNML             ; SYNTAXED PATHNAME -
000141                  STA         WRKPATH             ; LENGTH, NAME, LENGTH, NAME, ETC...
000142                  LDA         #<PATHBUF
000143                  STA         PATHNMH
000144                  STA         WRKPATH+1           ; (ASSUMES FULL PATHNAME, NO PREFIX).
000145                  LDX         #0                  ; USE INDEXED INDIRECT FOR SOURCE PATHNAME
000146                  TXA                             ; SET BEGINNING OF PATH
000147                  STA         (PATHNML,X)         ; TO ZERO TO INDICATE NOTHING PROCESSED.
000148                  TAY
000149                  LDA         (TPATH,X)           ; GET TOTAL LENGTH OF SOURCE PATHNAME
000150                  BMI         ERRSYN
000151                  BEQ         ERRSYN
000152                  STA         PATHCNT             ; (THIS IS USED AS A 'COUNT-DOWN')
000153                  JSR         INCTPTH             ; INCREMENT SOURCE POINTER
000154                  LDA         (TPATH,X)           ; GET FIRST CHARACTER OF PATHNAME
000155                  CMP         #DLIMIT             ; IS IT A FULL PATHNAME (NO PREFIX)?
000156                  BEQ         BUMPATH             ; YES, WE'RE READY TO DO IT.
000157                  CMP         #$2E                ; IS IT A DRIVE NAME '.'?
```



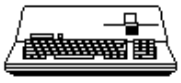
```
000158                  BNE     ADPREFIX            ; NO, ADD PREFIX TO BEGINNING
000159  DRIVENAM        LDA     (TPATH,X)           ; MOVE DRIVE NAME FOR VOLUME CALL
000160                  CMP     #DLIMIT             ; HAVE WE MOVED ENTIRE NAME?
000161                  BEQ     PREVOLM             ; YES, PROCESS IT.
000162                  INY                         ; (IF THIS IS THE FIRST, MAKE ROOM FOR LENGTH OF NAME)
000163                  STA     (WRKPATH),Y
000164                  JSR     INCTPTH             ; BUMP POINTER TO GIVEN NAME.
000165                  DEC     PATHCNT
000166                  BNE     DRIVENAM
000167                  BEQ     PREVOLM1
000168  *
000169                  PAGE
000170  PREVOLM         JSR     INCTPTH             ; MAKE IT SO POINTING PAST DELIMITER.
000171                  DEC     PATHCNT
000172  PREVOLM1        TYA                         ; SAVE LENGTH OF DRIVE NAME.
000173                  STA     (WRKPATH,X)
000174                  LDA     #>PATHBUF           ; POINT AT PATHNAME BUFFER FOR DEVICE ID CALL.
000175                  STA     DVNAMP
000176                  LDA     #<PATHBUF
000177                  STA     DVNAMP+1
000178                  LDA     #0                  ; MAKE VIRTUAL POINT AT SWITCHED IN BANK.
000179                  STA     SISTER+DVNAMP+1
000180                  JSR     SRCHDEV             ; GO IDENTIFY WHICH VOLUME
000181                  BCC     PREVOLM2            ; BRANCH IF NO ERROR
000182                  CMP     #VNFERR             ; WAS IT REPORTED AS 'VOLUME NOT FOUND'?
000183                  BNE     SPTHERR             ; NO SOME OTHER ERROR WAS ENCOUNTERED.
000184                  LDX     DUPLFLAG            ; YES, WAS IT NOT FOUND BECAUSE SOME OTHER 'OPEN' VOLUME HAS SAME NAME?
000185                  BEQ     SPTHERR             ; NO, IT SIMPLY WASN'T FOUND.
000186                  LDA     #DUPVOL             ; (CARRY IS SET)
000187                  RTS
000188  *
000189  PREVOLM2        LDY     #0                  ; (X CONTAINS AN INDEX TO VCB)
000190                  LDA     VCB,X               ; GET VOLUME NAME LENGTH.
000191                  STA     PATHBUF,Y
000192  SPATH2          INX                         ; MOVE VOLUME NAME INTO PATH NAME BUFFER IN
000193                  INY                         ; PLACE OF DISK DEVICE NAME ('.D1' OR SIMULAR)
000194                  LDA     VCB,X
000195                  STA     PATHBUF,Y
000196                  CPY     PATHBUF             ; HAVE ALL CHARACTERS BEEN MOVED?
000197                  BNE     SPATH2
000198                  LDX     #0                  ; RESET X FOR INDEXING
000199                  STX     PATHNML
000200                  LDA     #<PATHBUF
000201                  STA     PATHNMH
000202                  LDA     PATHCNT             ; IS THAT ALL THERE IS?
000203                  BNE     SPATH3              ; NO, MORE TO COME...
000204                  CLC
000205                  JMP     ENDPATH
000206  *
000207  SPATH3          INY                         ; BUMP TO END OF NAME+1
000208                  STY     WRKPATH             ; RESET WORKPATH POINTER TO CURRENT.
000209                  LDA     #0                  ; RESET PATHNAME BUFFER POINTER.
000210                  LDY     #<PATHBUF
000211                  BNE     NOPREFX             ; BRANCH ALWAYS...
000212  *
000213  ERRSYN          LDA     #BADPATH            ; RETURN SYNTAX ERROR
000214  SPTHERR         SEC
000215                  RTS
000216  *
000217  ADPREFIX        LDA     PFIXPTR             ; GET POINTER TO BEGINNING OF THE
000218                  LDY     PFIXPTR+1           ; PREFIX.
000219  NOPREFX         STA     PATHNML
000220                  STY     PATHNMH             ; IF NO PRESET PREFIX, THIS IS THE SAME AS
000221                  BNE     FRSTCHAR            ; PATHBUF ADDRESS. (BRANCH ALWAYS TAKEN)
000222  *
000223                  PAGE
000224  *
000225  BUMPATH         DEC     PATHCNT             ; FIRST ADJUST COUNT
000226                  CLC                         ; (JUST IN CASE OF LAST CHARACTER)
000227                  BEQ     ENDPATH             ; (MUST OF HAD TRAILING SPACES)
000228                  JSR     INCTPTH
000229  FRSTCHAR        LDY     #0                  ; INIT COUNT FOR THIS PORTION OF THE
000230                  TYA                         ; PATHNAME. ALSO PRESET LENGTH TO ZERO IN
000231                  STA     (WRKPATH,X)         ; CASE OF TRAILING SPACES.
000232                  LDA     (TPATH,X)           ; GET CHARACTER.
000233                  AND     #$7F                ; IGNORE HIGH BIT.
000234                  CMP     #$20                ; IS IT A LEADING SPACE?
000235                  BEQ     BUMPATH             ; IF SO, IGNORE IT.
000236                  CMP     #$5B                ; IS IT GREATER THAN (UPPER CASE) A 'Z'?
000237                  BCC     ALFA1               ; NO, MAKE SURE IT'S AN ALPHA CHARACTER
000238                  AND     #$5F                ; YES, ASSUME IT'S LOWER CASE, AND UPSHIFT
```



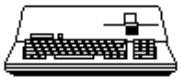
```
000239                  CMP     #$5B                ; WAS IT TRULY LOWER CASE?
000240                  BCS     ERRSYN              ; NO, GIVE ERROR.
000241  *
000242  ALFA1           CMP     #$41                ; IS IT LESS THAN 'A'?
000243                  BCC     ERRSYN              ; YES! IT'S CRAP...
000244                  BCS     SAVPATH             ; NO, IT'S GOOD. SAVE IT.
000245  *
000246  NXTCHAR         LDA     (TPATH,X)           ; GET THE NEXT CHARACTER.
000247                  AND     #$7F                ; THESE CHARACTERS MAY BE ALPHA, NUMERIC,
000248                  CMP     #$5B                ; OR A PERIOD - ONLY THE FIRST HAD TO BE ALPHA
000249                  BCC     ALFA2               ; BRANCH IF LESS THAN 'Z'
000250                  AND     #$5F                ; UPSHIFT LOWER CASE.
000251                  CMP     #$5B                ; NOW IS IT VALID?
000252                  BCS     ERRSYN              ; NOPE.
000253  *
000254  ALFA2           CMP     #$41                ; IS IT GREATER THAN 'A'?
000255                  BCS     SAVPATH             ; YUP, IT IS WORTH SAVIN.
000256                  CMP     #$3A                ; >9?
000257                  BCS     TSTDLIM             ; YES
000258                  CMP     #$30                ; NO, <0?
000259                  BCS     SAVPATH             ; NO, IT'S VALID NUMERIC.
000260  TSTDLIM         CMP     #DLIMIT             ; IS IT THE DELIMITER?
000261                  BEQ     ENDPATH             ; YES. CARRY SET INDICATES MORE TO COME.
000262                  CMP     #$2E                ; IS IT A '.' (PERIOD)?
000263                  BNE     ERRSYN              ; NO, IT'S AN ERROR (#@&###@!)
000264  SAVPATH         CLC
000265                  INY                         ; BUMP NAME LENGTH
000266                  STA     (WRKPATH),Y
000267                  DEC     PATHCNT             ; IF ZERO, THAT WAS THE LAST CHARACTER
000268                  BEQ     ENDPATH             ; (CARRY CLEAR INDICATES END OF PATH)
000269                  INC     TPATH               ; BUMP POINTER TO SOURCE PATHNAME.
000270                  BNE     NXTCHAR
000271                  INC     TPATH+1             ; HIGH ORDER, WHEN NECESSARY.
000272                  BNE     NXTCHAR             ; BRANCH ALWAYS.
000273                  PAGE
000274  *
000275  ENDPATH         TYA                         ; GET CURRENT NAME LENGTH
000276                  STA     (WRKPATH,X)         ; AND PUT IT IN FRONT OF NAME
000277                  BCC     LSTNAME             ; BRANCH IF THAT WAS THE LAST OF PATH
000278                  CMP     #$10                ; WAS THE NAME ILLEGALLY LONG?
000279                  BCS     ERRSYN1             ; YES, REPORT IT.
000280                  LDY     #0
000281                  SEC                         ; ADJUST WORK POINTER TO END OF PREVIOUS NAME.
000282                  ADC     WRKPATH
000283                  STA     WRKPATH             ; REPLACE OLD POINTER.
000284                  BCC     BUMPATH             ; DO NEXT NAME.
000285                  LDA     #TOOLONG            ; THIS IS A NEVER ERROR!
000286                  JSR     SYSDEATH            ; (NEVER RETURNS).
000287  *
000288  LSTNAME         BEQ     TSTVALD
000289                  CMP     #$10                ; MAKE SURE LAST ISN'T TOO LONG
000290                  BCS     ERRSYN1
000291                  INY                         ; PUT A ZERO AT END OF PROCESSED PATHNAME
000292                  LDA     #0
000293  TSTVALD         STA     (WRKPATH),Y
000294                  LDA     (PATHNML,X)         ; SURE THERE IS A PATHNAME
000295                  BEQ     ERRSYN1             ; IF NOT, REPORT ERROR.
000296                  CLC                         ; INDICATE NO ERROR.
000297                  RTS
000298  *
000299  ERRSYN1         JMP     ERRSYN
000300  *
000301  INCTPTH         INC     TPATH               ; POINT AT NEXT CHARACTER
000302                  BNE     INCPTH1
000303                  INC     TPATH+1
000304  INCPTH1         RTS
000305  *
000306                  PAGE
000307  SETPREFX        JSR     SETPATH             ; CALL IS MADE HERE SO A 'NUL' PATH MAY BE DETECTED.
000308                  BCC     SETPRFX1            ; BRANCH IF PATHNAME OK
000309                  TAX                         ; SAVE ERROR CODE
000310                  LDY     #0
000311                  LDA     (C.PATH),Y          ; TEST FOR A NUL PATHNAME
000312                  BEQ     RESETPFX            ; BRANCH IF PREFIX TO BE RESET.
000313                  TXA                         ; RESTORE ERROR CODE
000314                  RTS
000315  RESETPFX        STA     PFIXPTR
000316                  CLC
000317                  RTS
000318  SETPRFX1        LDA     PATHNML             ; MAKE SURE NAME STARTED WITH A '/' DELIMITER.
000319                  BNE     ERRSYN1             ; BRANCH IF IT DID.
```



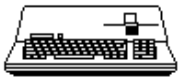
```
000320                  LDY     WRKPATH             ; FIND THE END OF THE INPUT PREFIX
000321                  CLC                         ; ADD LAST LOCAL NAME LENGTH TO FIND TRUE END.
000322                  LDA     (PATHNML),Y
000323                  BNE     SETPRFX3
000324                  DEY
000325                  TYA
000326                  BNE     SETPRFX4
000327  SETPRFX3        ADC     WRKPATH
000328                  TAY
000329  SETPRFX4        EOR     #$FF                ; GET COMPLIMENT TO FIND BEGINNING ADDRESS.
000330                  STA     PFIXPTR             ; OF NEW PREFIX IN THE PREFIX BUFFER
000331                  STA     WRKPATH             ; (PREFIX ALWAYS ENDS AT THE LAST BYTE OF BUFFER)
000332  MOVPRFX         LDA     (PATHNML),Y
000333                  STA     (WRKPATH),Y         ; MOVE IN NEW PREFIX
000334                  DEY
000335                  BPL     MOVPRFX
000336                  CLC                         ; AND WE'RE FINISHED!
000337                  RTS                         ; NO ERRORS POSIBLE FROM THIS ROUTINE.
000338  *
000339                  PAGE
000340  *
000341  GETPREFX        CLC                         ; CALCULATE HOW BIG A BUFFER IS NEEDED TO
000342                  LDA     PFIXPTR             ; PASS THE PREFIX BACK TO THE USER.
000343                  EOR     #$FF                ; (EVEN IF NO PREFIX, 1 BYTE IS NEEDED TO SHOW 0 LENGTH)
000344                  ADC     #2                  ; ADD 2 FOR LEADING AND ENDING "/".
000345                  CMP     C.MAXPTH            ; IS THERE ENOUGH SPACE IN USER'S BUFFER?
000346                  BCC     SENDPRFX            ; BRANCH IF YES
000347                  LDA     #BTSERR             ; TELL USER BUFFER IS TOO SMALL.
000348                  RTS                         ; (CARRY IS SET TO INDICATE ERROR.)
000349  *
000350  SENDPRFX        LDY     #0                  ; SAVE TOTAL LENGTH OF STRING TO BE RETURNED
000351                  STA     (C.PATH),Y
000352                  TAY
000353                  DEY                         ; DISCOUNT TRAILING DELIMITER.
000354                  BEQ     NULPREFX            ; BRANCH IF PREFIX IS SET TO NUL.
000355                  INY
000356                  LDX     PFIXPTR             ; GET BEGINNING ADDRESS OF PREFIX AGAIN
000357                  DEX
000358                  STX     WRKPATH
000359                  LDA     #<PATHBUF
000360                  STA     WRKPATH+1
000361  SNDLMIT         LDA     #DLIMIT             ; PLACE DELIMITER BEFORE, BETWEEN, AND AFTER LOCAL NAMES.
000362                  STA     (C.PATH),Y
000363  SNDPRFX1        DEY
000364                  BEQ     GOTPRFX             ; BRANCH IF ALL OF PREFIX IS TRANSFERED.
000365                  LDA     (WRKPATH),Y
000366                  STA     (C.PATH),Y          ; ASSUME IT'S A CHARACTER.
000367                  AND     #$F0                ; NOW TEST TO SEE IF IT WAS A LOCAL LENGTH.
000368                  BEQ     SNDLMIT             ; BRANCH IF IT WAS.
000369                  BNE     SNDPRFX1            ; GO MOVE NEXT CHAR IF IT WASN'T (ALWAYS TAKEN).
000370  NULPREFX        TYA                         ; RETURN NUL STRING.
000371                  STA     (C.PATH),Y
000372  GOTPRFX         CLC                         ; INDICATE NO ERROR.
000373                  RTS
000374                  PAGE
000375  *
000376  FINDFCB         LDA     FCBADDRH            ; INITIALIZE INDIRECT POINTER TO
000377                  STA     FCBPTR+1            ; FILE CONTROL BLOCK (ALLOCATED WHEN SYSTEM
000378                  LDA     #0                  ; WAS FIRST BOOTED).
000379                  STA     FCBPTR              ; NOTE: ALWAYS STARTS ON PAGE BOUNDARY.
000380                  LDA     FCBANKNM            ; SET SISTE PAGE BYTE TOO...
000381                  STA     SISFCBP
000382                  LDY     C.REFNUM            ; GET REQUESTED REFERENCE
000383                  BMI     ERRNOTBLK           ; BRANCH IF IT'S NOT A BLOCK DEVICE REFERENCE
000384                  DEY                         ; (SHOULD BE IN THE RANGE OF 1-16 BEFORE DECREMENT)
000385                  CPY     #$10                ; IS IT A VALID REFNUM?
000386                  BCS     REEFER              ; NO, THE USER'S SMOKIN DOPE!
000387                  TYA                         ; TO FIND ASSOCIATED FILE CONTROL STUFF,
000388                  ASL     A                   ; MULTIPLY (REFNUM-1) BY 32.
000389                  ASL     A
000390                  ASL     A
000391                  ASL     A
000392                  ASL     A
000393                  BCC     SVFCBLO             ; BRANCH IF IT'S WITHIN FIRST HALF OF FCB
000394                  INC     FCBPTR+1            ; BUMP TO SECOND HAVE (REFNUM>8)
000395  SVFCBLO         STA     FCBPTR              ; SAVE LOW ADDRESS OF REFERENCED FCB
000396                  LDA     C.REFNUM            ; NOW VERIFY THAT FILE IS OPEN.
000397                  LDY     #FCBREFN
000398                  CMP     (FCBPTR),Y          ; SHOULD BE EQUAL!
000399                  BNE     ERRNOREF            ; BRANCH IF THEY'RE NOT
000400  FNDFCBUF        LDY     #FCBBUFN            ; IT'S A LEGAL FILE, NOW SET UP
```



```
000401                  LDA     (FCBPTR),Y          ; INDIRECT POINTERS TO DATA
000402  GTBUFFRS        LDX     #DATPTR             ; (AND INDEX) BUFFER(S) IN ZERO PAGE
000403                  JSR     GETBUFADR           ; GET BUFFER ADDRESS UNLESS
000404                  BCS     REEFER1             ; BOB HAS BEEN SMOKIN DOPE...
000405                  LDA     #2                  ; (ASSUME AN INDEX BLOCK BUFFER IS ALSO PRESENT)
000406                  ADC     DATPTR+1
000407                  STA     TINDX+1
000408                  LDA     DATPTR
000409                  STA     TINDX
000410                  LDA     SISDATP
000411                  STA     SSTIDXH
000412                  LDY     #FCBDEVN
000413                  LDA     (FCBPTR),Y          ; MAKE SURE DEVICE
000414                  STA     D.DEV               ; NUMBER TEMPS MATCH
000415                  STA     DEVNUM              ; CURRENT FILE'S DEVICE
000416                  LDA     #0                  ; LOOK AT ALL VOLUMES LOGGED IN
000417  FNDFVOL         TAX
000418                  LDA     VCB+VCBDEV,X        ; GET VOLUMES DEVICE NUMBER
000419                  CMP     (FCBPTR),Y          ; HVE WE FOUND A MATCH.
000420                  BNE     FNDFV1
000421                  LDY     #FCBSWAP            ; SWAP BYTES
000422                  LDA     VCB+VCBSWAP,X       ; MISMATCH
000423                  CMP     (FCBPTR),Y          ; MEANS FILE BELONGS
000424                  BNE     FNDFV.1             ; TO ANOTHER VOLUME
000425                  LDA     VCB,X               ; IS THIS AN OPEN DEVICE?
000426                  BEQ     FNDFV.1             ; NO, TRY ANOTHER VOLUME
000427                  JSR     FVOLFOUND           ; YES, SAVE VCB ADDRESS
000428                  LDA     VCB+VCBSWAP,X       ; SWAPPED?
000429                  BEQ     REEFER1             ; NO, RETURN CALMLY TO USER
000430                  JSR     SWAPIN              ; YES, SWAP ME IN
000431                  BCC     REEFER1             ; RETURN WITHOUT ERROR
000432                  LDA     #XIOERROR           ; USER REFUSED TO MOUNT PROPER VOLUME
000433                  RTS
000434  *
000435  FNDFV.1         LDY     #FCBDEVN            ; RELOAD Y WITH DEVICE INDEX
000436  FNDFV1          TXA
000437                  CLC
000438                  ADC     #VCBSIZE
000439                  BCC     FNDFVOL             ; LOOP UNTIL FOUND
000440                  LDA     #VCBERR             ; OTHERWISE DIE A SYSTEM DEATH!
000441                  JSR     SYSDEATH
000442                  PAGE
000443  *
000444  ERRNOREF        LDA     #0                  ; DROP A ZERO INTO THIS FCB TO
000445                  STA     (FCBPTR),Y          ; SHOW FREE FCB
000446  *
000447  REEFER          LDA     #BADREFNUM          ; TELL USER THAT REQUESTED REFNUM
000448                  SEC                         ; IS ILLEGAL (OUT OF RANGE) FOR THIS CALL.
000449  REEFER1         RTS
000450  *
000451  ERRNOTBLK       LDA     #NOTBLKDEV          ; TELL USER THAT SPECIFIED DEVICE IS NOT A BLOCK DEVICE
000452                  SEC
000453                  RTS
000454  *
000455  SVCBADR         EQU     *
000456  FVOLFOUND       STX     VCBPTR
000457                  LDA     #VCB/256
000458                  STA     VCBPTR+1
000459                  CLC                         ; INDICATE LEGAL REFNUM
000460                  RTS
000461                  PAGE
000462  * NAME    : GETDNUM
000463  * FUNCTION: GET DEVICE NUMBER
000464  * INPUT   : DVNAMP SETUP
000465  * OUTPUT  : DEVNUM IN 'SCRTCH'
000466  *         : 'BPL' IF NOT BLOCK DEV
000467  *         : 'BCS' IF NO DEVICE
000468  * VOLATILE: ALL REGS
000469  *
000470  GETDNUM         EQU     *
000471                  LDA     #>SCRTCH+1          ; SET UP POINTER TO SCRATCH AREA
000472                  STA     DVDNUM              ; TO RECIEVE DEVICE NUMBER.
000473                  LDA     #SCRHIGH
000474                  STA     DVDNUM+1
000475                  LDA     #0                  ; PLACE A ZERO IN BANK BYTE SINCE
000476                  STA     SISTER+DVDNUM+1     ; IT'S NOT IN A BANK.
000477                  STA     VCBPTR+1
000478                  LDA     #4                  ; THE 'GET.DNUM' COMMAND.
000479                  STA     DHPCMD
000480                  JSR     RPEATIO0            ; CALL BOB FOR THE INFO.
000481                  RTS                         ; RETURN WITH DEVMGR CC'S
```

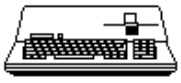


```
000482                  PAGE
000483  *
000484  * NAME    : SRCHDEV
000485  * FUNCTION: SEARCH FOR A VOLUME
000486  *
000487  SRCHDEV         EQU     *
000488                  JSR     GETDNUM             ; GET DEVNUM
000489                  BCS     VOLERR1             ; BRANCH IF ANY ERROR OTHER THAN NOTBLOCKDEV
000490                  BPL     ERRNOTBLK           ; BRANCH IF NOT A BLOCK DEVICE
000491                  LDA     #0                  ; NOW SEARCH FOR A VOL WITH THE
000492                  STA     NFOPEN              ; INIT TEMP VCB POINTER
000493  VOLOOK          TAX                         ; SAME DEVNUM AS SCRTCH
000494                  LDA     VCB+VCBSTAT,X       ; ANY FILES OPEN?
000495                  BNE     VLOOK00             ; BRANCH IF SOME FILE OPEN
000496                  STX     NFOPEN              ; ELSE SAVE THE VCB ENTRY PTR
000497  VLOOK00         EQU     *
000498                  LDA     VCB+VCBSWAP,X       ; VOLUME SWAPPED OUT?
000499                  BNE     VNOTEQ              ; YES, CANT BE THE ACTIVE VOL
000500                  LDA     VCB+VCBDEV,X
000501                  EOR     SCRTCH+1
000502                  BEQ     VLOOK0              ; BRANCH IF MATCH.
000503  VNOTEQ          LDA     VCB,X               ; IS THIS A FREE VCB?
000504                  BNE     VLOOK2              ; BRANCH IF NOT FREE, OTHEWISE TAKE NEXT BRANCH.
000505  VLOOK0          EOR     VCB,X               ; TEST FOR A VOLUME NAME LENGTH
000506                  BEQ     VLOOK1              ; BRANCH IF VCB FREE
000507                  JSR     SVCBADR             ; SAVE CURRENT ADDRESS OF VCB.
000508                  LDA     VCB+VCBSTAT,X       ; TEST FOR ANY OPEN FILES.
000509                  BPL     VLOOK3              ; LOG THE VOLUME IN JUST TO BE SURE
000510                  LDA     SCRTCH+1            ; SET UP
000511                  STA     DEVNUM              ; DEVICE NUMBER ARGUMENT
000512                  TXA                         ; SAVE PTR TO VCB
000513                  PHA                         ; ON STACK
000514                  JSR     VERFYVOL            ; COMPARES VCBPTR TO DEVNUM CONTENTS
000515                  BCC     VNOSWIT
000516                  CMP     #VNFERR             ; SEE IF NOTHING IN DRIVE
000517                  BEQ     VLOOK7              ; BRANCH IF NOTHING IN DRIVE
000518                  JSR     TSTSOS              ; IS THE VOLUME AN UNRECOGNIZED SOS OR (UCSD OR DOS)?
000519                  BCS     KNOTSOS             ; DEFINITELY NOT SOS FORMAT
000520                  LDX     #0                  ; START VCB SCAN AT BEGINNING
000521                  JSR     SNSWIT1             ; FIND A FREE VCB AND LOG IN THE NEW GUY
000522                  BCS     VNOSWIT1            ; CAN'T LOG IN NEW GUY--KEEP OLD
000523                  PLA
000524                  LDX     VCBPTR              ; PASS BACK X AS NEW VCB
000525                  RTS
000526  *
000527  NFOPEN          DS      1                   ; TEMP VCB PTR FOR VCB W/ NO FILES OPEN
000528  *
000529  VNOSWIT         CLC                         ; RETURN IT TO USER
000530                  PLA                         ; REMEMBER OLD VCB PTR
000531                  TAX                         ; AND PASS BACK TO USER
000532                  RTS
000533  ; RETURN TO CALLER X=POINTER TO VCB.
000534  *
000535  VOLERR1         SEC                         ; RETURN SOME VOLUME ERROR
000536                  RTS
000537  VNOSWIT1        CMP     #DUPVOL
000538                  BNE     VLOOK7              ; REPORT OTHER ERROR FROM LOGGING IN NEW VOL AS VNF
000539                  TAX
000540                  PLA                         ; MAKE STACK CORRECT
000541                  TXA                         ; RESTORE ERROR CODE
000542                  SEC
000543                  RTS                         ; IF DUPLICATE VOLUME ERROR, RETURN FACT TO USER
000544  KNOTSOS         PLA                         ; MAKE STACK CORRECT
000545                  LDA     #NOTSOS             ; FOR THE PASCAL FOLK
000546                  RTS                         ; NOTSOS MEANS UCSD OR DOS OR BAD SOS VOLUME
000547  *
000548  VLOOK7          PLA                         ; THROW AWAY OLD VCB PTR
000549                  JMP     NOVOLM              ; AND REPORT VOLUME NOT FOUND
000550  *
000551  VLOOK1          JSR     SVCBADR             ; SAVE ADDRESS OF FREE VCB.
000552  VLOOK2          TXA                         ; BUMP TO NEXT VOLUME ENTRY.
000553                  CLC
000554                  ADC     #VCBSIZE
000555                  BCC     VOLOOK              ; BRANCH IF MORE TO CHECK.
000556                  LDX     VCBPTR+1            ; FREE VCB YET FOUND?
000557                  BNE     VLOOK3              ; BRANCH IF YES
000558                  LDX     NFOPEN              ; SAVE POSSIBLE FREE VCB
000559                  JSR     SVCBADR             ; AND SAVE PTR PERMANENTLY
000560  VLOOK3          LDA     VCBPTR+1            ; WAS A FREE VCB FOUND?
000561                  BEQ     NOVOLM              ; BRANCH IF VOLUME CAN'T BE LOGGED IN.
000562                  LDA     SCRTCH+1            ; GET DEVICE NUMBER
```

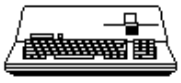


```
000563                  STA         DEVNUM              ; SAVE DEVICE NUMBER.
000564                  LDA         #1                  ; FAKE OUT 'LOKVOL'
000565                  STA         SCRTCH              ; TO THINK TO LOOK ONLY ONCE.
000566                  STA         TOTDEVS
000567                  LDA         #<VCB
000568                  STA         VCBPTR+1
000569                  STA         PATHNMH             ; (TO MAKE HARMLESS)
000570                  LDA         #0
000571                  STA         SISTER+PATHNMH
000572                  LDX         VCBPTR
000573                  STX         PATHNML
000574                  STA         VCB,X               ; FORCE CURRENT VOLUME OFF LINE, THEN LOG WHATS THERE.
000575                  JSR         FREEVCB             ; GO READ ROOT DIRECTORY.
000576                  BCS         RTVOLNAM            ; RETURN ANY ERRORS
000577                  LDX         VCBPTR              ; MAKE SURE VOLUME WAS LOGGED IN
000578                  LDA         VCB,X
000579                  BEQ         NOVOLM              ; RETURN ERROR
000580                  RTS                             ; ELSE RETURN NORMALLY
000581  NOVOLM          LDA         #VNFERR             ; TELL USER 'NO VOLUME'
000582                  SEC
000583  RTVOLNAM        TAX                             ; SAVE REAL ERROR WHILE DUPLICATE IS CHECKED
000584                  LDA         DUPLFLAG
000585                  BEQ         RTV1                ; BRANCH IF NOT DUPLICATE
000586                  LDX         #DUPVOL
000587  RTV1            TXA                             ; RECALL ERROR
000588                  RTS
000589
000590                  CHN         VOLUME,4,1
000591
000592  ********************************************************************************
000593  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: PATH
000594  ********************************************************************************
000595
000596
```

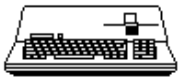
End of File -- Lines: 596 Characters: 24986



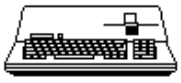
```
================================================================================
FILE: "SOS.POSN.OPEN.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: POSN.OPEN
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007  GETMARK         LDY     #FCBMARK            ; MOVE CURRENT POSITION MARKER TO
000008  GMARK1          LDA     (FCBPTR),Y          ; USER'S 4 BYTE BUFFER POINTED TO BY
000009                  PHA                         ; C.MRKPTR IN SOS ZPAGE
000010                  INY
000011                  CPY     #FCBMARK+3          ; USE STACK AS TEMPORARY STORAGE FOR THREE BYTE
000012                  BNE     GMARK1              ; POSITION VALUE.
000013                  LDA     #0                  ; THE FOURTH (HIGHEST ORDER) BYTE IS ALWAYS ZERO.
000014                  LDY     #3
000015                  PHA
000016  MOVMRK          PLA
000017                  STA     (C.MRKPTR),Y        ; MOVE TO USER'S SPACE
000018                  DEY                         ; IS THERE ANOTHER TO PULL FROM STACK?
000019                  BPL     MOVMRK              ; YES, GET NEXT LOWER BYTE FROM STACK.
000020                  CLC                         ; INDICATE NO ERROR.
000021                  RTS
000022  *
000023  SETMARK         JSR     ADJMARK             ; MAKE ADJUSTMENTS TO REQUESTED MARK ACCORDING TO BASE.
000024                  BCC     SMARK1              ; BRANCH IF ADJUSTMENT WAS VALID.
000025                  RTS
000026  SMARK1          LDX     #2                  ; NOW COMPARE END OF FILE WITH NEW
000027                  LDY     #FCBEOF+2           ; POSITION TO BE SURE IT'S WITHIN
000028  CMPEOF          LDA     TPOSLL,X            ; THE BOUNDS OF CURRENTLY DEFINED
000029                  CMP     (FCBPTR),Y          ; LIMITS.
000030                  BCC     CKSAMBLK            ; BRANCH IF MARK<EOF
000031                  BNE     ERRMEOF             ; RETURN ERROR IF MARK>= EOF
000032                  DEY
000033                  DEX
000034                  BPL     CMPEOF
000035                  BMI     CKSAMBLK            ; BRANCH ALWAYS
000036  ERRMEOF         LDA     #POSNERR            ; TELL USER MARK IS OUT OF RANGE.
000037                  RTS                         ; (CARRY IS SET TO INDICATE ERROR)
000038  *
000039  ADJMARK         LDA     C.MARK+3            ; MAKE SURE FOURTH BYTE OF DISPLACE IS ZIP
000040                  BNE     ERRPOSN             ; BRANCH TO ERR IF NOT
000041                  LDX     #$FD                ; ANTICIPATE OTHER THAN BASE OF ZERO
000042                  LDY     #FCBMARK            ; FURTHER ASSUME IT'S A BASE OFFSET FROM CURRENT POSITION
000043                  LDA     C.BASE              ; NOW FIND OUT WHAT IT REALLY IS.
000044                  LSR     A                   ; (CARRY SET=SUBTRACT, NON ZERO REMAINDER= OFFSET FROM EOF)
000045                  BCS     SUBMARK
000046                  BEQ     ADJMRK              ; BRANCH IF MARK IS FROM BEGINNING OF FILE
000047  ADDPOSN         LDA     (FCBPTR),Y          ; ADD USER QUANTITY TO CURRENT
000048                  ADC     C.MARK+3,X          ; POSITION TO FORM NEW POSITION.
000049                  STA     >TPOSLL-$FD,X       ; (NOTE: ZERO PAGE REFERENCE WRAPS AROUND IN Z-PAGE)
000050                  INY
000051                  INX
000052                  BNE     ADDPOSN             ; ADD ALL THREE BYTES
000053                  BCS     ERRPOSN             ; BRANCH IF OVERFLOW
000054                  BEQ     ADJMRK1             ; BRANCH ALWAYS
000055  *
000056                  PAGE
000057  SUBMARK         BNE     SUBPOSN             ; BRANCH IF IT'S AN OFFSET FROM CURRENT POSITION
000058                  LDY     #FCBEOF             ; OTHERWISE ASSUME OFFSET FROM END OF FILE.
000059  SUBPOSN         LDA     (FCBPTR),Y          ; SUBTRACT USER QUANTITY TO FORM
000060                  SBC     C.MARK+3,X          ; NEW POSITION. IF FINAL
000061                  STA     >TPOSLL-$FD,X       ; RESULT IS L.T. ZERO, THEN REPORT
000062                  INY                         ; POSITION ERROR...
000063                  INX
000064                  BNE     SUBPOSN
000065                  BCS     ADJMRK1             ; BRANCH IF LEGAL POSITION CALCULATED.
000066  ERRPOSN         LDA     #POSNERR
000067                  SEC                         ; INDICATE ERROR
000068                  RTS
000069  *
000070  ADJMRK          LDX     #2                  ; FIRST SET UP POSITION TEMPS USED
000071  ADJMRK0         LDA     C.MARK,X            ; BY BOTH POSITION ROUTINES
000072                  STA     TPOSLL,X
000073                  DEX
000074                  BPL     ADJMRK0
000075  ADJMRK1         CLC                         ; NO ERRORS
000076                  RTS
```



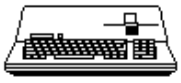
```
000077  *
000078  *
000079  RDPOSN          EQU     *
000080  CKSAMBLK        EQU     *
000081                  LDY     #FCBMARK+1          ; FIRST TEST TO SEE IF NEW POSITION IS
000082                  LDA     (FCBPTR),Y          ; WITHIN THE SAME (CURRENT) DATA BLOCK.
000083                  AND     #$FE
000084                  STA     SCRTCH
000085                  INY                         ; BUMP TO ACCESS HIGHEST ORDER ADDRESS BYTE
000086                  LDA     TPOSLH              ; GET MIDDLE BYTE OF NEW POSITION
000087                  SEC
000088                  SBC     SCRTCH
000089                  STA     SCRTCH
000090                  BCC     TYPMARK             ; BRANCH IF POSSIBLY L.T. CURRENT POSITION
000091                  CMP     #2                  ; MUST BE WITHIN 512 BYTES OF BEGINNING OF CURRENT
000092                  BCS     TYPMARK
000093                  LDA     TPOSHI              ; NOW MAKE SURE WERE TALKIN ABOUT
000094                  CMP     (FCBPTR),Y          ; THE SAME 64K CHUNK!
000095                  BNE     TYPMARK             ; BRANCH IF WE AREN'T.
000096                  JMP     SVMARK              ; IF WE IS, ADJUST FCB AND POSPTR AND RETURN.
000097  *
000098  TYPMARK         LDY     #FCBSTYP            ; NOW FIND OUT WHICH TYPE
000099                  LDA     (FCBPTR),Y          ; OF FILE WE'RE POSITIONING ON.
000100                  BEQ     FERRTYP             ; THERE IS NO SUCH TYPE AS ZERO, BRANCH NEVER!
000101                  CMP     #4                  ; IS IT A TREE CLASS FILE?
000102                  BCC     CHKDSKSW            ; YES, GO POSITION
000103                  JMP     DIRMARK             ; NO, TEST FOR DIRECTORY TYPE.
000104  *
000105  CHKDSKSW        EQU     *                   ; MAKE SURE S/HE HASN'T MOVED THE VOLUME
000106                  LDY     #FCBDEVN
000107                  LDA     (FCBPTR),Y
000108                  STA     DEVNUM              ; MAKE SURE DEVICE NUMBER PARM IS CURRENT
000109                  JSR     TWRPROT1            ; PASSES DEVNUM (CHECK DISK SWITCH)
000110                  LDA     DSWGLOB             ; DISK SWITCH GLOBAL
000111                  BEQ     TREPOS              ; BRANCH IF NONE DETECTED
000112  CHKDSKS1        JSR     VERFYVOL            ; MATCHES VCBPTR VS. DEVNUM
000113                  BCC     TREPOS              ; BRANCH IF DISK HASN'T SWITCHED
000114                  JSR     USRREQ              ; POLITELY ASK USER TO MOUNT
000115                  BCC     CHKDSKS1            ; SAID HE DID, CHECK AGAIN
000116                  LDA     #VNFERR             ; REFUSES TO MOUNT
000117                  RTS
000118  *
000119  FERRTYP         LDY     #FCBREFN            ; CLEAR ILLEGALLY TYPED FCB ENTRY
000120                  STA     (FCBPTR),Y
000121                  LDA     #BADREFNUM          ; TELL EM THERE IS NO SUCH FILE
000122                  SEC
000123                  RTS
000124  *
000125                  PAGE
000126  TREPOS          LDY     #FCBSTYP            ; USE STORAGE TYPE AS NUMBER
000127                  LDA     (FCBPTR),Y          ; OF LEVELS (SINCE 1=SEED, 2=SAPLING, AND 3=TREE)
000128                  STA     LEVELS
000129                  LDY     #FCBSTAT            ; SINCE IT'S A DIFFERENT DATA
000130                  LDA     (FCBPTR),Y          ; BLOCK, MUST NOT FORGET PREVIOUS DATA.
000131                  AND     #DATMOD             ; THEREFORE, SEE IF PREVIOUS DATA WAS MODIFIED
000132                  BEQ     POSNEW1             ; THEN DISK MUST BE UPDATED.
000133                  JSR     WFCBDAT             ; GO WRITE CURRENT DATA BLOCK.
000134                  BCS     POSERR              ; RETURN ANY ERROR ENCOUNTERED.
000135  *
000136  POSNEW1         LDY     #FCBMARK+2          ; TEST TO SEE IF CURRENT
000137                  LDA     (FCBPTR),Y          ; INDEX BLOCK IS GOING TO BE USABLE...
000138                  AND     #$FE                ; OR IN OTHER WORDS-
000139                  STA     SCRTCH              ; IS NEW POSITION WITHIN 128K OF THE BEGINNING
000140                  LDA     TPOSHI              ; OF CURRENT SAPLING LEVEL CHUNK.
000141                  SEC
000142                  SBC     SCRTCH
000143                  BCC     POSNEW2             ; BRANCH IF A NEW INDEX BLOCK IS ALSO NEEDED
000144                  CMP     #2                  ; NEW POSITION IS > THAN BEGINING OF OLD. IS IT WITHIN 128K?
000145                  BCS     POSNEW2             ; BRANCH IF NOT.
000146                  LDX     LEVELS              ; IS THE FILE WE'RE DEALING WITH A SEED?
000147                  DEX
000148                  BNE     DATLEVEL            ; NO, USE CURRENT INDEXES.
000149  TSTINY          LDA     TPOSLH              ; IS NEW POSITION UNDER 512?
000150                  LSR     A
000151                  ORA     TPOSHI
000152                  BNE     NOIDXDAT            ; NO, MARK BOTH DATA AND INDEX BLOCK AS UN-ALLOCATED.
000153                  LDY     #FCBFRST
000154                  LDA     (FCBPTR),Y          ; FIRST BLOCK IS ONLY BLOCK AND IT'S DATA!
000155                  STA     BLOKNML
000156                  INY
000157                  LDA     (FCBPTR),Y          ; (HIGH BLOCK ADDRESS)
```



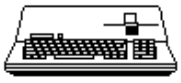
```
000158                JMP       RNEWPOS             ; GO READ IN BLOCK AND SET APPROPRIATE STATUSES.
000159  *
000160                PAGE
000161  POSNEW2       LDY       #FCBSTAT            ; GOTA CHECK TO SEE IF PREVIOUS
000162                LDA       (FCBPTR),Y          ; INDEX BLOCK WAS MODIFIED.
000163                AND       #IDXMOD
000164                BEQ       POSNIDX             ; READ IN OVER IT IF CURRENT IS UP TO DATE.
000165                JSR       WFCBIDX             ; GO UPDATE INDEX ON DISK (BLOCK ADDR IN FCB)
000166                BCS       POSERR
000167  POSNIDX       LDX       LEVELS              ; BEFORE READING IN TOP INDEX, CHECK TO BE SURE
000168                CPX       #3                  ; THAT THERE IS A TOP INDEX...
000169                BEQ       POSINDEX            ; BRANCH IF FILE IS FULL BLOWN TREE.
000170                LDA       TPOSHI              ; IS NEW POSITION WITHIN RANGE OF A
000171                LSR       A                   ; SAPLING FILE (L.T. 128K)?
000172                PHP                           ; ANTICIPATE NO GOOD.
000173                LDA       #TOPALC+IDXALC+DATALC ; (TO INDICATE NO LEVEL IS ALLOCATED FOR NEW POSITION.)
000174                PLP                           ; Z FLAG TELLS ALL...
000175                BNE       NODATA              ; GO MARK 'EM ALL DUMMY.
000176                JSR       CLRSTATS            ; GO CLEAR STATUS BITS 0,1,2 (INDEX/DATA ALLOC STATUS).
000177                DEX                           ; (UNAFFECTED SINCE LOADED ABOVE) CHECK FOR SEED
000178                BEQ       TSTINY              ; IF SEED, CHECK FOR POSITION L.T. 512...
000179                JSR       RFCBFST             ; GO GET ONLY INDEX BLOCK
000180                BCS       POSERR              ; BRANCH IF ERROR
000181                LDY       #FCBIDXB            ; SAVE NEWLY LOADED INDEX BLOCK'S ADDRESS
000182                LDA       BLOKNML
000183                STA       (FCBPTR),Y
000184                INY
000185                LDA       BLOKNMH
000186                STA       (FCBPTR),Y
000187                BCC       DATLEVEL            ; BRANCH ALWAYS...
000188  POSERR        SEC
000189                RTS
000190  *
000191  POSINDEX      JSR       CLRSTATS            ; CLEAR ALL ALLOCATION REQUIREMENTS FOR PREVIOUS POSITION
000192                JSR       RFCBFST             ; GET HIGHEST LEVEL INDEX BLOCK.
000193                BCS       POSERR
000194                LDA       TPOSHI              ; THEN TEST FOR A SAP LEVEL INDEX BLOCK
000195                LSR       A
000196                TAY
000197                LDA       (TINDX),Y
000198                INC       TINDX+1
000199                CMP       (TINDX),Y           ; (BOTH HI AND LO WILL BE ZERO IF NO INDEX EXISTS)
000200                BNE       SAPLEVEL
000201                CMP       #0                  ; ARE BOTH BYTES ZERO?
000202                BNE       SAPLEVEL
000203                DEC       TINDX+1             ; DON'T LEAVE WRONG POINTERS LAYING AROUND!
000204  NOIDXDAT      LDA       #IDXALC+DATALC      ; SHOW NEITHER INDEX OR DATA BLOCK ALLOCATED.
000205                JMP       NODATA
000206  *
000207                PAGE
000208  SAPLEVEL      STA       BLOKNML             ; READ IN NEXT LOWER INDEX BLOCK
000209                LDA       (TINDX),Y           ; (HI ADDRESS)
000210                STA       BLOKNMH
000211                DEC       TINDX+1
000212                JSR       RFCBIDX             ; READ IN SAPLING LEVEL
000213                BCS       POSERR
000214  DATLEVEL      LDA       TPOSHI              ; NOW GET BLOCK ADDRESS OF DATA BLOCK
000215                LSR       A
000216                LDA       TPOSLH              ; ( IF THERE IS ONE )
000217                ROR       A
000218                TAY
000219                LDA       (TINDX),Y           ; DATA BLOCK ADDRESS LOW
000220                INC       TINDX+1
000221                CMP       (TINDX),Y
000222                BNE       POSNEW3
000223                CMP       #0
000224                BNE       POSNEW3
000225                LDA       #DATALC             ; SHOW DATA BLOCK AS NEVER BEEN ALLOCATED
000226                DEC       TINDX+1
000227  *
000228  NODATA        LDY       #FCBSTAT
000229                ORA       (FCBPTR),Y          ; SET STATUS TO SHOW WHATS MISSIN'
000230                STA       (FCBPTR),Y
000231                LSR       A                   ; THROW AWAY BIT THAT SAYS DATA BLOCK UN-ALLOCATED
000232                LSR       A                   ; CUZ WE KNOW THAT. CARRY NOW INDICATES IF INDEX BLOCK
000233                JSR       ZIPDATA             ; ALSO IS INVALID AND NEEDS TO BE ZEROED (CARRY UNDISTURBED)
000234                BCC       SVMARK              ; BRANCH IF INDEX BLOCK DOESN'T NEED ZIPPIN.
000235  ZIPIDX        STA       (TINDX),Y
000236                INY
000237                BNE       ZIPIDX
000238                INC       TINDX+1
```



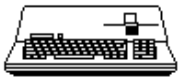
```
000239  ZPIDX1          STA         (TINDX),Y
000240                  INY
000241                  BNE         ZPIDX1
000242                  DEC         TINDX+1             ; RESTORE PROPER ADDRESS
000243                  JMP         SVMARK
000244  *
000245  ZIPDATA         LDA         #0                  ; ALSO IS INVALID AND NEEDS TO BE ZEROED.
000246                  TAY
000247  ZIPDAT0         STA         (DATPTR),Y          ; ZERO OUT DATA AREA
000248                  INY
000249                  BNE         ZIPDAT0
000250                  INC         DATPTR+1
000251  ZPDAT1          STA         (DATPTR),Y
000252                  INY
000253                  BNE         ZPDAT1
000254                  DEC         DATPTR+1
000255                  RTS
000256  *
000257                  PAGE
000258  *
000259  POSNEW3         STA         BLOKNML             ; GET DATA BLOCK OF NEW POSITION
000260                  LDA         (TINDX),Y           ; (HI ADDRESS)
000261                  DEC         TINDX+1
000262  RNEWPOS         STA         BLOKNMH
000263                  JSR         RFCBDAT
000264                  BCS         PRITZ               ; RETURN ANY ERROR
000265                  JSR         CLRSTATS            ; SHOW WHOLE CHAIN IS ALLOCATED
000266  SVMARK          LDY         #FCBMARK+2          ; UPDATE POSITION IN FILE CONTROL BLOCK
000267                  LDX         #2
000268  SVMRK1          LDA         (FCBPTR),Y          ; REMEMBER OLDMARK IN CASE
000269                  STA         OLDMARK-FCBMARK,Y   ; CALLING ROUTINE FAILS LATER
000270                  LDA         TPOSLL,X
000271                  STA         (FCBPTR),Y
000272                  DEY
000273                  DEX                             ; MOVE 3 BYTE POSITION MARKER
000274                  BPL         SVMRK1
000275  *
000276                  CLC                             ; LAST, BUT NOT LEAST, SET UP
000277                  LDA         DATPTR              ; INDIRECT ADDRESS TO BUFFER PAGE POINTED
000278                  STA         POSPTR              ; TO BY THE CURRENT POSITION MARKER.
000279                  LDA         TPOSLH
000280                  AND         #1
000281                  ADC         DATPTR+1
000282                  STA         POSPTR+1
000283                  LDA         SISDATP
000284                  STA         SISPOSP             ; SISTER PAGE BYTE ALSO.
000285                  RTS                             ; CARRY SHOULD ALWAYS BE CLEAR
000286  PRITZ           SEC                             ; RANDOM ERROR
000287                  RTS                             ; RETURN
000288  *
000289  *
000290  CLRSTATS        LDY         #FCBSTAT            ; CLEAR ALLOCATION STATES FOR DATA BLOCK
000291                  LDA         (FCBPTR),Y          ; AND BOTH LEVELS OF INDEXES.
000292                  AND         #$FF-TOPALC-IDXALC-DATALC
000293                  STA         (FCBPTR),Y          ; THIS SAYS THAT EITHER THEY EXIST CURRENTLY
000294                  RTS                             ; OR THAT THEY'RE UNNECESSARY FOR CURRENT POSITION.
000295  *
000296                  PAGE
000297  *
000298  DIRMARK         CMP         #DIRTYP             ; IS IT A DIRECTORY?
000299                  BEQ         DIRPOS              ; YES...
000300                  LDA         #CPTERR             ; NO, THERE IS A COMPATABLITY PROBLEM-
000301                  JSR         SYSERR              ; THE DAMN THING SHOULD OF NEVER BEEN OPENED!
000302  *
000303  DIRPOS          LDA         SCRTCH              ; RECOVER RESULTS OF PREVIOUS SUBTRACTION.
000304                  LSR         A                   ; USE DIFFERENCE AS COUNTER AS TO HOW MANY
000305                  STA         CNTENT              ; BLOCKS MUST BE READ TO GET TO NEW POSITION.
000306                  LDY         #FCBMARK+1          ; TEST FOR POSITION DIRECTION.
000307                  LDA         (FCBPTR),Y
000308                  CMP         TPOSLH              ; CARRY INDICATES DIRECTION...
000309                  BCC         DIRFWRD             ; IF SET, POSITION FORWARD.
000310  DIRVRSE         LDY         #0                  ; OTHERWISE, READ DIRECTORY FILE IN REVERSE ORDER.
000311                  JSR         DIRPOS1             ; READ PREVIOUS BLOCK.
000312                  BCS         DRPOSERR            ; BRANCH IF ANYTHING GOES WRONG.
000313                  INC         CNTENT              ; COUNT UP TO 128
000314                  BPL         DIRVRSE             ; LOOP IF THERE IS MORE BLOCKS TO PASS OVER.
000315                  BMI         SVMARK              ; BRANCH ALWAYS.
000316  *
000317  DIRFWRD         LDY         #2                  ; POSITION IS FORWARD FROM CURRENT POSITION.
000318                  JSR         DIRPOS1             ; READ NEXT DIRECTORY BLOCK.
000319                  BCS         DRPOSERR
```



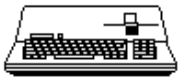
```
000320                  DEC         CNTENT
000321                  BNE         DIRFWRD             ; LOOP IF POSITION NOT FOUND IN THIS BLOCK.
000322                  BEQ         SVMARK              ; BRANCH ALWAYS.
000323  *
000324  DIRPOS1         LDA         (DATPTR),Y          ; GET LINK ADDRESS OF PREVIOUS OR
000325                  STA         BLOKNML             ; NEXT DIRECTORY BLOCK.
000326                  INY                             ; BUT FIRST BE SURE THERE IS A LINK.
000327                  CMP         (DATPTR),Y
000328                  BNE         DIRPOS2             ; BRANCH IF CERTAIN LINK EXISTS
000329                  CMP         #0                  ; ARE BOTHE LINK BYTES 0?
000330                  BNE         DIRPOS2             ; NOPE, JUST HAPPEN TO BE THE SAME VALUE.
000331                  LDA         #EOFERR             ; SOMETHING IS WRONG WITH THIS DIRECTORY FILE!
000332  DRPOSERR        SEC                             ; INDICATE ERROR
000333                  RTS
000334  *
000335  DIRPOS2         LDA         (DATPTR),Y          ; (HIGH ORDER BLOCK ADDRESS)
000336                  STA         BLOKNMH
000337  * DROP INTO 'RFCBDAT' (READ FILE'S DATA BLOCK)
000338  *
000339  * NOTE: FOR DIRECTORY POSITIONING NO OPTIMIZATION HAS BEEN
000340  * DONE SINCE DIRECTORY FILES WILL ALMOST ALWAYS BE LESS
000341  * THAN 6 BLOCKS. IF MORE SPEED IS REQUIRED OR DIRECTORY
000342  * TYPE FILES ARE TO BE USED FOR OTHER PURPOSES REQUIRING
000343  * MORE BLOCKS, THEN THE RECOMMENDED METHOD IS TO CALL
000344  * 'RFCBDAT' FOR THE FIRST BLOCK AND GO DIRECTLY TO
000345  * DEVICE (VIA JMP (IOUNITL)) HANDLER FOR SUBSEQUENT
000346  * ACCESSES.
000347  * ALSO NOTE THAT NO CHECKING IS DONE FOR READ/WRITE
000348  * ENABLE SINCE A DIRECTORY FILE CAN ONLY BE OPENED
000349  * FOR READ ACCESS.
000350  *
000351                  PAGE
000352  *
000353  RFCBDAT         LDA         #RDCMD              ; SET READ COMMAND.
000354                  STA         DHPCMD
000355                  LDX         #DATPTR             ; USE X TO POINT AT ADDRESS OF DATA BUFFER
000356                  JSR         FILEIO1             ; GO DO FILE INPUT.
000357                  LDY         #FCBDATB            ; SAVE BLOCK NUMBER JUST READ IN FCB.
000358                  BCC         FCBLOKNM            ; BRANCH IF NO ERRORS HAPPENED.
000359                  RTS                             ; RETURN ERROR
000360  *
000361  RFCBIDX         LDA         #RDCMD              ; PREPARE TO READ IN INDEX BLOCK.
000362                  STA         DHPCMD
000363                  LDX         #TINDX              ; POINT AT ADDRESS OF CURRENT INDEX BUFFER
000364                  JSR         FILEIO1             ; GO READ INDEX BLOCK.
000365                  BCS         RDFCBERR            ; REPORT ERROR
000366                  LDY         #FCBIDXB            ; SAVE BLOCK ADDRESS OF THIS INDEX IN FCB.
000367  FCBLOKNM        LDA         BLOKNML
000368                  STA         (FCBPTR),Y
000369                  INY
000370                  LDA         BLOKNMH
000371                  STA         (FCBPTR),Y
000372                  CLC
000373  RDFCBERR        RTS
000374  *
000375  RFCBFST         LDX         #TINDX              ; POINT AT ADDRESS OF INDEX BUFFER
000376                  LDY         #FCBFRST            ; AND BLOCK ADDRESS OF FIRST FILE BLOCK IN FCB
000377                  LDA         #RDCMD              ; AND LASTLY, MAKE IT A READ!
000378  * DROP INTO DOFILEIO
000379  *
000380  DOFILEIO        STA         DHPCMD              ; SAVE COMMAND.
000381                  LDA         (FCBPTR),Y          ; GET DISK BLOCK ADDRESS FROM FCB.
000382                  STA         BLOKNML
000383                  INY                             ; BLOCK ZERO NOT LEGAL.
000384                  CMP         (FCBPTR),Y
000385                  BNE         FILEIO
000386                  CMP         #0                  ; ARE BOTH BYTES ZERO?
000387                  BNE         FILEIO              ; NO, CONTINUE WITH REQUEST.
000388                  LDA         #ALCERR             ; OTHERWISE REPORT ALLOCATION ERROR.
000389                  JSR         SYSDEATH            ; NEVER RETURNS...
000390  *
000391                  PAGE
000392  FILEIO          LDA         (FCBPTR),Y          ; GET HIGH ADDRESS OF DISK BLOCK
000393                  STA         BLOKNMH
000394  FILEIO1         LDA         0,X                 ; GET MEMORY ADDRESS OF BUFFER FROM
000395                  STA         DBUFPL              ; S.O.S. ZERO PAGE POINTED TO BY
000396                  JSR         WRAPADJ             ;GO ADJUST FOR BANK CROSSING <SRS 82.162>
000397                  LDA         1,X
000398                  STA         DBUFPH              ; SET HI BYTE
000399                  LDA         SISTER+1,X          ; AND BANK PAIR BYTE. <SRS 82.162>
000400                  STA         SISBPH
```



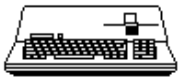
```
000401                  LDY     #FCBDEVN
000402                  LDA     (FCBPTR),Y          ; OF COURSE HAVING THE DEVICE NUMBER
000403                  STA     DEVNUM              ; WOULD MAKE THE WHOLE OPERATION MORE MEANINGFUL...
000404  FILEIO2         LDA     #2                  ; ALSO, SET UP BYTE COUNT TO 512 AND
000405                  STA     RQCNTH              ; SET 'BYTES READ' POINTER TO
000406                  STA     IOACCESS            ; (INTERUPT! SET TO INDICATE REG CALL MADE TO DEV HANDLER. RETURN
INTERUPT!)
000407                  LDA     #>TRASH             ; A PLACE TO THROW BYTES READ AWAY
000408                  STA     BRDPTR
000409                  LDA     #<TRASH             ; LOCALLY DEFINED
000410                  STA     BRDPTR+1
000411                  LDA     #0                  ; SO THAT IT DOESN'T MESS UP ANY OTHER DATA.
000412                  STA     RQCNTL
000413                  STA     SSBRDPH             ; ('BYTES READ' IS THROWN AWAY)
000414  RPEATIO1        LDA     DEVNUM              ; TRANSFER THE DEVICE NUMBER FOR DISPATCHER TO CONVERT TO UNIT NUMBER.
000415                  STA     UNITNUM
000416  RPEATIO0        LDY     #$9                 ; PREPARE TO SAVE DEVICE PARMS
000417  SAVPRMS         LDA     DEVICE,Y            ; MOVE FROM Z PAGE
000418                  STA     RPTBLOK,Y           ; TO MY OWN SPACE
000419                  DEY                         ; FROM $C9 THROUGH $C0
000420                  BPL     SAVPRMS
000421  DMGRGO          EQU     *                   ; CALL EXTERNAL DEVICE MANAGER
000422                  LDA     #0
000423                  STA     SERR                ; CLEAR GLOBAL ERROR VALUE
000424                  JSR     DMGR                ; CALL THE DRIVER
000425                  BCC     RRITZ               ; RTS IF NO ERRORS
000426                  CMP     #XDISKSW            ; DISKSWITCH ITERATES
000427                  BEQ     RPEATIO2            ; BRANCH IF DISK SWITCH AND REPEAT I/O REQUEST
000428                  SEC                         ; REPORT ERROR
000429  RRITZ           RTS
000430  RPEATIO2        LDY     #$9                 ; LENGTH OF PARM BLOCK
000431  GETPRMS         LDA     RPTBLOK,Y
000432                  STA     DEVICE,Y            ; RESTORE POSSIBLY DISTURBED PARM BLOCK
000433                  DEY
000434                  BPL     GETPRMS
000435                  JMP     DMGRGO              ; AND TRY THE I/O AGAIN
000436  *
000437  *
000438  TRASH           DS      2                   ; ONLY USED TO PUT BYTES READ TO SLEEP
000439  RPTBLOK         DS      10                  ; DMGR PARM SAVE BLOCK
000440  *
000441  *
000442  WFCBFST         LDY     #FCBDEVN            ; FETCH THE
000443                  LDA     (FCBPTR),Y          ;  DEVICE NUMBER
000444                  TAX                         ;    AND UPDATE
000445                  JSR     UPBMAP              ;      ITS BITMAP
000446                  LDX     #TINDX              ; POINT AT ADDRESS OF INDEX BLOCK
000447                  LDY     #FCBFRST            ; AND THE DISK ADDRESS OF FILE'S FIRST BLOCK IN FCB
000448                  LDA     #WRTCMD             ; LASTLY, MAKE IT A WRITE REQUEST.
000449                  JMP     DOFILEIO            ; AND GO DO IT!
000450  *
000451  WFCBDAT         LDX     #DATPTR
000452                  LDY     #FCBDATB            ; POINT AT MEMORY ADDRESS WITH X AND DISK ADDRESS WITH Y.
000453                  LDA     #WRTCMD             ; WRITE DATA BLOCK.
000454                  JSR     DOFILEIO
000455                  BCS     FILIOERR            ; REPORT ANY ERRORS
000456                  LDA     #$FF-DATMOD         ; MARK DATA STATUS AS CURRENT.
000457                  JMP     FCBUPDAT
000458  *
000459  WFCBIDX         LDY     #FCBDEVN            ; MAKE SURE
000460                  LDA     (FCBPTR),Y          ;  THE BITMAP
000461                  TAX                         ;    FOR THIS DEVICE ("X")
000462                  JSR     UPBMAP              ;      IS UPDATED
000463                  LDX     #TINDX              ; POINT AT ADDRESS OF INDEX BUFFER
000464                  LDY     #FCBIDXB            ; AND BLOCK ADDRESS OF THAT INDEX BLOCK.
000465                  LDA     #WRTCMD
000466                  JSR     DOFILEIO            ; GO WRITE OUT INDEX BLOCK.
000467                  BCS     FILIOERR            ; REPORT ANY ERRORS
000468                  LDA     #$FF-IDXMOD         ; MARK INDEX STATUS AS CURRENT.
000469  FCBUPDAT        LDY     #FCBSTAT            ; CHANGE STATUS BYTE TO
000470                  AND     (FCBPTR),Y          ; REFLECT SUCCESSFUL DISK FILE UPDATE.
000471                  STA     (FCBPTR),Y          ; (CARRY IS UNAFFECTED)
000472  FILIOERR        RTS
000473  *
000474  *
000475                  PAGE
000476  OPEN            JSR     FINDFILE            ; FIRST OF ALL LOOK UP THE FILE...
000477                  BCC     OPEN0
000478                  CMP     #BADPATH            ; IS AN ATTEMPT TO OPEN A ROOT DIRECTORY?
000479                  BNE     ERROPN              ; NO, PASS BACK ERROR
000480  *
```



```
000481  OPEN0           JSR       TSTOPEN              ; FIND OUT IF ANY OTHER FILES ARE WRITING
000482                  BCC       OPEN1                ; TO THIS SAME FILE. (BRANCH IF NOT)
000483  ERRBUSY         LDA       #FILBUSY             ; REPORT SHARED ACCESS NOT ALLOWED.
000484  ERROPN          SEC
000485                  RTS                            ; RETURN ERROR.
000486  *
000487  OPEN1           LDA       DATPTR               ; GET ADDRESS OF FIRST FREE FCB FOUND
000488                  STA       FCBPTR               ; DURING TEST OPEN SEQUENCE AND USE
000489                  LDA       DATPTR+1             ; IT AS FILE CONTROL AREA. IF HIGH BYTE OF
000490                  STA       FCBPTR+1             ; POINTER IS ZERO, THEN NO FCB
000491                  BNE       ASGNFCB              ; IS AVAILABLE FOR USE.
000492                  LDA       #FCBFULL             ; REPORT FCB FULL ERROR.
000493                  SEC
000494                  RTS
000495  *
000496  ASGNFCB         LDY       #$1F                 ; ASSIGN FCB, BUT FIRST
000497                  LDA       #0                   ; CLEAN OUT ANY OLD RUBBISH LEFT AROUND...
000498  CLRFCB          STA       (FCBPTR),Y
000499                  DEY
000500                  BPL       CLRFCB
000501                  LDY       #FCBENTN             ; NOW BEGIN CLAIM BY MOVING IN FILE
000502  FCBOWNR         LDA       D.DEV-1,Y            ; OWNERSHIP INFORMATION.
000503                  STA       (FCBPTR),Y           ; NOTE: THIS CODE DEPENDS UPON THE DEFINED
000504                  DEY                            ; ORDER OF BOTH THE FCB AND DIRECTORY ENTRY
000505                  BNE       FCBOWNR              ; BUFFER (D.). BEWARE OF CHANGES!!! *************
000506                  LDA       DFIL+D.STOR          ; GET STORAGE TYPE.
000507                  LSR       A                    ; STRIP OFF FILE NAME LENGTH.
000508                  LSR       A
000509                  LSR       A                    ; (BY DIVIDING BY 16)
000510                  LSR       A
000511                  TAX                            ; SAVE IN X FOR LATER TYPE COMPARISON
000512                  LDY       #FCBSTYP
000513                  STA       (FCBPTR),Y           ; SAVE STORAGE TYPE.
000514                  LDA       C.OPLSTLN            ; IS THERE AN OPEN LIST?
000515                  BEQ       DEFOPEN              ; NO, USE DEFAULT REQUST ACCESS...
000516                  LDY       #0                   ; YES, FIND OUT WHAT ACCESS IS REQUESTED.
000517                  LDA       (C.OPLIST),Y         ; IF REQ-ACCESS IS ZERO, THEN
000518                  BEQ       DEFOPEN              ; USE DEFAULTS...
000519                  AND       DFIL+D.ATTR          ; CHECK REQUEST AGAINST ATTRIBUTES.
000520                  CMP       (C.OPLIST),Y         ; WERE ALL ACCESS REQUESTS SATISFIED?
000521                  BEQ       SVATTRB              ; YES, SAVE ATTRIBUTES.
000522                  LDA       #ACCSERR             ; REPORT ACCESS REQUEST CAN'T BE MET.
000523                  SEC
000524                  RTS
000525                  PAGE
000526  DEFOPEN         LDA       DFIL+D.ATTR          ; GET FILES ATTRIBUTES AND
000527                  AND       #READEN+WRITEN       ; USE IT AS A DEFAULT ACCESS REQUEST.
000528  SVATTRB         LDY       #FCBATTR
000529                  CPX       #DIRTYP              ; IF DIRECTORY, DON'T ALLOW WRITE ENABLE
000530                  BNE       SVATTR1
000531                  AND       #READEN
000532  SVATTR1         STA       (FCBPTR),Y
000533                  AND       #WRITEN              ; CHECK FOR WRITE ENABLED REQUESTED.
000534                  BEQ       OPEN2                ; BRANCH IF READ ONLY OPEN.
000535                  LDA       TOTENT               ; OTHERWISE, BE SURE NO ONE ELSE IS READING SAME
000536                  BNE       ERRBUSY              ; FILE (SET UP BY TSTOPEN).
000537  OPEN2           LDA       DFIL+D.COMP          ; OH, BY THE WAY... IS THIS FILE
000538                  BEQ       OPEN3                ; COMPATABLE WITH VERSION 0000? ***************
000539  ERRCMPAT        LDA       #CPTERR              ; REPORT FILE IS INCOMPATABLE!
000540                  SEC
000541                  RTS
000542  *
000543  OPEN3           CPX       #TRETYP+1            ; IS IT A TREE TYPE FILE?
000544                  BCC       OPEN4                ; TEST FOR FURTHER COMPATABLITY. IT MUST
000545                  CPX       #DIRTYP              ; BE EITHER A TREE OR A DIRECTORY.
000546                  BNE       ERRCMPAT             ; REPORT INCOMPATABLE.
000547  OPEN4           LDY       #FCBFRST             ; MOVE ADDRESS OF FIRST BLOCK OF FILE
000548                  LDA       DFIL+D.FRST          ; INTO FCB. NO CHECKING IS DONE FOR VALIDITY.
000549                  STA       (FCBPTR),Y
000550                  STA       BLOKNML
000551                  INY
000552                  LDA       DFIL+D.FRST+1
000553                  STA       (FCBPTR),Y           ; NOTE: THE FCB HAS NOT BEEN OFFICIALLY
000554                  STA       BLOKNMH              ; CLAIMED YET. TO DO THIS, THE FIRST BYTE
000555                  LDY       #FCBEOF              ; MUST CONTAIN A VALID REFERENCE NUMBER.
000556  EOFCBMV         LDA       DFIL+D.EOF-FCBEOF,Y  ; MOVE CURRENT END OF FILE
000557                  STA       (FCBPTR),Y           ; TO FCB.
000558                  INY
000559                  CPY       #FCBEOF+3
000560                  BNE       EOFCBMV
000561                  LDA       DFIL+D.USAGE
```

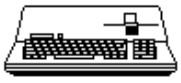


```
000562                  STA     (FCBPTR),Y          ; AND CURRENT BLOCK COUNT OF FILE.
000563                  INY
000564                  LDA     DFIL+D.USAGE+1
000565                  STA     (FCBPTR),Y
000566                  LDA     C.OPLSTLN           ; NOW THAT WE'VE COME THIS FAR, FIND
000567                  BEQ     DEFBUFR             ; OUT WHICH TYPE OF BUFFER AND ALLOCATE IT!
000568                  CMP     #1                  ; WAS IT ONLY TO SET ATTRIBUTES?
000569                  BEQ     DEFBUFR
000570                  CMP     #4                  ; IS A FULL ADDRESS INCLUDED?
000571                  BEQ     UBUFSPEC
000572                  LDA     #BADLSTCNT
000573                  SEC
000574                  RTS
000575  *
000576                  PAGE
000577  UBUFSPEC        LDY     #1                  ; (INDEX TO 'PAGECNT' OF OPEN LIST)
000578                  LDA     (C.OPLIST),Y        ; IS USER SPECIFING THE BUFFER?
000579                  BEQ     DEFBUFR             ; NO, USE DEFAULT BUFFER (DYNAMIC)
000580                  CPX     #TRETYP+1           ; IF TREE TYPE FILE, THEN AT LEAS 4 PAGES ARE NEEDED.
000581                  BCC     ONEKTST             ; BRANCH IF TREE TYPE.
000582                  CMP     #2                  ; DID USER GIVE AT LEAST 2 PAGES FOR DIRECTORY TYPE?
000583                  BCS     FIXDBUF             ; YES, LOG IT WITH BUFFER MANAGER
000584  ERRBTS          LDA     #BTSERR             ; REPORT NOT ENOUGH BUFFER SPACE.
000585                  SEC
000586                  RTS
000587  *
000588  ONEKTST         CMP     #4                  ; IS THERE AT LEAST ONE KILOBYTE BUFFER FOR TREES?
000589                  BCC     ERRBTS              ; NO, THEN TO HELL WITH IT!.
000590  FIXDBUF         JSR     REQFXBUF            ; CALL BOB AND ASK FOR HIM TO FIX IT...
000591                  BCC     FCBUFFER            ; GO SAVE BUFFER NUMBER.
000592  ERROPN1         RTS                         ; RETURN ANY ERROR ENCOUNTERED.
000593  *
000594  DEFBUFR         LDA     #4                  ; ASSUME TREE FILE (4 PAGES REQUIRED)
000595                  CPX     #TRETYP+1
000596                  BCC     BUFREQST            ; BRANCH IF IT IS A TREE.
000597                  LDA     #2                  ; OTHERWIZE, WE JUST NEED TWO PAGES.
000598  BUFREQST        JSR     REQBUF              ; CALL BOB TO ALLOCATE A DYNAMIC BUFFER.
000599                  BCS     ERROPN1             ; REPORT ANY ERRORS.
000600  FCBUFFER        LDY     #FCBBUFN            ; SAVE BUFFER NUMBER AND THEN
000601                  STA     (FCBPTR),Y          ; FIND OUT WHERE IT IS.
000602                  JSR     GTBUFFRS            ; HAVE BOB RETURN ADDRESS IN DATA & INDEX POINTERS.
000603                  BCS     ERROPEN2            ; IF ERROR, FREE BUFFER BEFOR RETURNING.
000604                  LDY     #FCBREFN            ; NOW CLAIM FCB FOR THIS FILE.
000605                  LDA     CNTENT              ; THIS WAS SET UP BY 'TSTOPEN'.............
000606                  STA     (FCBPTR),Y
000607                  LDY     #FCBLEVL            ; MARK LEVEL
000608                  LDA     LEVEL               ; AT WHICH
000609                  STA     (FCBPTR),Y          ; FILE WAS OPENED
000610                  LDY     #FCBSTYP            ; GET STORAGE TYPE AGAIN.
000611                  LDA     (FCBPTR),Y          ; FILE MUST BE POSITIONED TO BEGINNING.
000612                  CMP     #TRETYP+1           ; IS IT A TREE FILE?
000613                  BCS     OPNDIR              ; NO, ASSUME IT'S A DIRECTORY.
000614                  LDA     #$FF                ; FOOL THE POSITION ROUTINE INTO GIVING
000615                  LDY     #FCBMARK            ; A VALID POSITION WITH PRELOADED DATA, ETC.
000616  OPNPOS          STA     (FCBPTR),Y
000617                  INY
000618                  CPY     #FCBMARK+3
000619                  BNE     OPNPOS
000620                  LDY     #2                  ; SET DESIRED POSITION TO ZERO.
000621                  LDA     #0
000622  OPNPOS1         STA     TPOSLL,Y
000623                  DEY
000624                  BPL     OPNPOS1
000625                  JSR     RDPOSN              ; LET TREE POSITION ROUTINE DO THE REST.
000626                  BCC     OPENDONE            ; BRANCH IF SUCCESSFUL.
000627  *
000628                  PAGE
000629  ERROPEN2        PHA                         ; SAVE ERROR CODE.
000630                  LDY     #FCBBUFN            ; SINCE ERROR WAS ENCOUNTERED BEFORE FILE
000631                  LDA     (FCBPTR),Y          ; WAS SUCCESSFULLY OPENED, THEN
000632                  JSR     RELBUF              ; IT'S NECESSARY TO FREE THE BUFFER AND
000633                  LDY     #FCBREFN            ; FILE CONTROL BLOCK.
000634                  LDA     #0
000635                  STA     (FCBPTR),Y
000636                  PLA
000637                  SEC
000638                  RTS
000639  *
000640  OPNDIR          JSR     RFCBDAT             ; READ IN FIRST BLOCK OF DIRECTORY FILE.
000641                  BCS     ERROPEN2            ; RETURN ANY ERROR AFTER FREEING BUFFER & FCB
000642  OPENDONE        LDY     #VCBOPNC            ; INCREMENT OPEN COUNT FOR THIS
```

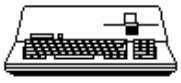


```
000643                  LDA         (VCBPTR),Y          ; VOLUME. ALSO MARK STATUS.
000644                  CLC
000645                  ADC         #1
000646                  STA         (VCBPTR),Y
000647                  LDY         #VCBSTAT            ; HI BIT INDICATES VOLUME BUSY
000648                  LDA         (VCBPTR),Y
000649                  ORA         #$80
000650                  STA         (VCBPTR),Y          ; DOESN'T MATTER HOW MANY, JUST BE SURE IT'S SET.
000651                  LDY         #FCBREFN            ; PASS USER HIS REFERENCE NUMBER
000652                  LDA         (FCBPTR),Y
000653                  LDY         #0
000654                  STA         (C.OUTREF),Y
000655                  CLC
000656                  RTS
000657  *
000658                  PAGE
000659  *
000660  TSTOPEN         LDA         FCBADDRH            ; TEST FOR SHARED ACCESS FILES WITH WRITE ENABLED.
000661                  STA         FCBPTR+1
000662                  LDA         FCBANKNM
000663                  STA         SISFCBP
000664                  LDA         #0
000665                  STA         DATPTR+1            ; MARK AS NO FREE FOUND.
000666                  STA         CNTENT
000667                  STA         TOTENT              ; ALSO, INIT COUNT OF MATCHING FILES
000668  TSTOPN1         STA         FCBPTR              ; SAVE NEW LOW ORDER ADDRESS
000669                  LDX         DATPTR+1            ; FIND OUT IF A FREE SPOT HAS BEEN FOUND YET.
000670                  BNE         TSTOPN2             ; YES, DON'T INCREMENT REFNUM (CNTENT).
000671                  INC         CNTENT              ; BUMP REFNUM
000672  TSTOPN2         LDY         #FCBREFN            ; TEST FOR IN USE FCB
000673                  LDA         (FCBPTR),Y          ; (NON ZERO)
000674                  BNE         CHKACTV             ; THIS FCB IS IN USE, COPARE OWNERSHIP.
000675                  TXA                             ; TEST AGAIN FOR FREE FCB
000676                  BNE         TSNXFCB             ; BRANCH IF A FREE SPOT HAS ALREADY BEEN FOUND.
000677                  LDA         FCBPTR              ; TRANSFER CURRENT POINTER SO IT MAY BE
000678                  STA         DATPTR              ; USED AS A FREE FCB BY OPEN.
000679                  LDA         FCBPTR+1            ; HIGH BYTE ALWAYS NON ZERO.
000680                  STA         DATPTR+1
000681                  JMP         TSNXFCB
000682  *
000683  CHKACTV         EQU         *                   ; IF MATCHING FILE IS SWAPPED, IT DOESNT COUNT
000684                  LDY         #FCBSWAP
000685                  LDA         (FCBPTR),Y
000686                  BNE         TSNXFCB             ; BRANCH IF SWAPPED
000687                  LDY         #FCBENTN            ; NOTE: THIS CODE DEPENDS ON THE
000688  WHOWNS          LDA         (FCBPTR),Y          ; DEFINED ORDER OF FCB AND DIRECTORY
000689                  CMP         D.DEV-1,Y           ; ****************************
000690                  BNE         TSNXFCB             ; BRANCH IF THIS ONE HAS A DIFFERENT OWNER.
000691                  DEY
000692                  BNE         WHOWNS
000693                  INC         TOTENT              ; REPORT THIS ONE AS A CO-OWNER.
000694                  LDY         #FCBATTR            ; NOW FIND OUT IF THIS ONE WANTS TO WRITE.
000695                  LDA         (FCBPTR),Y
000696                  AND         #WRITEN             ; IF WRITE IS NOT ENABLED THEN CONTINUE.
000697                  BEQ         TSNXFCB
000698                  SEC                             ; OTHERWISE, JUST SET THE CARRY TO SHOW
000699                  RTS                             ; THAT THE FILE CAN'T BE SHARED.
000700  *
000701  TSNXFCB         LDA         FCBPTR              ; CALCULATE NEXT FCB AREA (+$20)
000702                  CLC
000703                  ADC         #$20
000704                  BCC         TSTOPN1             ; LOOP IF NO PAGE CROSS.
000705                  LDX         FCBPTR+1
000706                  INC         FCBPTR+1
000707                  CPX         FCBADDRH            ; HAVE WE LOOKED AT BOTH PAGES?
000708                  BEQ         TSTOPN1             ; NOPE, LOOK AT PAGE TWO.
000709                  CLC                             ; INDICATE NO FILES THAT SHARE HAVE WRITE ENABLED,
000710                  RTS
000711  *
000712                  CHN         READ/WRITE,4,2
000713
000714  ********************************************************************************
000715  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: POSN.OPEN
000716  ********************************************************************************
000717
000718
```

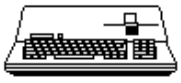
End of File -- Lines: 718 Characters: 31954



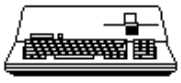
```
================================================================================
FILE: "SOS.PRINT.TEXT"
================================================================================

000001  ****************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: PRINT
000003  ****************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       'SOS 1.1 BLOCK FILE MANAGER' L
000007  * 01-FEB-82
000008                  REL
000009                  IBUFSIZ    1
000010                  SBUFSIZ    40
000011                  INCLUDE    SOSORG,6,1,254
000012                  ORG        ORGBFM               ; BITMAPS $B800-$BBFF
000013  ZZORG           EQU        *
000014                  REP        60
000015  *          (C) COPYRIGHT 1981 BY APPLE COMPUTER INC.
000016  *                      ALL RIGHTS RESERVED
000017                  REP        60
000018                  MSB        OFF
000019                  LST        VSYM
000020                  CHN        EQUATES,4,1
000021                  CHN        ALLOC
000022                  INCLUDE    POSN/OPEN
000023                  INCLUDE    READ/WRITE,2,,4
000024                  INCLUDE    CLOSE/EOF,2,,4
000025                  INCLUDE    DESTROY,2,,4
000026                  INCLUDE    SWAPOUT/IN,2,,4
000027
000028  ****************************************************************************
000029  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: PRINT
000030  ****************************************************************************
000031

End of File -- Lines: 31 Characters: 1032
```



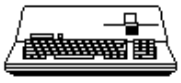
```
================================================================================
FILE: "SOS.PUBLICRELEASE.TEXT"
================================================================================

000001  ****************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: PUBLICRELEASE
000003  ****************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  :T 15,19,32
000007  ::PR#1,L58        132N
000008  ::SL4:DR1:ASM PRINT,BFM.OBJ,S6,D1
000009  ::END
000010
000011  ****************************************************************************
000012  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: PUBLICRELEASE
000013  ****************************************************************************

End of File -- Lines: 13 Characters: 531
```



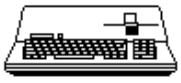
```
================================================================================
FILE: "SOS.READ.WRITE.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: READ.WRITE
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007  READ            CLC                             ; FIRST DETERMINE IF REQESTED
000008                  LDY         #FCBATTR            ; READ IS LEGAL
000009                  LDA         (FCBPTR),Y
000010                  AND         #READEN             ; IS READ ENABLED?
000011                  BNE         READ1               ; YES, CONTINUE...
000012                  LDA         #ACCSERR            ; REPORT ILLEGAL ACCESS.
000013                  SEC
000014                  RTS
000015  *
000016  READ1           LDY         #FCBMARK            ; GET CURRENT MARK INTO 'TPOS' AND
000017                  LDA         (FCBPTR),Y          ; DETERMINE IF RESULTING POSITION
000018                  STA         TPOSLL              ; EXCEEDS CURRENT END OF FILE.
000019                  ADC         C.BYTES
000020                  STA         SCRTCH
000021                  INY
000022                  LDA         (FCBPTR),Y
000023                  STA         TPOSLH
000024                  ADC         C.BYTES+1           ; (THIS WAS DONE STRAIT-LINE SINCE
000025                  STA         SCRTCH+1            ; WE'RE ADDING A TWO BYTE TO A THREE
000026                  INY                             ; BYTE QUANTITY)
000027                  LDA         (FCBPTR),Y
000028                  STA         TPOSHI
000029                  ADC         #0                  ; ADD IN REMAINING CARRY.
000030                  STA         SCRTCH+2
000031                  LDY         #FCBEOF+2           ; NOW TEST EOF AGAINST POSITION GENERATED
000032  EOFTEST         LDA         SCRTCH-FCBEOF,Y
000033                  CMP         (FCBPTR),Y          ; IS NEW POSITION > EOF?
000034                  BCC         READ2               ; NO, PROCEED.
000035                  BNE         ADJSTCNT            ; YES, ADJUST 'C.BYTES' REQUEST
000036                  DEY
000037                  CPY         #FCBEOF-1           ; HAVE WE COMPARED ALL TREE BYTES?
000038                  BNE         EOFTEST             ; NO, TEST NEXT LOWEST.
000039  ADJSTCNT        EQU         *                   ; ADJUST REQUEST TO READ UP TO (BUT
000040                  LDY         #FCBEOF             ; NOT INCLUDING) END OF FILE.
000041                  LDA         (FCBPTR),Y          ; RESULT= (EOF-1)-POSITION
000042                  SBC         TPOSLL
000043                  STA         C.BYTES
000044                  INY
000045                  LDA         (FCBPTR),Y
000046                  SBC         TPOSLH
000047                  STA         C.BYTES+1
000048                  ORA         C.BYTES             ; IF BOTH BYTES ARE ZERO, REPORT EOF ERROR.
000049                  BNE         READ2
000050                  LDA         #EOFERR
000051                  JSR         SYSERR
000052  READ2           LDA         C.BYTES
000053                  STA         RWREQL
000054                  BNE         READ3               ; BRANCH IF READ REQUEST DEFINITELY NON-ZERO.
000055                  CMP         C.BYTES+1
000056                  BNE         READ3               ; BRANCH IF READ REQUEST<>ZERO
000057                  STA         RWREQH
000058  GORDDNE         JMP         READONE             ; DO NOTHING.
000059                  PAGE
000060  *
000061  READ3           LDA         C.BYTES+1
000062                  STA         RWREQH
000063                  LDA         C.OUTBUF            ; MOVE POINTER TO USERS BUFFER TO BFM
000064                  STA         USRBUF              ; Z-PAGE AREA.
000065                  LDX         #C.OUTBUF           ; <SRS 82.162>
000066                  JSR         WRAPADJ             ; ADJUST FOR BANK CROSSING. <SRS 82.162>
000067                  STA         USRBUF+1
000068                  STY         SISUSRBF            ; SAVE VALID USER BUFFER ADDRESS (THAT WILL NOT CROSS BANKS)
000069                  LDY         #FCBSTYP            ; NOW FIND OUT IF IT'S A TREE READ OR OTHER.
000070                  LDA         (FCBPTR),Y
000071                  CMP         #TRETYP+1
000072                  BCC         TREAD               ; BRANCH IF A TREE FILE.
000073                  JMP         DREAD               ; OTHEWISE ASSUME IT'S A DIRECTORY.
000074  *
000075  TREAD           JSR         RDPOSN              ; GET DATA POINTER SET UP.
000076                  BCC         TREAD0              ; REPORT ANY ERRORS
```



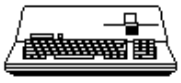
```
000077                  JMP     ERRFIX1
000078  TREAD0          JSR     PREPRW          ; TEST FOR NEWLINE, SETS UP FOR PARTIAL READ.
000079                  JSR     READPART        ; MOVE CURRENT DATA BUFFER CONTENTS TO USER AREA
000080                  BVS     GORDDNE         ; BRANCH IF REQUEST IS SATISFIED.
000081                  BCS     TREAD           ; CARRY SET INDICATES NEWLINE IS SET.
000082                  LDA     RWREQH          ; FIND OUT HOW MANY BLOCKS ARE TO BE READ
000083                  LSR     A               ; IF LESS THAN TWO, THEN DO IT THE SLOW WAY.
000084                  BEQ     TREAD
000085                  STA     BULKCNT         ; SAVE BULK BLOCK COUNT.
000086                  LDY     #FCBSTAT        ; MAKE SURE CURRENT DATA AREA
000087                  LDA     (FCBPTR),Y      ; DOESN'T NEED TO BE WRITTEN BEFORE
000088                  AND     #DATMOD         ; RESETTING POINTER TO READ DIRECTLY INTO
000089                  BNE     TREAD           ; USER'S AREA. BRANCH IF DATA NEED TO BE WRITTEN
000090                  STA     IOACCESS        ; TO FORCE FIRST CALL THRU ALL DEVICE HANDLER CHECKING.
000091                  LDA     USRBUF          ; MAKE THE DATA BUFFER THE USER'S SPACE.
000092                  STA     DATPTR
000093                  LDA     USRBUF+1
000094                  STA     DATPTR+1
000095                  LDA     SISUSRBF
000096                  STA     SISDATP
000097  *
000098                  PAGE
000099  RDFAST          JSR     RDPOSN          ; GET NEXT BLOCK DIRECTLY INTO USER SPACE.
000100                  BCS     ERRFIX          ; BRANCH ON ANY ERROR.
000101  RDFAST0         INC     DATPTR+1        ; BUMP ALL POINTERS BY 512 (ONE BLOCK)
000102                  INC     DATPTR+1
000103                  DEC     RWREQH
000104                  DEC     RWREQH
000105                  INC     TPOSLH
000106                  INC     TPOSLH
000107                  BNE     RDFAST1         ; BRANCH IF POSITION DOES NOT GET TO A 64K BOUNDARY.
000108                  INC     TPOSHI          ; OTHERWISE, MUST CHECK FOR A 128K BOUNDARY
000109                  LDA     TPOSHI          ; SET CARRY IF MOD 128K HAS BEEN REACHED
000110                  EOR     #1
000111                  LSR     A
000112  RDFAST1         DEC     BULKCNT         ; HAVE WE READ ALL WE CAN FAST?
000113                  BNE     RDFAST2         ; BRANCH IF MORE TO READ.
000114                  JSR     FXDATPTR        ; GO FIX UP DATA POINTER TO SOS BUFFER.
000115                  LDA     RWREQL          ; TEST FOR END OF READ.
000116                  ORA     RWREQH          ; ARE BOTH ZERO?
000117                  BEQ     READONE
000118                  BNE     TREAD           ; NO, READ LAST PARTIAL BLOCK.
000119  *
000120  RDFAST2         BCS     RDFAST
000121                  LDA     TPOSHI          ; GET INDEX TO NEXT BLOCK ADDRESS
000122                  LSR     A
000123                  LDA     TPOSLH
000124                  ROR     A
000125                  TAY                     ; INDEX TO ADDRESS IS INT(POS/512)
000126                  LDA     (TINDX),Y       ; GET LOW ADDRESS
000127                  STA     BLOKNML
000128                  INC     TINDX+1
000129                  CMP     (TINDX),Y       ; ARE BOTH HI AND LOW ADDRESS THE SAME?
000130                  BNE     REALRD          ; NO, IT'S A REAL BLOCK ADDRESS.
000131                  CMP     #0              ; ARE BOTH BYTES ZERO?
000132                  BNE     REALRD          ; NOPE -- MUST BE REAL DATA
000133                  STA     IOACCESS        ; DON'T DO REPEATIO JUST AFTER SPARSE
000134                  BEQ     NOSTUF          ; BRANCH ALWAYS (CARRY SET)
000135  REALRD          LDA     (TINDX),Y       ; GET HIGH ADDRESS BYTE
000136                  CLC
000137  NOSTUF          DEC     TINDX+1
000138                  BCS     RDFAST          ; BRANCH IF NO BLOCK TO READ
000139                  STA     BLOKNMH
000140                  LDA     IOACCESS        ; HAS FIRST CALL GONE TO DEVICE YET?
000141                  BEQ     RDFAST          ; NOPE, GO THRU NORMAL ROUTE...
000142                  LDA     DATPTR+1        ; RESET HI BUFFER ADDRESS FOR DEVICE HANDLER
000143                  STA     DBUFPH
000144                  JSR     REPEATIO
000145                  BCC     RDFAST0         ; BRANCH IF NO ERRORS.
000146                  PAGE
000147  ERRFIX          PHA                     ; SAVE ERROR CODE
000148                  JSR     FXDATPTR        ; GO RESTORE DATA POINTERS, ETC...
000149                  PLA
000150  ERRFIX1         PHA                     ; SAVE ERROR CODE
000151                  JSR     READONE         ; PASS BACK NUMBER OF BYTES ACTUALLY READ.
000152                  PLA
000153                  SEC                     ; REPORT ERROR
000154                  RTS
000155  *
000156  READONE         LDY     #0              ; RETURN TOTAL NUMBER OF BYTES ACTUALLY READ
000157                  SEC                     ; THIS IS DERIVED FROM C.BYTES-RWREQ
```



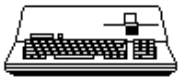
```
000158                  LDA     C.BYTES
000159                  SBC     RWREQL
000160                  STA     (C.OUTCNT),Y
000161                  INY
000162                  LDA     C.BYTES+1
000163                  SBC     RWREQH
000164                  STA     (C.OUTCNT),Y
000165                  JMP     RDPOSN              ; LEAVE WITH VALID POSITION IN FCB.
000166  *
000167  PREPRW          SEC                         ; ADJUST POINTER TO USER'S BUFFER TO
000168                  LDA     USRBUF              ; MAKE THE TRANSFER
000169                  SBC     TPOSLL
000170                  STA     USRBUF
000171                  BCS     PREPRW1             ; BRANCH IF NO ADJUSTMENT TO HI ADDR. NEEDED.
000172                  DEC     USRBUF+1            ; NOTE: SARA ALLOWS INDIRECT FROM $101 UP
000173  PREPRW1         LDY     #FCBATTR            ; AS LONG AS ACTUAL RESULTING ADDRESS IS >=$200
000174                  LDA     (FCBPTR),Y          ; TEST FOR NEW LINE ENABLED
000175                  AND     #NLINEN             ; SET CARRY IF IT IS.
000176                  CLC
000177                  BEQ     NONEWLIN            ; BRANCH IF NEWLINE IS NOT ENABLED
000178                  SEC
000179                  LDY     #FCBNEWL
000180                  LDA     (FCBPTR),Y          ; MOVE NEWLINE CHARACTER TO MORE
000181                  STA     NLCHAR              ; ACCESSABLE SPOT.
000182  NONEWLIN        LDY     TPOSLL              ; GET INDEX TO FIRST DATA
000183                  LDA     DATPTR              ; RESET LOW ORDER OF POSPTR TO BEGINNING OF PAGE.
000184                  STA     POSPTR
000185                  LDX     RWREQL              ; AND LASTLY GET LOW ORDER COUNT OF REQUESTED BYTES.
000186                  RTS                         ; RETURN STATUSES...
000187  *
000188  READPART        TXA
000189                  BNE     RDPART0             ; BRANCH IF REQUEST IS NOT A EVEN PAGES
000190                  LDA     RWREQH              ; A CALL OF ZERO BYTES SHOULD NEVER GET HERE!
000191                  BEQ     SETRDNE             ; BRANCH IF NOTHIN' TO DO.
000192                  DEC     RWREQH
000193  RDPART0         DEX
000194  RDPART          LDA     (POSPTR),Y          ; MOVE DATA TO USER'S BUFFER
000195                  STA     (USRBUF),Y          ; ONE BYTE AT A TIME.
000196                  TXA                         ; NOTE: THIS ROUTINE IS CODED TO BE
000197                  BEQ     ENDRQCHK            ; FASTEST WHEN NEWLINE IS DISABLED.
000198  RDPART1         BCS     TSTNEWL             ; BRANCH IF NEW LINE NEEDS TO BE TESTED.
000199  RDPART2         DEX
000200                  INY                         ; PAGE CROSSED?
000201                  BNE     RDPART              ; NO. MOVE NEXT BYTE.
000202                  LDA     POSPTR+1            ; TEST FOR END OF BUFFER
000203                  INC     USRBUF+1            ; BUT FIRST ADJUST USER BUFFER POINTER
000204                  INC     TPOSLH              ; AND POSITION.
000205                  BNE     RDPART3
000206                  INC     TPOSHI
000207  RDPART3         INC     POSPTR+1            ; AND SOS BUFFER HIGH ADDRESS.
000208                  EOR     DATPTR+1            ; (CARRY HAS BEEN CLEVERLY UNDISTURBED.)
000209                  BEQ     RDPART              ; BRANCH IF MORE TO READ IN BUFFER.
000210                  CLV                         ; INDICATE NOT FINISHED.
000211                  BVC     RDPRTDNE            ; BRANCH ALWAYS.
000212  *
000213  ENDRQCHK        LDA     RWREQH
000214                  BEQ     RDRQDNE             ; BRANCH IF REQEST SATISFIED.
000215                  INY                         ; DONE WITH THIS BLOCK OF DATA?
000216                  BNE     ENDRCHK1            ; NO, ADJUST HIGH BYTE OF REQUEST.
000217                  LDA     POSPTR+1            ; MAYBE- CHECK FOR END OF BLOCK BUFFER.
000218                  EOR     DATPTR+1            ; (DON'T DISTURB CARRY)
000219                  BNE     ENDRCHK2            ; BRANCH IF HI COUNT CAN BE DEALT WITH NEXT TIME.
000220  ENDRCHK1        DEC     RWREQH
000221  ENDRCHK2        DEY                         ; RESTORE PROPER VALUE TO 'Y'
000222                  JMP     RDPART1
000223  *
000224  TSTNEWL         LDA     (POSPTR),Y          ; GET LAST BYTE TRANSFERED AGAIN.
000225                  EOR     NLCHAR              ; HAVE WE MATCHED NEWLINE CHARACTER?
000226                  BNE     RDPART2             ; NO, READ NEXT.
000227  RDRQDNE         INY                         ; ADJUST POSITION.
000228                  BNE     SETRDNE
000229                  INC     USRBUF+1            ; BUMP POINTERS.
000230                  INC     TPOSLH
000231                  BNE     SETRDNE
000232                  INC     TPOSHI
000233  SETRDNE         BIT     SETVFLG             ; (SET V FLAG)
000234  RDPRTDNE        STY     TPOSLL              ; SAVE LOW POSITION
000235                  BVS     RDONE1
000236                  INX                         ; LEAVE REQUEST AS +1 FOR NEXT CALL
000237  RDONE1          STX     RWREQL              ; AND REMAINDER OF REQUEST COUNT.
000238                  PHP                         ; SAVE STATUSES
```



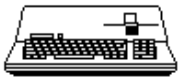
```
000239                  CLC                             ; ADJUST USER'S LOW BUFFER ADDRESS
000240                  TYA
000241                  ADC         USRBUF
000242                  STA         USRBUF
000243                  BCC         RDPART4
000244                  INC         USRBUF+1            ; ADJUST HI ADDRESS AS NEEDED.
000245  RDPART4         PLP                             ; RESTORE RETURN STATUSES
000246  SETVFLG         RTS                             ; (THIS BYTE <$60> IS USED TO SET V FLAG)
000247  *
000248  FXDATPTR        LDA         DATPTR              ; PUT CURRENT USER BUFFER
000249                  STA         USRBUF              ; ADDRESS BACK TO NORMAL
000250                  LDA         DATPTR+1
000251                  STA         USRBUF+1            ; BANK PAIR BYTE SHOULD BE MOVED ALSO.
000252                  LDA         SISDATP
000253                  STA         SISUSRBF
000254                  LDY         #FCBBUFN            ; RESTORE BUFFER ADDRESS
000255                  LDA         (FCBPTR),Y
000256                  LDX         #DATPTR
000257                  JMP         GETBUFADR           ; END VIA CALL TO BOB'S CODE.
000258  *
000259                  PAGE
000260  *
000261  * READ DIRECTORY FILE...
000262  *
000263  DREAD           JSR         RDPOSN
000264                  BCS         ERRDRD              ; PASS BACK ANY ERRORS
000265                  JSR         PREPRW              ; PREPARE FOR TRANSFER.
000266                  JSR         READPART            ; MOVE DATA TO USER'S BUFFER
000267                  BVC         DREAD               ; REPEAT UNTIL REQUEST IS SATISFIED.
000268                  JSR         READONE             ; UPDATE FCB AS TO NEW POSITION.
000269                  BCC         DREDONE             ; BRANCH IF ALL IS WELL.
000270                  CMP         #EOFERR             ; WAS LAST READ TO END OF FILE?
000271                  SEC                             ; ANTICIPATE SOME OTHER PROBLEM
000272                  BNE         DREDERR             ; BRANCH IF NOT EOF ERROR.
000273                  JSR         SVMARK
000274                  JSR         ZIPDATA             ; CLEAR OUT DATA BLOCK.
000275                  LDY         #FCBDATB+1          ; PROVIDE DUMMY BACK POINTER FOR FUTURE RE-POSITION
000276                  LDA         (FCBPTR),Y          ; GET HI BYTE OF LAST BLOCK.
000277                  PHA
000278                  DEY
000279                  LDA         (FCBPTR),Y          ; AND LOW BYTE.
000280                  PHA
000281                  LDA         #0                  ; NOW MARK CURRENT BLOCK AS IMPOSIBLE.
000282                  STA         (FCBPTR),Y
000283                  INY
000284                  STA         (FCBPTR),Y
000285                  TAY                             ; NOW MOVE LAST BLOCK ADDRESS TO DATA BUFFER AS BACK POINTER.
000286                  PLA
000287                  STA         (DATPTR),Y
000288                  PLA
000289                  INY
000290                  STA         (DATPTR),Y
000291  DREDONE         CLC                             ; INDICATE NO ERROR
000292  DREDERR         RTS
000293  *
000294  ERRDRD          JMP         ERRFIX1             ; REPORT HOW MUCH WE COULD TRANSFER BEFORE ERROR.
000295  *
000296                  PAGE
000297  WRITE           CLC                             ; FIRST DETERMINE IF REQESTED
000298                  LDY         #FCBATTR            ; WRITE IS LEGAL
000299                  LDA         (FCBPTR),Y
000300                  AND         #WRITEN             ; IS WRITE ENABLED?
000301                  BNE         WRITE1              ; YES, CONTINUE...
000302  ERRACCS         LDA         #ACCSERR            ; REPORT ILLEGAL ACCESS.
000303                  SEC
000304  WPERROR         RTS
000305  *
000306  WRITE1          JSR         TSTWPROT            ; OTHERWISE, MAKE SURE DEVICE IS NOT WRITE PROTECTED.
000307                  BCS         WPERROR             ; REPORT WRITE PROTECTED AND ABORT OPERATION.
000308  *
000309                  LDY         #FCBMARK            ; GET CURRENT MARK INTO 'TPOS' AND
000310                  LDA         (FCBPTR),Y          ; DETERMINE IF RESULTING POSITION
000311                  STA         TPOSLL              ; EXCEEDS CURRENT END OF FILE.
000312                  ADC         C.BYTES
000313                  STA         SCRTCH
000314                  INY
000315                  LDA         (FCBPTR),Y
000316                  STA         TPOSLH
000317                  ADC         C.BYTES+1           ; (THIS WAS DONE STRAIGHT-LINE SINCE
000318                  STA         SCRTCH+1            ; WE'RE ADDING A TWO BYTE TO A THREE
000319                  INY                             ; BYTE QUANTITY)
```



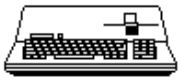
```
000320                  LDA     (FCBPTR),Y
000321                  STA     TPOSHI
000322                  ADC     #0                      ; ADD IN REMAINING CARRY.
000323                  STA     SCRTCH+2
000324                  LDY     #FCBEOF+2               ; NOW TEST EOF AGAINST POSITION GENERATED
000325  WEOFTST         LDA     SCRTCH-FCBEOF,Y
000326                  CMP     (FCBPTR),Y              ; IS NEW POSITION > EOF?
000327                  BCC     WRITE2                  ; NO, PROCEED.
000328                  BNE     WADJEOF                 ; YES, ADJUST END OF FILE
000329                  DEY
000330                  CPY     #FCBEOF-1               ; HAVE WE COMPARED ALL TREE BYTES?
000331                  BNE     WEOFTST                 ; NO, TEST NEXT LOWEST.
000332  WADJEOF         CLC                             ; ADJUST REQUEST TO WRITE UP TO (BUT
000333                  LDY     #FCBEOF                 ; NOT INCLUDING) END OF FILE.
000334  WRTADJEOF       LDA     (FCBPTR),Y              ; SAVE OLD EOF IN CASE OF LATER ERROR
000335                  STA     OLDEOF-FCBEOF,Y
000336                  LDA     SCRTCH-FCBEOF,Y         ; RESULT=EOF
000337  *
000338                  STA     (FCBPTR),Y
000339                  INY
000340                  CPY     #FCBEOF+3
000341                  BNE     WRTADJEOF
000342  WRITE2          LDA     C.BYTES
000343                  STA     RWREQL
000344                  BNE     WRITE3                  ; BRANCH IF WRITE REQUEST DEFINITELY NON-ZERO.
000345                  CMP     C.BYTES+1
000346                  BNE     WRITE3                  ; BRANCH IF WRITE REQUEST<>ZERO
000347                  STA     RWREQH
000348                  JMP     WRITDONE                ; DO NOTHING.
000349  *
000350                  PAGE
000351  WRITE3          LDA     C.BYTES+1
000352                  STA     RWREQH
000353                  LDA     C.OUTBUF                ; MOVE POINTER TO USERS BUFFER TO BFM
000354                  STA     USRBUF                  ; Z-PAGE AREA.
000355                  LDA     C.OUTBUF+1
000356                  STA     USRBUF+1                ; (SO IT MAY BE ADJUSTED WITHOUT LOOSING
000357                  LDA     SISOUTBF                ; ORIGINAL ADDRESS.)
000358                  STA     SISUSRBF
000359                  LDY     #FCBSTYP                ; NOW FIND OUT IF IT'S A TREE WRITE OR OTHER.
000360                  LDA     (FCBPTR),Y
000361                  CMP     #TRETYP+1
000362                  BCC     TWRITE                  ; BRANCH IF A TREE FILE.
000363                  JMP     ERRACCS                 ; OTHEWISE RETURN AN ACCESS ERROR!
000364  TWRITE          JSR     RDPOSN                  ; READ BLOCK WE'RE
000365                  BCS     WRITERROR
000366                  LDY     #FCBSTAT
000367                  LDA     (FCBPTR),Y
000368                  AND     #DATALC+IDXALC+TOPALC
000369                  BEQ     TREWRT1
000370                  LDY     #0                      ; FIND OUT IF ENOUGH DISK SPACE IS AVAILABLE FOR
000371  TWRTALC         INY                             ; INDEXES AND DATA BLOCK
000372                  LSR     A
000373                  BNE     TWRTALC
000374                  STY     REQL
000375                  STA     REQH
000376                  JSR     TSFRBLK
000377                  BCS     WRITERROR               ; PASS BACK ANY ERRORS.
000378                  LDY     #FCBSTAT
000379                  LDA     (FCBPTR),Y              ; NOW GET MORE SPECIFIC.
000380                  AND     #TOPALC                 ; ARE WE LACKING A TREE TOP?
000381                  BEQ     TSTSAPWR                ; NO, TEST FOR LACK OF SAPLING LEVEL INDEX.
000382                  JSR     TOPDOWN                 ; GO ALLOCATE TREE TOP AND ADJUST FILE TYPE.
000383                  BCC     DBLOKALC                ; CONTINUE WITH ALLOCATION OF DATA BLOCK.
000384  WRITERROR       PHA                             ; SAVE ERROR
000385                  LDY     #FCBEOF
000386  WRITERR01       LDA     OLDEOF-FCBEOF,Y
000387                  STA     (FCBPTR),Y              ; RESTORE OLD EOF UPON ERR
000388                  INY
000389                  CPY     #FCBEOF+3
000390                  BNE     WRITERR01
000391                  LDY     #FCBMARK
000392  WRITERR02       LDA     OLDMARK-FCBMARK,Y
000393                  STA     (FCBPTR),Y              ; AND RESTORE OLD MARK!
000394                  INY
000395                  CPY     #FCBMARK+3
000396                  BNE     WRITERR02
000397                  PLA
000398                  SEC
000399                  RTS                             ; ERROR RETURN
000400  *
```



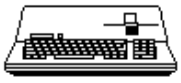
```
000401  TWRITEGO        BVC       TWRITE              ; A PIGGY-BACK BACKWARD BRANCH
000402  *
000403                  PAGE
000404  TSTSAPWR        LDA       (FCBPTR),Y          ; GET STATUS BYTE AGAIN.
000405                  AND       #IDXALC             ; DO WE NEED A SAPLING LEVEL INDEX BLOCK?
000406                  BEQ       DBLOKALC            ; NO, ASSUME IT'S JUST A DATA BLOCK NEEDED.
000407                  JSR       SAPDOWN             ; GO ALLOCATE AN INDEX BLOCK AND UPDATE TREE TOP.
000408                  BCS       WRITERROR           ; RETURN ANY ERRORS.
000409  DBLOKALC        JSR       ALCWBLK             ; GO ALLOCATE FOR DATA BLOCK.
000410                  BCS       WRITERROR
000411                  LDA       TPOSHI              ; CALCULATE POSITION WITHIN INDEX BLOCK.
000412                  LSR       A
000413                  LDA       TPOSLH
000414                  ROR       A
000415                  TAY                           ; NOW PUT BLOCK ADDRESS INTO INDEX BLOCK
000416                  INC       TINDX+1             ; HIGH BYTE FIRST.
000417                  LDA       SCRTCH+1
000418                  TAX
000419                  STA       (TINDX),Y
000420                  DEC       TINDX+1             ; (RESTORE POINTER TO LOWER PAGE OF INDEX BLOCK)
000421                  LDA       SCRTCH              ; GET LOW BLOCK ADDRESS
000422                  STA       (TINDX),Y           ; NOW STORE LOW ADDRESS.
000423                  LDY       #FCBDATB            ; ALSO UPDATE FILE CONTROL BLOCK TO INDICATE
000424                  STA       (FCBPTR),Y          ; THAT THIS BLOCK IS ALLOCATED.
000425                  INY
000426                  TXA                           ; GET HIGH ADDRESS AGAIN.
000427                  STA       (FCBPTR),Y
000428                  LDY       #FCBSTAT
000429                  LDA       (FCBPTR),Y
000430                  ORA       #IDXMOD
000431                  AND       #$FF-DATALC-IDXALC-TOPALC ; CLEAR ALLOCATION REQUIREMENT BITS.
000432                  STA       (FCBPTR),Y
000433  TREWRT1         LDX       #USRBUF             ; LOCATE POINTER TO ADJUST <SRS 82.162>
000434                  JSR       WRAPADJ             ; ADJUST FOR BANK CROSSING <SRS 82.162>
000435                  JSR       PREPRW              ; WRITE ON
000436                  JSR       WRTPART
000437                  BVC       TWRITEGO
000438  WRITDONE        JMP       RDPOSN              ; UPDATE FCB WITH NEW POSITION.
000439  *
000440                  PAGE
000441  WRTPART         TXA
000442                  BNE       WRPART              ; BRANCH IF REQUEST IS NOT A EVEN PAGES
000443                  LDA       RWREQH              ; A CALL OF ZERO BYTES SHOULD NEVER GET HERE!
000444                  BEQ       SETWRDNE            ; DO NOTHING!
000445  *
000446                  DEC       RWREQH
000447  WRPART          DEX
000448                  LDA       (USRBUF),Y          ; MOVE DATA FROM USER'S BUFFER
000449                  STA       (POSPTR),Y          ; ONE BYTE AT A TIME.
000450                  TXA
000451                  BEQ       ENDWQCHK
000452  WRPART2         INY                           ; PAGE CROSSED?
000453                  BNE       WRPART              ; NO. MOVE NEXT BYTE.
000454                  LDA       POSPTR+1            ; TEST FOR END OF BUFFER
000455                  INC       USRBUF+1            ; BUT FIRST ADJUST USER BUFFER POINTER
000456                  INC       TPOSLH              ; AND POSITION.
000457                  BNE       WRPART3
000458                  INC       TPOSHI
000459  WRPART3         INC       POSPTR+1            ; AND SOS BUFFER HIGH ADDRESS.
000460                  EOR       DATPTR+1            ; (CARRY HAS BEEN CLEVERLY UNDISTURBED.)
000461                  BEQ       WRPART              ; BRANCH IF MORE TO WRITE TO BUFFER.
000462                  CLV                           ; INDICATE NOT FINISHED.
000463                  BVC       WRPRTDNE            ; BRANCH ALWAYS.
000464  *
000465  ENDWQCHK        LDA       RWREQH
000466                  BEQ       WRTRQDNE            ; BRANCH IF REQEST SATISFIED.
000467                  INY                           ; ARE WE DONE WITH THIS BLOCK OF DATA?
000468                  BNE       ENDWCHK1            ; BRANCH IF NOT.
000469                  LDA       POSPTR+1
000470                  EOR       DATPTR+1            ; WHILE THIS IS REDUNDANT, IT'S NECESSARY FOR
000471                  BNE       ENDWCHK2            ; PROPER ADJUSTMENT OF REQUEST COUNT.
000472  ENDWCHK1        DEC       RWREQH              ; (NOT FINISHED- OK TO ADJUST HI BYTE.)
000473  ENDWCHK2        DEY                           ; RESET MODIFIED Y
000474                  JMP       WRPART2
000475  *
000476  WRTRQDNE        INY                           ; AND POSITION.
000477                  BNE       SETWRDNE
000478                  INC       USRBUF+1            ; BUMP POINTERS.
000479                  INC       TPOSLH
000480                  BNE       SETWRDNE
000481                  INC       TPOSHI
```



```
000482  SETWRDNE        BIT       SETVFLG             ; (SET V FLAG)
000483  WRPRTDNE        STY       TPOSLL              ; SAVE LOW POSITION
000484                  STX       RWREQL              ; AND REMAINDER OF REQUEST COUNT.
000485                  PHP                           ; SAVE STATUSES
000486                  LDY       #FCBSTAT
000487                  LDA       (FCBPTR),Y
000488                  ORA       #DATMOD+USEMOD
000489                  STA       (FCBPTR),Y
000490                  CLC                           ; ADJUST USER'S LOW BUFFER ADDRESS
000491                  LDA       TPOSLL
000492                  ADC       USRBUF
000493                  STA       USRBUF
000494                  BCC       WRPART4
000495                  INC       USRBUF+1            ; ADJUST HI ADDRESS AS NEEDED.
000496  WRPART4         JSR       FCBUSED             ; SET DIRECTORY FLUSH BIT
000497                  PLP                           ; RESTORE RETURN STATUSES
000498                  RTS
000499                  PAGE
000500  TOPDOWN         JSR       SWAPDOWN            ; FIRST MAKE CURRENT 1ST BLOCK AN ENTRY IN NEW TOP.
000501                  BCS       TPDWNERR            ; RETURN ANY ERRORS
000502                  LDY       #FCBSTYP            ; FIND OUT IF STORAGE TYPE HAS BEEN CHANGED TO 'TREE'.
000503                  LDA       (FCBPTR),Y          ; (IF NOT, ASSUME IT WAS ORIGINALLY A SEED AND
000504                  CMP       #TRETYP             ; BOTH LEVELS NEED TO BE BUILT.
000505                  BEQ       TOPDWN1             ; OTHERWISE, ONLY AN INDEX NEED BE ALLOCATED)
000506                  JSR       SWAPDOWN            ; MAKE PREVIOUS SWAP A SAP LEVEL INDEX BLOCK.
000507                  BCS       TPDWNERR
000508  TOPDWN1         JSR       ALCWBLK             ; GET ANOTHER BLOCK ADDRESS FOR THE SAP LEVEL INDEX.
000509                  BCS       TPDWNERR
000510                  LDA       TPOSHI              ; CALCULATE POSITION OF NEW INDEX BLOCK
000511                  LSR       A                   ; IN THE TOP OF THE TREE.
000512                  TAY
000513                  LDA       SCRTCH              ; GET ADDRESS OF NEWLY ALOCATED INDEX BLOCK AGAIN
000514                  TAX
000515                  STA       (TINDX),Y
000516                  INC       TINDX+1
000517                  LDA       SCRTCH+1
000518                  STA       (TINDX),Y           ; SAVE HI ADDRESS
000519                  DEC       TINDX+1
000520                  LDY       #FCBIDXB+1          ; MAKE NEWLY ALLOCATED BLOCK THE CURRENT INDEX BLOCK.
000521                  STA       (FCBPTR),Y
000522                  TXA
000523                  DEY
000524                  STA       (FCBPTR),Y
000525                  JSR       WFCBFST             ; SAVE NEW TOP OF TREE.
000526                  BCS       TPDWNERR
000527                  JMP       ZTMPIDX             ; END BY RE-CLEARING CURRENT (NEW) INDEX BLOCK.
000528  *
000529  SAPDOWN         LDY       #FCBSTYP            ; FIND OUT IF WE'RE DEALING WITH A TREE
000530                  LDA       (FCBPTR),Y          ; OR A SIMPLE SEED.
000531                  CMP       #SEEDTYP            ; IF SEED THEN AN ADJUSTMENT TO FILE TYPE IS NECESSARY.
000532                  BEQ       SAPDWN1             ; BRANCH IF SEED.
000533                  JSR       RFCBFST             ; OTHERWISE READ IN TOP OF TREE.
000534                  BCC       TOPDWN1             ; BRANCH IF NO ERROR.
000535  TPDWNERR        RTS                           ; RETURN ERRORS
000536  *
000537                  PAGE
000538  SAPDWN1         EQU       *                   ; MAKE CURRENT SEED INTO A SAPLING
000539  *
000540  SWAPDOWN        JSR       ALCWBLK             ; ALLOCATE A BLOCK BEFORE SWAP
000541                  BCS       SWAPERR             ; RETURN ERRORS IMMEDIATELY.
000542                  LDY       #FCBFRST            ; GET PREVIOUS FIRST BLOCK
000543                  LDA       (FCBPTR),Y          ; ADDRESS INTO INDEX BLOCK.
000544                  PHA                           ; SAVE TEMPORARLY WHILE SWAPPING IN NEW TOP INDEX
000545                  LDA       SCRTCH              ; GET NEW BLOCK ADDRESS (LOW)
000546                  TAX
000547                  STA       (FCBPTR),Y
000548                  INY
000549                  LDA       (FCBPTR),Y
000550                  PHA
000551                  LDA       SCRTCH+1            ; AND HIGH ADDRESS TOO.
000552                  STA       (FCBPTR),Y
000553                  LDY       #FCBIDXB+1          ; MAKE NEW TOP ALSO THE CURRENT INDEX IN MEMORY.
000554                  STA       (FCBPTR),Y
000555                  TXA                           ; GET LOW ADDRESS AGAIN
000556                  DEY
000557                  STA       (FCBPTR),Y
000558                  LDY       #0                  ; MAKE PREVIOUS THE FIRST ENTRY IN SUB INDEX
000559                  INC       TINDX+1
000560                  PLA
000561                  STA       (TINDX),Y
000562                  DEC       TINDX+1
```

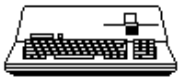


```
000563                  PLA
000564                  STA         (TINDX),Y
000565                  JSR         WFCBFST             ; SAVE NEW FILE TOP.
000566                  BCS         SWAPERR
000567                  LDY         #FCBSTYP            ; NOW ADJUST STORAGE TYPE
000568                  LDA         #1                  ; BY ADDING 1 (THUS SEED BECOMES SAPLING BECOMES TREE)
000569                  ADC         (FCBPTR),Y
000570                  STA         (FCBPTR),Y
000571                  LDY         #FCBSTAT
000572                  LDA         (FCBPTR),Y          ; MARK STORAGE TYPE MODIFIED.
000573                  ORA         #STPMOD
000574                  STA         (FCBPTR),Y
000575                  CLC                             ; RETURN 'NO ERROR' STATUS.
000576  SWAPERR         RTS
000577  *
000578                  PAGE
000579  ALCWBLK         JSR         ALC1BLK
000580                  BCS         ALUSERR
000581                  LDY         #FCBUSE
000582                  LDA         (FCBPTR),Y          ; BUMP CURRENT USAGE COUNT BY 1.
000583                  CLC
000584                  ADC         #1
000585                  STA         (FCBPTR),Y
000586                  BCC         INCUSG1
000587                  INY
000588                  LDA         (FCBPTR),Y
000589                  ADC         #0
000590                  STA         (FCBPTR),Y
000591  INCUSG1         LDY         #FCBSTAT            ; MARK USAGE AS MODIFIED.
000592                  LDA         (FCBPTR),Y
000593                  ORA         #USEMOD
000594                  STA         (FCBPTR),Y
000595                  CLC                             ; INDICATE NO ERROR
000596  ALUSERR         RTS                             ; ALL DONE
000597  *
000598  TSTWPROT        LDY         #FCBSTAT            ; CHECK FOR A 'NEVER BEEN MODIFIED' CONDITION
000599                  LDA         (FCBPTR),Y          ; GET STATUS BYTE
000600                  AND         #USEMOD+DATMOD+IDXMOD+EOFMOD
000601                  CLC                             ; ANTICIPATE WRITE OK
000602                  BNE         ALUSERR             ; ORDINARY RTS
000603                  LDY         #FCBDEVN            ; GET FILE'S DEVICE NUMBER
000604                  LDA         (FCBPTR),Y
000605                  STA         DEVNUM              ; GET CURRENT STATUS OF BLOCK DEVICE
000606  TWRPROT1        LDA         #STATCMD
000607                  STA         DHPCMD
000608                  LDA         #STATSUB            ; STORE SUB COMMAND OF STATUS CALL
000609                  STA         DSTATREQ
000610                  LDA         #>TWRCODE
000611                  STA         DSTATBFL            ; FETCH RETURN CODE IN SCRATCH AREA
000612                  LDA         #<TWRCODE
000613                  STA         DSTATBFH
000614                  LDA         #0                  ; MAKE SURE REGULAR RAM IS SELECTED (NO BANKS)
000615                  STA         SISDSTAT
000616                  STA         SERR                ; CLEAR GLOBAL ERROR FLAG
000617                  LDA         DEVNUM              ; SET UP LAST PARM
000618                  STA         UNITNUM             ; FOR DEVICE CALL
000619                  JSR         DMGR                ; MAKE THE EXTERNAL CALL
000620                  BCS         WPROTRET            ; RETURN ANY SPECIFIC ERRORS
000621                  LDA         TWRCODE             ; GET STATUS BYTE
000622                  LSR         A                   ; SHIFT WRITE PROTECT STATE INTO CARRY
000623                  LSR         A
000624                  LDA         #XNOWRITE           ; ANTICIPATE WRITE PROTECTED.
000625                  RTS                             ; CARRY IS INDETERMINATE
000626  WPROTRET        EQU         *
000627                  CMP         #XDISKSW            ; IF EXPLICITLY DISK SWITCH
000628                  BNE         WPROT1              ; BRANCH IF XNODRIVE OR XNOWRITE
000629                  STA         DSWGLOB             ; IF DISKSW, FLAG UNTIL ENTIRE OPERATION IS COMPLETE
000630                  CLC
000631                  RTS                             ; DISKSWITCH DOESNT SET CARRY
000632  WPROT1          SEC
000633                  RTS
000634  DSWGLOB         DS          1                   ; DISK SWITCH GLOBAL
000635  TWRCODE         DS          1                   ; A RARE EMBEDDED TEMP STORE
000636  *
000637                  PAGE
000638  *
000639  * MEMORY 'WRAP-AROUND' ADJUST ROUTINE.  THIS ROUTINE ADJUSTS
000640  * ADDRESSES THAT CROSS BANK PAIR BOUNDARIES.  ON ENTRY, X CONTAINS
000641  * THE OFFSET OF THE ZERO PAGE EXTENDED POINTER TO BE ADJUSTED.
000642  * ON EXIT, THE POINTER WILL HAVE BEEN ADJUSTED, IF NECESSARY,
000643  * AND THE ASSOCIATED X-BYTE WILL ALSO HAVE BEEN ADJUSTED.
```

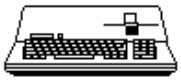


```
000644  * ONLY ADDRESSES IN THE RANGE $8200-$8E00 WILL BE ADJUSTED.
000645  *
000646  * UPON EXIT, A CONTAINS HIGH BYTE OF ADDRESS & Y CONTAINS UPDATED X-BYTE.
000647  * THIS ROUTINE LEAVES X UNCHANGED.
000648  *
000649  WRAPADJ         LDA        1,X                 ; GET HIGH ADDRESS BYTE <SRS 82.162>
000650                  LDY        SISTER+1,X          ; CHECK X-BYTE <SRS 82.162>
000651                  BPL        WRAPDNE             ; NOT AN EXTENDED ADDRESS. <SRS 82.162>
000652                  CMP        #$82                ; DOES IT NEED UPDATING? <SRS 82.162>
000653                  BCC        WRAPDNE             ; NO <SRS 82.162>
000654                  CPY        #$8F                ; SPECIAL BANK? <SRS 82.162>
000655                  BCS        WRAPDNE             ; NO <SRS 82.162>
000656                  AND        #$7F                ; ADJUST THE ADDRESS <SRS 82.162>
000657                  STA        1,X                 ; UPDATE <SRS 82.162>
000658                  INC        SISTER+1,X          ; INCREMENT X-BYTE <SRS 82.162>
000659                  INY                            ; UPDATE Y ALSO <SRS 82.162>
000660  *
000661  WRAPDNE         RTS                            ; RETURN VALID HIGH ADDRESS AND BANK BYTE.
000662
000663                  CHN        CLOSE/EOF,4,2
000664
000665  ************************************************************************
000666  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: READ.WRITE
000667  ************************************************************************
000668
000669
```

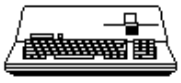
End of File -- Lines: 669 Characters: 27951



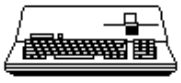
```
================================================================================
FILE: "SOS.SCMGR.SRC.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SCMGR.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1 SYSTEM CALL MANAGER"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGSCMGR
000010  ZZORG           EQU        *
000011                  MSB        OFF
000012                  REP        60
000013  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000014  *                     ALL RIGHTS RESERVED
000015                  REP        60
000016  *
000017  * SYSTEM CALL MANAGER (VERSION = 1.1O  )
000018  *                     (DATE    = 8/04/81)
000019  *
000020  * THE SYSTEM CALL MANAGER:
000021  * (1) RETRIEVE THE SYSCALL #,
000022  * (2) DETERMINE THE LOCATION OF THE SYSTEM CALL PARMS AND
000023  *     MOVE THEM TO THE SOS ZPAGE,
000024  * (3) TRANSFER CONTROL TO THE APPROPRIATE INTERFACE MANAGER,
000025  *     (FILE,DEVICE,UTILITY,MEMORY)
000026  *
000027                  REP        60
000028  *
000029                  ENTRY      SCMGR
000030  *
000031                  EXTRN      FMGR
000032                  EXTRN      DMGR
000033                  EXTRN      UMGR
000034                  EXTRN      MMGR
000035                  EXTRN      DBUGBRK
000036  *
000037                  EXTRN      SYSERR
000038                  EXTRN      SERR
000039                  EXTRN      BADSCNUM
000040                  EXTRN      BADCZPAGE
000041                  EXTRN      BADXBYTE
000042                  EXTRN      BADSCPCNT
000043                  EXTRN      BADSCBNDS
000044  *
000045                  EXTRN      SZPAGE
000046                  EXTRN      SXPAGE
000047                  EXTRN      CZPAGE
000048                  EXTRN      CXPAGE
000049                  EXTRN      CSPAGE
000050                  PAGE
000051                  REP        60
000052  *
000053  * SYSTEM CALL PARAMETER DEFINITION TABLES
000054  *
000055  * EACH ENTRY IS FOUR BYTES LONG.  THE FIRST BYTE CONTAINS THE
000056  * NUMBER OF PARMS IN THE CALL.  THE REMAINING SIX NIBBLES, EACH
000057  * DEFINE A PARAMETER IN THE CALL.  THE FIRST BIT OF THE
000058  * NIBBLE DEFINES WHETHER THE PARM IS INPUT (0) OR OUTPUT (1).
000059  * THE NEXT BIT DEFINES WHETHER THE PARM IS BY VALUE (0)
000060  * OR BY REFERENCE (1).  THE FINAL TWO BITS SPECIFY THE
000061  * PARM LENGTH IN BYTES (E.G. 0=LENGTH OF 1, 3=LENGTH OF 4 BYTES)
000062  *
000063                  REP        60
000064  *
000065  *   FILE SYSTEM CALL DEFINITIONS
000066  *
000067  FSC.CNT         EQU        $13
000068  FSC.TBL         EQU        *
000069                  DFB        $3,$5D,$00,$00       ; SCNUM=$C0 - CREATE
000070                  DFB        $1,$50,$00,$00       ;   "  =$C1 - DESTROY
000071                  DFB        $2,$55,$00,$00       ;   "  =$C2 - RENAME
000072                  DFB        $3,$5D,$00,$00       ;   "  =$C3 - SET.FILE.INFO
000073                  DFB        $3,$5D,$00,$00       ;   "  =$C4 - GET.FILE.INFO
000074                  DFB        $4,$55,$99,$00       ;   "  =$C5 - VOLUME
000075                  DFB        $1,$50,$00,$00       ;   "  =$C6 - SET.PREFIX
000076                  DFB        $2,$50,$00,$00       ;   "  =$C7 - GET.PREFIX
```



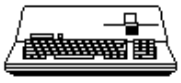
```
000077                  DFB         $4,$58,$D0,$00      ;   "  =$C8 - OPEN
000078                  DFB         $3,$00,$00,$00      ;   "  =$C9 - NEW.LINE
000079                  DFB         $4,$05,$19,$00      ;   "  =$CA - READ
000080                  DFB         $3,$05,$10,$00      ;   "  =$CB - WRITE
000081                  DFB         $1,$00,$00,$00      ;   "  =$CC - CLOSE
000082                  DFB         $1,$00,$00,$00      ;   "  =$CD - FLUSH
000083                  DFB         $3,$00,$30,$00      ;   "  =$CE - SET.MARK
000084                  DFB         $2,$0B,$00,$00      ;   "  =$CF - GET.MARK
000085                  DFB         $3,$00,$30,$00      ;   "  =$D0 - SET.EOF
000086                  DFB         $2,$0B,$00,$00      ;   "  =$D1 - GET.EOF
000087                  DFB         $1,$00,$00,$00      ;   "  =$D2 - SET.LEVEL
000088                  DFB         $1,$80,$00,$00      ;   "  =$D3 - GET.LEVEL
000089                  PAGE
000090  *
000091  *   DEVICE SYSTEM CALL DEFINITIONS
000092  *
000093  DSC.CNT         EQU         5
000094  DSC.TBL         EQU         *
000095                  DFB         $5,$05,$11,$90      ; SCNUM=$80 - D.READ
000096                  DFB         $4,$05,$11,$00      ;   "  =$81 - D.WRITE
000097                  DFB         $3,$00,$50,$00      ;   "  =$82 - D.STATUS
000098                  DFB         $3,$00,$50,$00      ;   "  =$83 - D.CONTROL
000099                  DFB         $2,$58,$00,$00      ;   "  =$84 - GET.DEV.NUM
000100                  DFB         $4,$05,$D0,$00      ;   "  =$85 - D.INFO
000101  *
000102  *   UTILITY SYSTEM CALL DEFINITIONS
000103  *
000104  USC.CNT         EQU         5
000105  USC.TBL         EQU         *
000106                  DFB         $1,$00,$00,$00      ; SCNUM=$60 - SET.FENCE
000107                  DFB         $1,$80,$00,$00      ;   "  =$61 - GET.FENCE
000108                  DFB         $1,$50,$00,$00      ;   "  =$62 - SET.TIME
000109                  DFB         $1,$50,$00,$00      ;   "  =$63 - GET.TIME
000110                  DFB         $2,$0B,$00,$00      ;   "  =$64 - JOYSTICK
000111                  DFB         $0,$00,$00,$00      ;   "  =$65 - COLD.START
000112  *
000113  *   MEMORY SYSTEM CALL DEFINITIONS
000114  *
000115  MSC.CNT         EQU         5
000116  MSC.TBL         EQU         *
000117                  DFB         $4,$11,$08,$00      ; SCNUM=$40 - REQUEST.SEG
000118                  DFB         $6,$00,$99,$98      ;   "  =$41 - FIND.SEG
000119                  DFB         $3,$00,$90,$00      ;   "  =$42 - CHANGE.SEG
000120                  DFB         $5,$09,$99,$80      ;   "  =$43 - GET.SEG.INFO
000121                  DFB         $2,$18,$00,$00      ;   "  =$44 - GET.SEG.NUM
000122                  DFB         $1,$00,$00,$00      ;   "  =$45 - RELEASE.SEG
000123  *
000124  *   DEBUG SYSTEM CALL DEFINITION
000125  *
000126  DBUG            EQU         $FE
000127                  PAGE
000128                  REP         60
000129  *
000130  * DATA DECLARATIONS
000131  *
000132                  REP         60
000133  Z.REG           EQU         $FFD0
000134  SP.SAVE         EQU         $01FF
000135  Z.SAVE          EQU         $01FD
000136  B.SAVE          EQU         $01FC
000137  *
000138  ADR.LOW         EQU         $2000               ; LOW    ADDRESS   (BOUNDS CHECKING)
000139  ADR.HIGH        EQU         $B800               ; HIGH   ADDRESS
000140  ADR.MID         EQU         $A000               ; MIDDLE ADDRESS
000141  *
000142  * SCMGR'S VARIABLES
000143  *
000144  SCM.VARS        EQU         $E0
000145  SCNUM           EQU         SCM.VARS+0          ; SYSTEM CALL NUMBER
000146  SCRNUM          EQU         SCM.VARS+0          ; SYSTEM CALL REQUEST NUMBER
000147  SCPTR           EQU         SCM.VARS+1          ;&2  SYSTEM CALL POINTER
000148  MOVE.VARS       EQU         SCPTR+2             ; !! (LOOKOUT) !!
000149  *
000150  *
000151  F.TPARMX        EQU         $A0                 ; FILE SYS CALL PARM START LOC
000152  D.TPARMX        EQU         $C0                 ; DEVICE SYS CALL PARM START LOC
000153  U.TPARMX        EQU         $C0                 ; UTILITY SYS CALL PARM START LOC
000154  M.TPARMX        EQU         $60                 ; MEMORY SYS CALL PARM START LOC
000155  *
000156  * MOVE.PARM'S VARIABLES
000157  *
```



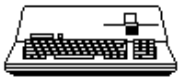
```
000158  TPARMX          EQU     MOVE.VARS+0         ; TARGET ADR OF SYS CALL PARMS
000159  DFN.PTR         EQU     MOVE.VARS+1         ;&2
000160  DFN.PTRX        EQU     MOVE.VARS+3
000161  SCPTRX          EQU     MOVE.VARS+4
000162  RGHT.NIB        EQU     MOVE.VARS+5
000163  SCT.DFN         EQU     MOVE.VARS+6
000164  SCT.DCNT        EQU     MOVE.VARS+7
000165  PARM.CNT        EQU     MOVE.VARS+8
000166                  PAGE
000167                  REP     60
000168  *
000169  * SYSTEM CALL MANAGER
000170  *
000171                  REP     60
000172  *
000173  SCMGR           EQU     *
000174                  LDA     #<SZPAGE            ; SET Z REG TO SOS ZPAGE
000175                  STA     Z.REG
000176  *
000177  * SET SYSTEM X BYTES TO ABSOLUTE ADDRESS MODE.
000178  *
000179                  LDA     #0
000180                  STA     SXPAGE+SCPTR+1
000181                  STA     SERR                ; AND INIT SYSTEM ERR CODE
000182  *
000183  * CALLER'S Z REG MUST BE $1A !!
000184  * (B REG NOT CHECKED)
000185  *
000186                  LDA     Z.SAVE
000187                  CMP     #<CZPAGE
000188                  BEQ     SCM005
000189                  LDA     #>BADCZPAGE
000190                  JSR     SYSERR              ; EXIT TO DISPATCHER
000191  *
000192  * RETRIEVE CALLER'S PC ON HIS STACK
000193  *
000194  SCM005          LDX     SP.SAVE
000195                  LDA     CSPAGE+6,X
000196                  STA     SCPTR+1
000197                  LDA     CSPAGE+5,X
000198                  STA     SCPTR
000199                  BNE     SCM010              ; AND POINT IT TO SYS CALL NUM
000200                  DEC     SCPTR+1
000201  SCM010          DEC     SCPTR
000202  *
000203  * ADVANCE CALLER'S PC ON HIS STACK.
000204  *
000205                  CLC
000206                  LDA     CSPAGE+5,X
000207                  ADC     #2
000208                  STA     CSPAGE+5,X
000209                  BCC     SCM020
000210                  INC     CSPAGE+6,X
000211  *
000212  * RETRIEVE SYSTEM CALL NUMBER
000213  *
000214  SCM020          LDY     #0
000215                  LDA     (SCPTR),Y
000216                  CMP     #DBUG
000217                  BNE     SCM025
000218                  JSR     DBUGBRK             ; DEBUG SYSTEM CALL
000219  SCM025          STA     SCNUM
000220  *
000221  * RETRIEVE SYSTEM CALL PARAMETER ADDRESS
000222  *
000223                  INY
000224                  LDX     #>SCPTR
000225                  JSR     POINTER
000226                  BCC     SCM030
000227                  RTS                         ; ERROR EXIT
000228  *
000229  * CASE INTERFACE CODE OF SYSTEM CALL NUMBER
000230  *  (INTERFACE CODE STRIPPED, LEAVING REQUEST CODE)
000231  *
000232  SCM030          LDA     #$20
000233                  BIT     SCNUM
000234                  BPL     SCM050
000235                  LDA     SCNUM
000236                  AND     #$3F
000237                  STA     SCRNUM
000238                  BVC     SCM040
```



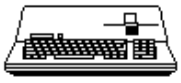
```
000239  *
000240                  LDA         #F.TPARMX           ; "11XXXXXX" - JMP TO FILE MANAGER.
000241                  STA         TPARMX
000242                  LDX         #>FSC.TBL
000243                  LDY         #<FSC.TBL
000244                  LDA         #FSC.CNT
000245                  JSR         MOVE.PARMS
000246                  BCS         SCM.ERR1            ; ERR EXIT
000247                  JMP         FMGR
000248  *
000249  SCM040          LDA         #D.TPARMX           ; "10XXXXXX" - JMP TO DEVICE MANAGER.
000250                  STA         TPARMX
000251                  LDX         #>DSC.TBL
000252                  LDY         #<DSC.TBL
000253                  LDA         #DSC.CNT
000254                  JSR         MOVE.PARMS
000255                  BCS         SCM.ERR1            ; ERR EXIT
000256                  JMP         DMGR
000257  *
000258  SCM050          BVC         SCM.ERR
000259                  PHP
000260                  LDA         SCNUM
000261                  AND         #$1F
000262                  STA         SCRNUM
000263                  PLP
000264                  BEQ         SCM060
000265  *
000266                  LDA         #U.TPARMX           ; "011XXXXX" - JMP TO UTILITY MANAGER.
000267                  STA         TPARMX
000268                  LDX         #>USC.TBL
000269                  LDY         #<USC.TBL
000270                  LDA         #USC.CNT
000271                  JSR         MOVE.PARMS
000272                  BCS         SCM.ERR1            ; ERR EXIT
000273                  JMP         UMGR
000274  *
000275  SCM060          LDA         #M.TPARMX           ; "010XXXXX" - JMP TO MEMORY MANAGER.
000276                  STA         TPARMX
000277                  LDX         #>MSC.TBL
000278                  LDY         #<MSC.TBL
000279                  LDA         #MSC.CNT
000280                  JSR         MOVE.PARMS
000281                  BCS         SCM.ERR1            ; ERR EXIT
000282                  JMP         MMGR
000283  *
000284  SCM.ERR         LDA         #>BADSCNUM          ; ERROR, INVALID SYSTEM CALL NUMBER.
000285  SCM.ERR1        JSR         SYSERR              ;   EXIT TO DISPATCHER ON ERROR
000286                  PAGE
000287                  REP         60
000288  *
000289  * MOVE.PARMS
000290  *
000291  * MOVES THE CALLER'S PARAMETERS TO THE OPERATING SYSTEM'S
000292  * ZERO PAGE, ACCORDING TO THE SPECIFICATIONS CONTAINED
000293  * IN THE SPECIFIED SYS CALL DFN TABLE.
000294  *
000295  * INPUT: (A) = MAX # ENTRIES IN PARM DFN TABLE
000296  *        (X) = PARM DFN TBL ADR (LO)
000297  *        (Y) =          "         (HI)
000298  *      SCPTR = ADR OF CALLER'S SYS CALL PARMS
000299  * ERROR: CARRY SET (SYSERR)
000300  *
000301                  REP         60
000302  *
000303  MOVE.PARMS      EQU         *
000304                  STX         DFN.PTR             ; SAVE ADR OF DEFINITION TABLE
000305                  STY         DFN.PTR+1
000306  *
000307  * IF REQ NUM > MAX REQ NUM (A REG)
000308  *
000309                  CMP         SCRNUM
000310                  BCS         MOVE010
000311  *
000312  *   THEN ERR(BAD SYS CALL NUM)
000313  *
000314                  LDA         #>BADSCNUM
000315                  BCC         SYSERR1             ;BRANCH ALWAYS TAKEN
000316  *
000317  * CALCULATE DEFINITION TABLE INDEX
000318  *  AND INIT SYS CALL PARM INDEX
000319  *
```



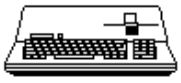
```
000320  MOVE010         LDA        SCRNUM
000321                  ASL        A
000322                  ASL        A
000323                  STA        DFN.PTRX
000324                  LDA        #0
000325                  STA        SXPAGE+DFN.PTR+1     ; AND X BYTE
000326                  STA        SCPTRX
000327  *
000328  * IF SCPTR(SCPTRX)<>DFN.PTR(DFN.PTRX) THEN ERR
000329  *
000330                  TAY
000331                  LDA        (SCPTR),Y
000332                  LDY        DFN.PTRX
000333                  CMP        (DFN.PTR),Y
000334                  BEQ        INITLOOPCT
000335  *
000336                  LDA        #>BADSCPCNT          ; ERR, CALLER'S PARM COUNT INVALID
000337  SYSERR1         JSR        SYSERR               ; EXIT
000338  *
000339  * INIT LOOP CTR(PARM.CNT) TO # OF PARMS IN SYS CALL
000340  *
000341  INITLOOPCT      STA        PARM.CNT
000342  *
000343  * ADVANCE PTRS
000344  *
000345  *
000346                  INC        SCPTRX
000347                  INC        DFN.PTRX
000348  *
000349  * MOVE REQ CODE TO SYS ZPAGE PARM LIST
000350  *  AND ADVANCE SYS ZPAGE PTR (X=TPARMX)
000351  *
000352                  LDA        SCRNUM
000353                  LDX        TPARMX
000354                  STA        0,X
000355                  INX
000356  *
000357  * INIT NIBBLE FLAG TO "RIGHT" NIBBLE
000358  *  ZERO STATE="LEFT" NIBBLE
000359  *
000360                  LDA        #$FF
000361                  STA        RGHT.NIB
000362                  REP        60
000363  *
000364  *  BEGIN PARAMETER PROCESSING LOOP
000365  *
000366  PARMLOOP        LDA        RGHT.NIB
000367                  EOR        #$FF                 ; COMPLEMENT NIBBLE FLAG
000368                  STA        RGHT.NIB
000369  *
000370  * IF "LEFT" NIBBLE
000371  *
000372                  BNE        ELSE.RNIB
000373  *
000374  * THEN FETCH SYS CALL PARM DFN
000375  *  AND # OF BYTES IN PARM WITHIN IT
000376  *
000377                  LDY        DFN.PTRX
000378                  LDA        (DFN.PTR),Y
000379                  STA        SCT.DFN
000380                  AND        #$30
000381                  LSR        A
000382                  LSR        A
000383                  LSR        A
000384                  LSR        A
000385                  STA        SCT.DCNT
000386                  BPL        VALUE                ;BRANCH ALWAYS
000387  *
000388  * ELSE FETCH SYS CALL PARM DFN
000389  *  AND # OF BYTES IN PARM WITHIN IT
000390  *  FROM "RIGHT" NIBBLE OF DFN BYTE
000391  *
000392  ELSE.RNIB       LDA        SCT.DFN
000393                  TAY
000394                  AND        #$03
000395                  STA        SCT.DCNT
000396                  TYA
000397                  ASL        A
000398                  ASL        A
000399                  ASL        A
000400                  ASL        A
```



```
000401                  STA         SCT.DFN
000402                  INC         DFN.PTRX            ; ADVANCE SYS CALL DFN PTR
000403                  REP         60
000404  *
000405  *  PARAMETER PASSED BY VALUE
000406  *
000407                  REP         60
000408  VALUE           BIT         SCT.DFN
000409                  BVS         REFERENCE
000410                  BMI         VAL.OUT
000411  *
000412  *  INPUT BY VALUE
000413  *
000414                  LDY         SCPTRX              ; MOVE BYTES TO ZPAGE
000415  VAL.IN          LDA         (SCPTR),Y
000416                  STA         0,X
000417                  INY
000418                  INX
000419                  DEC         SCT.DCNT
000420                  BPL         VAL.IN
000421                  STY         SCPTRX
000422                  JMP         ENDLOOP1
000423  *
000424  *  OUTPUT BY VALUE
000425  *
000426  VAL.OUT         CLC                             ; BUILD PTR TO PARM ON ZPAGE
000427                  LDA         SCPTR
000428                  ADC         SCPTRX
000429                  STA         0,X
000430                  INX
000431                  LDA         SCPTR+1
000432                  ADC         #0
000433                  STA         0,X
000434  *
000435                  CLC                             ; ADVANCE INDEX TO NEXT PARM
000436                  LDA         SCPTRX
000437                  ADC         SCT.DCNT
000438                  STA         SCPTRX
000439  *
000440                  LDA         SXPAGE+SCPTR+1      ; INCLUDE X BYTE
000441                  STA         SXPAGE,X
000442                  JMP         ENDLOOP2
000443                  REP         60
000444  *
000445  *  PARAMETER PASSED BY REFERENCE
000446  *
000447                  REP         60
000448  REFERENCE       BPL         REF1
000449  *
000450  * "LIST" PTR FOUND, CHK IF "LENGTH" PARM = 0
000451  *
000452                  LDY         SCPTRX
000453                  INY
000454                  INY
000455                  LDA         (SCPTR),Y
000456                  BEQ         ENDLOOP0            ; "LENGTH" PARM=0, SKIP "LIST" PARM
000457  *
000458  REF1            LDY         SCPTRX              ; MOVE PTR TO ZPAGE
000459                  JSR         POINTER
000460                  BCS         PARM.ERR            ; ERROR EXIT
000461  *
000462  * ADVANCE SYSTEM ZPAGE POINTER (X), CALLER'S PARM PTR.
000463  * DECREMENT PARM CTR AND CHECK IF LAST PARM PROCESSED.
000464  *
000465  ENDLOOP0        INX
000466                  INC         SCPTRX
000467  ENDLOOP2        INX
000468                  INC         SCPTRX
000469  ENDLOOP1        DEC         PARM.CNT
000470                  BEQ         PARM.EXIT
000471                  BMI         PARM.EXIT           ;SPECIAL FOR 'COLD START'
000472                  JMP         PARMLOOP
000473  *
000474  *  END OF PARAMETER PROCESSING LOOP
000475  *
000476                  REP         60
000477  *
000478  PARM.EXIT       CLC                             ; NO ERRORS
000479  PARM.ERR        RTS                             ; RETURN TO SYS CALL MANAGER
000480                  PAGE
000481                  REP         60
```

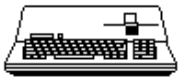


```
000482  *
000483  * POINTER
000484  *
000485  * INPUT:   SRC ADR   (SCPTR),Y & (SCPTR),Y+1
000486  *          DEST ADR  (X)
000487  *
000488  * OUTPUT:  SCPTR     UNCHANGED
000489  *          X REG       "
000490  *          A,Y REGS  FLATTENED
000491  *
000492  * ERROR:   CARRY SET (SYSERR)
000493  *
000494  * POINTER.  RETRIEVES THE CALLER'S POINTER PARAMETER IN
000495  * (SCPTR),Y, PERFORMS ADDRESS COMPENSATION, IF NECESSARY
000496  * AND PLACES THE RESULTING POINTER AT X, X+1 AND SXPAGE+1,X.
000497  *
000498                  REP        60
000499  *
000500  POINTER         EQU        *
000501                  LDA        (SCPTR),Y
000502                  PHA
000503                  INY
000504                  LDA        (SCPTR),Y
000505                  BEQ        INDIRECT
000506  *
000507                  STA        1,X                 ; DIRECT POINTER
000508                  PLA
000509                  STA        0,X
000510                  LDY        #0
000511                  BEQ        PTR010
000512  *
000513  INDIRECT        PLA                            ; INDIRECT POINTER
000514                  TAY
000515                  LDA        CZPAGE,Y
000516                  STA        0,X
000517                  LDA        CZPAGE+1,Y
000518                  STA        1,X
000519                  LDA        CXPAGE+1,Y
000520                  TAY
000521  *
000522  PTR010          LDA        1,X
000523  *
000524  * CHECK BOUNDS OF CALLER'S POINTER PARAMETER
000525  *
000526                  CPY        #$8F
000527                  BCC        PTR.X808E
000528                  BEQ        PTR.X8F
000529                  BCS        PTR.ERR1            ; ERROR, INVALID X BYTE
000530  PTR.X8F         CMP        #<ADR.LOW
000531                  BCC        PTR.ERR
000532                  CMP        #<ADR.HIGH
000533                  BCS        PTR.ERR
000534                  BCC        PTR.EXIT
000535  *
000536  * X BYTE = 80..8E
000537  *
000538  PTR.X808E       CPY        #$80
000539                  BCC        PTR.X0
000540                  CMP        #0
000541                  BEQ        PTR.ERR
000542                  CMP        #$FF
000543                  BNE        PATCH
000544                  INY                            ; $8N:FFXX --> $8N+1:7FXX
000545                  LDA        #$7F
000546                  BNE        PTR.EXIT
000547  *
000548  * X BYTE = 0
000549  *
000550  PTR.X0          CPY        #0
000551                  BNE        PTR.ERR1
000552                  CMP        #<ADR.LOW
000553                  BCC        PTR.ERR
000554                  CMP        #<ADR.HIGH
000555                  BCS        PTR.ERR
000556                  CMP        #<ADR.MID
000557                  BCS        PTR.EXIT
000558  *
000559                  PHA
000560                  LDA        B.SAVE
000561                  AND        #$0F
000562                  BNE        PTR030
```

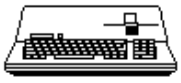


```
000563                  PLA                             ; $B=0:2000..9FFF --> $8F:2000.9FFF
000564                  LDY         #$8F
000565                  BNE         PTR.EXIT
000566  *
000567  PTR030          ORA         #$80                ; $B<>0:2000..9FFF --> $8B:0000..7FFF
000568                  TAY
000569                  PLA
000570                  SEC
000571                  SBC         #$20
000572                  BNE         PATCH
000573                  DEY                             ; $8B:00XX --> $8B-1:80XX
000574                  LDA         #$80
000575  *
000576  PATCH           CPY         #$80                ; KLUDGE FOR BFM:  $8N:01XX --> $8N-1:81XX
000577                  BCC         PTR.EXIT
000578                  CMP         #1
000579                  BNE         PTR.EXIT
000580                  CPY         #$80
000581                  BEQ         PTR.ERR             ; ERROR, $80:01XX NOT ALLOWED
000582                  DEY
000583                  LDA         #$81
000584  *
000585  PTR.EXIT        STA         1,X
000586                  TYA
000587                  STA         SXPAGE+1,X
000588                  CLC
000589                  RTS
000590  *
000591  *
000592  PTR.ERR         LDA         #>BADSCBNDS
000593                  JSR         SYSERR
000594  PTR.ERR1        LDA         #>BADXBYTE
000595                  JSR         SYSERR
000596  *
000597                  LST         ON
000598  ZZEND           EQU         *
000599  ZZLEN           EQU         ZZEND-ZZORG
000600                  IFNE        ZZLEN-LENSCMGR
000601                  FAIL        2,"SOSORG           FILE IS INCORRECT FOR SCMGR"
000602                  FIN
000603
000604  ********************************************************************************
000605  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SCMGR.SRC
000606  ********************************************************************************
000607
```

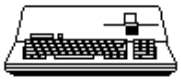
End of File -- Lines: 607 Characters: 15630



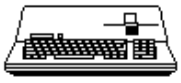
```
================================================================================
FILE: "SOS.SOS.BLOAD.TEXT"
================================================================================

000001  ****************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.BLOAD
000003  ****************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  MON I
000007  CALL-151
000008  1600:0
000009  1601<1600.93FEM
000010  3D0G
000011  MON I
000012  BLOAD SOSLDR.ABS,A$2000
000013  BLOAD INIT.ABS,A$2AF8
000014  BLOAD SYSGLOB.ABS,A$2CF8
000015  BLOAD BFM.INIT2.ABS,A$2E00
000016  BLOAD BFM.ABS,A$3200
000017  BLOAD OPRMSG.ABS,A$5466
000018  BLOAD IPL.ABS,A$55C0
000019  BLOAD UMGR.ABS,A$5A8B
000020  BLOAD DISK3.ABS,A$5E99
000021  BLOAD SYSERR.ABS,A$6404
000022  BLOAD DEVMGR.ABS,A$64D9
000023  BLOAD SCMGR.ABS,A$665E
000024  BLOAD FMGR.ABS,A$68F4
000025  BLOAD CFMGR.ABS,A$6955
000026  BLOAD BUFMGR.ABS,A$6B52
000027  BLOAD MEMMGR.ABS,A$6E6E
000028  NOMON I
000029
000030  ****************************************************************************
000031  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.BLOAD
000032  ****************************************************************************
000033

End of File -- Lines: 33 Characters: 864
```



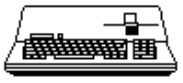
```
================================================================================
FILE: "SOS.SOS.LINK.TEXT"
================================================================================

000001  ****************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.LINK
000003  ****************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  SOSLDR.OBJ
000007  INIT.OBJ
000008  SYSGLOB.OBJ
000009  BFM.INIT2.OBJ
000010  BFM.OBJ
000011  OPRMSG.OBJ
000012  IPL.OBJ
000013  UMGR.OBJ
000014  DISK3.OBJ
000015  SYSERR.OBJ
000016  SCMGR.OBJ
000017  FMGR.OBJ
000018  CFMGR.OBJ
000019  DEVMGR.OBJ
000020  BUFMGR.OBJ
000021  MEMMGR.OBJ
000022  END
000023
000024  ****************************************************************************
000025  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.LINK
000026  ****************************************************************************

End of File -- Lines: 26 Characters: 607
```



```
================================================================================
FILE: "SOS.SOS.RENAME.TEXT"
================================================================================

000001  ****************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.RENAME
000003  ****************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  MON I
000007  RENAME SOSLDR.OBJ.ABS,SOSLDR.ABS
000008  RENAME INIT.OBJ.ABS,INIT.ABS
000009  RENAME SYSGLOB.OBJ.ABS,SYSGLOB.ABS
000010  RENAME OPRMSG.OBJ.ABS,OPRMSG.ABS
000011  RENAME BFM.OBJ.ABS,BFM.ABS
000012  RENAME BFM.INIT2.OBJ.ABS,BFM.INIT2.ABS
000013  RENAME IPL.OBJ.ABS,IPL.ABS
000014  RENAME UMGR.OBJ.ABS,UMGR.ABS
000015  RENAME DISK3.OBJ.ABS,DISK3.ABS
000016  RENAME SYSERR.OBJ.ABS,SYSERR.ABS
000017  RENAME SCMGR.OBJ.ABS,SCMGR.ABS
000018  RENAME FMGR.OBJ.ABS,FMGR.ABS
000019  RENAME CFMGR.OBJ.ABS,CFMGR.ABS
000020  RENAME DEVMGR.OBJ.ABS,DEVMGR.ABS
000021  RENAME BUFMGR.OBJ.ABS,BUFMGR.ABS
000022  RENAME MEMMGR.OBJ.ABS,MEMMGR.ABS
000023  NOMON I
000024
000025  ****************************************************************************
000026  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.RENAME
000027  ****************************************************************************

End of File -- Lines: 27 Characters: 961
```



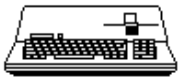
```
================================================================================================
FILE: "SOS.SOSLDR.A.SRC.TEXT"
================================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.A.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP        110
000008  *
000009  *   $1E00 +----------------+
000010  *         !    SOSLDR     !<-ENTRY        SOS MEMORY MAP
000011  *   $1FFF +----------------+  -----       (128K APPLE ///)
000012  *
000013  *              BANK 0               BANK 1               BANK 2
000014  *   $2000 +----------------+    +----------------+    +----------------+
000015  *         !                !    !                !    !                !
000016  *         !                !    !                !    !    SOSLDR      !
000017  *         !                !    !                !    !      &         !
000018  *         !                !    !                !    !  INIT MODULE   !
000019  *         !                !    !                !    !                !
000020  *         !                !    !                !    ! - - - - - - - !
000021  *         !                !    !                !    !    GLOBALS     !
000022  *         !                !    !                !    ! - - - - - - - !
000023  *         !                !    !                !    !                !
000024  *         !                !    !                !    !                !
000025  *         !                !    !                !    !                !          !
000026  *         !                !    !                !    !                !          !
000027  *         !                !    !                !    !                !          !
000028  *         !                !    !                !    !                !          !
000029  *         !                !    !                !    !                !          !
000030  *         !                !    !                !    !                !          !
000031  *         !                !    !                !    !                !  KERNEL  !
000032  *         !                !    !                !    !                !          !
000033  *         !                !    !                !    !                !          !
000034  *         !                !    !                !    !                !          !
000035  *         !                !    !                !    !                !          !
000036  *         !                !    !                !    !                !          !
000037  *         !                !    !                !    !                !          !
000038  *         !                !    !                !    !                !          !
000039  *         !                !    !                !    !                !          !
000040  *         !                !    !                !    !                !          !
000041  *         !                !    !                !    !                !-- EOF --!
000042  *         !                !    !                !    !                !          !
000043  *         !                !    !                !    !                !          !
000044  *         !                !    !                !    !                !          !
000045  *         !                !    !                !    !                !          !
000046  *         !                !    !                !    !                !          !
000047  *   $9FFF +----------------+    +----------------+    +----------------+
000048  *
000049  *
000050  *   $A000 +----------------+
000051  *     .   !    SOSBOOT     !
000052  *     .   +----------------+
000053  *
000054  *
000055  *  FIGURE 1.  SOS KERNEL FILE READ INTO $2:1E00..9FFF BY SOS BOOT IN BLOCKS 0,1.
000056  *             SOS LOADER BEGINS EXECUTION AT THIS POINT.
000057  *
000058  *
000059  *
000060  *
000061                  REP        110
000062                  PAGE
000063                  REP        110
000064  *
000065  *   $1E00 +----------------+
000066  *         !    SOSLDR     !          SOS MEMORY MAP
000067  *   $1FFF +----------------+          (128K APPLE ///)
000068  *
000069  *              BANK 0               BANK 1               BANK 2
000070  *   $2000 +----------------+    +----------------+    +----------------+
000071  *         !                !    !                !    !                !          !
000072  *         !    SOSLDR      !    !                !    !                !
000073  *         !      &         !    !                !    !                !
000074  *         !  INIT MODULE   !    !                !    !                !
000075  *         !                !    !                !    !                !          !
000076  *  LDREND ! - - - - - - - - !    !                !    !                !
```



```
000077  *         !  FILE BUFFER  !    !               !    !               !
000078  *         ! - - - - - - - !    !               !    !               !
000079  *         !               !    !               !    !               !         !
000080  *         !               !    !               !    !               !         !
000081  *         !               !    !               !    !               !         !
000082  *         !               !    !               !    !               !         !
000083  *         !               !    !               !    !               !         !
000084  *         !               !    !               !    !               !         !
000085  *         !  INTERPRETER  !    !  INTERPRETER  !    !               !
000086  *         !     FILE      !    !     FILE      !    !               !
000087  *         !               !    !               !    !               !         !
000088  *         !               !    !               !    !               !         !
000089  *         !               !    !               !    !               !         !
000090  *         !               !    !               !    !               !         !
000091  *         !               !    !               !    !               !         !
000092  *         !               !    !               !    !               !         !
000093  *         !               !    !               !    !               !         !
000094  *         !               !    !               !    !               !         !
000095  *         !               !    !               !    !               !         !
000096  *         !               !    !               !    !               !         !
000097  *         !               !    !               !    !               !         !
000098  *         !               !    !               !    !               !         !
000099  *         !               !    !               !    !               !         !
000100  *         !               !    !               !    !               !         !
000101  *         !               !    !               !    !               !         !
000102  *         !               !    !- - - EOF - - -!    !               !
000103  *   $9FFF +---------------+    +---------------+    +---------------+
000104  *
000105  *
000106  *
000107  *
000108  *  FIGURE 2.  SOS INTERPRETER FILE READ INTO BANKS 0 AND 1
000109  *             USING EXTENDED ADDRESSING (X=$80).
000110  *
000111  *
000112  *
000113  *
000114  *
000115                  REP        110
000116
000117                  CHN        SOSLDR.B.SRC
000118
000119  ************************************************************************
000120  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.A.SRC
000121  ************************************************************************
000122
000123
```

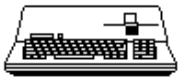
End of File -- Lines: 123 Characters: 4326

```
================================================================================
FILE: "SOS.SOSLDR.B.SRC.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.B.SRC
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  .PAGE
000007
000008  ;*************************************************************************
000009  ;
000010  ;   $1E00 +----------------+
000011  ;         !    SOSLDR      !       SOS MEMORY MAP
000012  ;   $1FFF +----------------+       (128K APPLE ///)
000013  ;
000014  ;              BANK 0                BANK 1               BANK 2
000015  ;   $2000 +----------------+    +----------------+    +----------------+
000016  ;         !                !    !                !    !                !
000017  ;         !    SOSLDR      !    !                !    !                !
000018  ;         !      &         !    !                !    !                !
000019  ;         !  INIT MODULE   !    !                !    !                !
000020  ;         !                !    !                !    !                !
000021  ;         ! - - - - - - - -!    !                !    !                !
000022  ;         !  FILE BUFFER   !    !                !    !                !
000023  ;         ! - - - - - - - -!    !                !    !                !
000024  ;         !                !    !                !    !                !
000025  ;         !                !    !                !    !                !              !
000026  ;         !                !    !                !    !                !              !
000027  ;         !                !    !                !    !                !              !
000028  ;         !                !    !                !    !                !              !
000029  ;         !                !    !                !    !                !              !
000030  ;         !                !    !                !    !                !              !
000031  ;         !                !    !                !    !                !              !
000032  ;         !                !    !                !    !                !              !
000033  ;         !                !    !                !    !                !              !
000034  ;         !                !    !                !    !                !              !
000035  ;         !                !    !                !    !                !              !
000036  ;         !                !    !                !    !                ! - - - - - - !
000037  ;         !    DRIVER      !    !    DRIVER      !    !                !
000038  ;         !     FILE       !    !     FILE       !    !                !
000039  ;         !                !    !                !    !                !              !
000040  ;         !                !    !                !    !                !              !
000041  ;         !                !    !                !    !                ! INTERPRETER !
000042  ;         !                !    !                !    !                !     CODE     !
000043  ;         !                !    !                !    !                !              !
000044  ;         !                !    !                !    !                !              !
000045  ;         !                !    !                !    !                !              !
000046  ;         !                !    !                !    !                !              !
000047  ;         !                !    !- - - EOF - - -!    !                !
000048  ;   $9FFF +----------------+    +----------------+    +----------------+
000049  ;
000050  ;
000051  ;
000052  ;
000053  ;                FIGURE 3.  SOS DRIVER FILE READ INTO BANKS 0 AND 1
000054  ;                           USING EXTENDED ADDRESSING (X=$80).
000055  ;
000056  ;
000057  ;
000058  ;
000059  ;*************************************************************************
000060
000061                  .PAGE
000062
000063  ;         !                !    !                !    !        !               !
000064  ;   $9FFF +----------------+    +----------------+    !        +---------------+
000065  ;                                                     !
000066  ;                                                     !
000067  ;                                                     !   (SYSTEM DEVICE TABLE)
000068  ;                                                     !
000069  ;  FIGURE 4. SOS LOADER FINISHED.  JUMP TO          DIB   ADR   BANK  UNIT
000070  ;             FIRST BYTE OF INTERPRETER'S CODE.    !-----!-----!-----!-----!
000071  ;                                                  !     !     !     !     !
000072  ;                                                  !     !     !     !     !
000073  ;                                                  !     !     !     !     !
000074  ;                                                  !     !     !     !     !
000075  ;                                                  !-----!-----!-----!-----!
000076  ;
```

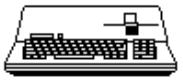


```
000077  ;
000078  ;
000079  ;*******************************************************************************
000080
000081  ********************************************************************
000082  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.B.SRC
000083  ********************************************************************
000084
000085
```

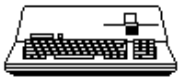
End of File -- Lines: 85 Characters: 2974



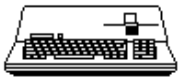
```
================================================================================
FILE: "SOS.SOSLDR.C.SRC.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.C.SRC
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP        100
000008  *
000009  * SUBROUTINES:
000010  *
000011  * SOSLDR                "MAIN PROGRAM"
000012  *
000013  *    SOSLDR1            "PROCESSES KERNEL/INTERPRETER/DRIVER FILES"
000014  *
000015  * (1)   MOVE            "MOVES SRC.P..SRC.P+CNT-1 TO DST.P..DST.P+CNT-1"
000016  *
000017  *       INIT.KRNL       "CALLS KERNEL INITIALIZATION MODULES"
000018  *
000019  *       WELCOME         "PRINTS WELCOME MESSAGE ("APPLE ///", VERSION, DATE/TIME, COPYRIGHT)
000020  *
000021  *       ADVANCE         "ADVANCES WRK.PTR TO NEXT INTERP/KERNEL MODULE.  INITS SRC.P, DST.P, CNT FOR MOVE"
000022  *
000023  *       REVERSE         "REVERSES TITLE/CODE/RELOC COUNTS TO ALLOW DRIVER FILE TO BE PROCESSED FM BACK TO FRONT"
000024  *
000025  *       DADVANCE        "ADVANCES WORK.P TO NEXT DRIVER MODULE.  INITS SRC.P, CNT, REL.P FOR MOVE"
000026  *
000027  *          DADD         "ADVANCES WORK.P TO NEXT DRIVER FIELD"
000028  *
000029  *       FLAGS           "PROCESSES "INACTIVE" & "PAGE ALIGN" FLAGS IN DRIVER MODULE'S DIBS"
000030  *
000031  *          NEXT.DIB     "ADVANCES TO NEXT DIB IN DRIVER MODULE"
000032  *
000033  *       GETMEM          "COMPUTES DESTINATION BASE ADDRESS FOR NEXT DRIVER MODULE"
000034  *
000035  *          NEWDST       "COMPUTES DESTINATION BASE ADDRESS, ALIGNING ON PAGE BOUNDARY IF REQUESTED"
000036  *
000037  *          BUILD.DSEG   "COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT"
000038  *
000039  *       RELOC           "RELOCATES DRIVER MODULE'S CODE FIELD USING RELOCATION FIELD"
000040  *
000041  * (1)   LINK            "LINKS FIRST DIB TO PREVIOUS DRIVER'S LAST "ACTIVE" DIB, AND ADDS SDT ENTRY"
000042  *
000043  *          SET.DRIVES   "INITIALIZES DIB LINKS IN KERNEL'S FLOPPY DRIVER"
000044  *
000045  * (1)      ALLOC.DEV    "ADDS A NEW ENTRY TO THE DEVICE MANAGER'S SYSTEM DEVICE TABLE (SDT)"
000046  *
000047  *       ALLOC.SEG       "ALLOCATES SEGMENTS FOR KERNEL, INTERPRETER AND SYSTEM WORK AREA"
000048  *
000049  *          RSEG         "CALLS MEMORY MANAGER TO ALLOCATE SEGMENTS FOR THE KERNEL AND INTERPRTER"
000050  *
000051  *       ALLOC.DSEG      "ALLOCATES SEGMENTS FOR DRIVER MODULES"
000052  *
000053  *    ERROR              "DISPLAYS ERROR MESSAGE, SOUNDS BELL AND LOOPS UNTIL CONTROL/RESET PRESSED"
000054  *
000055  * (1) - INDICATES THAT THE ROUTINE PERFORMS BANK SWITCHING AND MUST(!) BE OUTSIDE THE 32K RAM BANKS.
000056                  REP        100
000057                  PAGE
000058                  REP        100
000059  *
000060  * SOS.KERNEL FILE FORMAT
000061  *
000062  *  (8)   LABEL                    <---+
000063  *          = "SOS KRNL"               !
000064  *                                     !
000065  *  (2)   HEADER COUNT                 !
000066  *        HEADER                       !
000067  *          = # OF FLOPPY DRIVES       !    CONTAINED IN THIS LISTING
000068  *          = INTERPRETER PATHNAME     !
000069  *          = DRIVER PATHNAME          !
000070  *                                     !
000071  *  (4)   ADR & COUNT                  !
000072  *        SOSLDR CODE              <---+
000073  *
000074  *  (4)   ADR & COUNT
000075  *        GLOBALS
000076  *
```



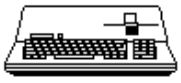
```
000077  *  (4)   ADR & COUNT
000078  *        KERNEL CODE
000079  *
000080                  REP        100
000081  *
000082  * SOS.INTERP FILE FORMAT
000083  *
000084  *  (8)   LABEL
000085  *           = "SOS NTRP"
000086  *
000087  *  (2)   HEADER COUNT
000088  *
000089  *  (4)   ADR & COUNT
000090  *        INTERPRETER CODE
000091  *
000092                  REP        100
000093  *
000094  * SOS.DRIVER FILE FORMAT
000095  *
000096  *  (8)   LABEL
000097  *           = "SOS DRVR"
000098  *
000099  *  (2)   HEADER COUNT
000100  *           = # OF FLOPPY DRIVES
000101  *           = CHARACTER SET TABLE
000102  *           = KEYBOARD TABLE
000103  *        ...
000104  *                                                          +---------------------------------------+
000105  *  (2)   DM #N TITLE COUNT        <---+                     !       RELOCATION FIELD FORMAT         !
000106  *              TITLE FIELD            !                     !       ----------------------          !
000107  *  (2)   DM #N CODE  COUNT            !    DRIVER MODULE #N     ! CONSISTS OF A LIST OF 2 BYTE POINTERS !
000108  *              CODE  FIELD            !                     ! WHICH POINT TO THE LOW BYTE OF A TWO  !
000109  *  (2)   DM #N RELOC COUNT            !                     ! BYTE QUANTITY TO BE RELOCATED.        !
000110  *              RELOC FIELD        <---+                     +---------------------------------------+
000111  *        ...
000112  *
000113  *        $FFFF = THE END
000114  *
000115                  REP        100
000116                  PAGE
000117                  REP        100
000118  *
000119  * SOSLDR - EXTERNAL DECLARATIONS
000120  *
000121                  REP        100
000122                  EXTRN      SYSBANK
000123                  EXTRN      MEMSIZE
000124                  EXTRN      SCRNMODE
000125                  EXTRN      SOSVER
000126                  EXTRN      SOSVERL
000127  *
000128                  EXTRN      INT.INIT            ; (IPL) INTERRUPT INIT
000129                  EXTRN      EVQ.INIT            ; (IPL) EVENT QUEUE INIT
000130                  EXTRN      DMGR.INIT           ; DEVICE MANAGER INIT
000131                  EXTRN      MAX.DNUM            ;        "
000132                  EXTRN      SDT.SIZE
000133                  EXTRN      SDT.DIBL
000134                  EXTRN      SDT.DIBH
000135                  EXTRN      SDT.ADRL
000136                  EXTRN      SDT.ADRH
000137                  EXTRN      SDT.BANK
000138                  EXTRN      SDT.UNIT
000139                  EXTRN      BLKD.SIZE
000140                  EXTRN      BLKDLST
000141                  EXTRN      CFMGR.INIT          ; CHAR FILE MANAGER INIT
000142                  EXTRN      MMGR.INIT           ; MEMORY MANAGER INIT
000143                  EXTRN      BMGR.INIT           ; BUFFER FILE MANAGER INIT
000144                  EXTRN      BFM.INIT            ; BLOCK FILE MANAGER INIT
000145                  EXTRN      BFM.INIT2           ; BLOCK FILE MANAGER INIT2
000146                  EXTRN      CLK.INIT            ; CLOCK SYSTEM CALL INIT
000147  *
000148                  EXTRN      DIB1                ; ON BOARD DISK DRIVER'S DIBS (1-4)
000149                  EXTRN      DIB2
000150                  EXTRN      DIB3
000151                  EXTRN      DIB4
000152  *
000153  *ENTRY I.BASE.P ; USED BY BFM.INIT2  (HARDWIRED!)
000154                  PAGE
000155                  REP        100
000156  *
000157  * FILE DATA DECLARATIONS
```



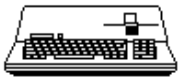
```
000158  *
000159                  REP         100
000160  * KERNEL FILE
000161                  REP         100
000162  K.FILE          ASC         "SOS KRNL"
000163  K.HDR.CNT       DW          LDR.ADR-K.DRIVES
000164  K.DRIVES        DFB         $1
000165  K.FLAGS         DFB         $0                  ; RESERVED FOR FUTURE USE
000166  I.PATH          DFB         $E
000167                  ASC         ".D1/SOS.INTERP"
000168                  DS          $30-$F
000169  D.PATH          DFB         $E
000170                  ASC         ".D1/SOS.DRIVER"
000171                  DS          $30-$F
000172  LDR.ADR         DW          $0
000173  LDR.CNT         DW          ZZEND-SOSLDR
000174                  REP         100
000175  * INTERPRETER/DRIVER FILES    <--+
000176  * ERROR MESSAGES                 !    DEFINED IN BACK OF THIS LISTING
000177  * WELCOME MESSAGES            <--+
000178                  REP         100
000179                  PAGE
000180                  REP         100
000181  *
000182  * SOSLDR - DATA DECLARATIONS (1)
000183  *
000184                  REP         100
000185  TRUE            EQU         $80
000186  FALSE           EQU         $0
000187  *
000188  Z.REG           EQU         $FFD0
000189  E.REG           EQU         $FFDF
000190  B.REG           EQU         $FFEF
000191  *
000192  CZPAGE          EQU         $1A00
000193  CSPAGE          EQU         $1B00
000194  CXPAGE          EQU         $1600
000195  SZPAGE          EQU         $1800
000196  SXPAGE          EQU         $1400
000197  SSPAGE          EQU         $0100
000198  *
000199  ROM.ADR         EQU         $F1B9
000200  ROM.ID          EQU         $A0
000201                  PAGE
000202                  REP         100
000203  *
000204  * SOSLDR - DATA DECLARATIONS (2)
000205  *
000206                  REP         100
000207  ZPAGE           EQU         $00
000208  *
000209  K.BASE          EQU         ZPAGE+$0            ; SOSLDR1 SUBROUTINE    +--------------------------------------+
000210  I.BASE.P        EQU         ZPAGE+$2            ;                       ! <VARNAME>.P ::= 3 BYTE ZPAGE POINTER !
000211  RDBUF.P         EQU         ZPAGE+$4            ;                       +--------------------------------------+
000212  SYSBUF.P        EQU         ZPAGE+$6
000213  TEMP.BANK       EQU         ZPAGE+$8
000214  TEMP.ADRH       EQU         ZPAGE+$9
000215  WORK.P          EQU         ZPAGE+$A
000216  *
000217  REV.SAVE        EQU         ZPAGE+$C            ; REVERSE SUBROUTINE
000218  *
000219  FIRST.ADIB      EQU         ZPAGE+$10           ; FLAGS SUBROUTINE
000220  PREV.ADIB.P     EQU         ZPAGE+$12
000221  DIB.P           EQU         ZPAGE+$14
000222  PG.ALIGN        EQU         ZPAGE+$16
000223  DIB.FLAGS       EQU         $14
000224  DIB.DCB         EQU         $20
000225  *
000226  PREVBANK        EQU         ZPAGE+$18           ; GETMEM SUBROUTINE
000227  PREVDST         EQU         ZPAGE+$19
000228  *
000229  CODE.P          EQU         ZPAGE+$1C           ; RELOCATION SUBROUTINE
000230  REL.P           EQU         ZPAGE+$1E
000231  REL.END         EQU         ZPAGE+$20
000232  *
000233  SRC.P           EQU         ZPAGE+$22           ; MOVE SUBROUTINE
000234  DST.P           EQU         ZPAGE+$24
000235  CNT             EQU         ZPAGE+$26
000236  *
000237  DSTBANK         EQU         ZPAGE+$2A           ; LINK SUBROUTINE
000238  LINK.P          EQU         ZPAGE+$2C
```



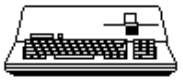
```
000239  *
000240  DIB.ENTRY       EQU         2                   ; ALLOC.DEV SUBROUTINE
000241  DIB.UNIT        EQU         4+16+2
000242  DIB.DTYPE       EQU         4+16+3
000243  *
000244  ETEMP           EQU         ZPAGE+$2E           ; ERROR SUBROUTINE
000245  *
000246  WTEMP           EQU         ZPAGE+$2F           ; WELCOME SUBROUTINE
000247
000248  ********************************************************************************
000249  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.C.SRC
000250  ********************************************************************************
000251
000252

End of File -- Lines: 252 Characters: 7483
```



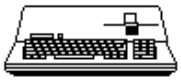
```
================================================================================================
FILE: "SOS.SOSLDR.D.SRC.TEXT"
================================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.D.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP         100
000008  *
000009  * SOS LOADER -
000010  *
000011  * (MAIN PROGRAM)
000012                  REP         100
000013  SOSLDR          EQU         *                   ;                                   +---------------+
000014                  LDA         #0                  ; ZERO SOS/USER X, Z AND STACK PAGES  ! SEE FIGURE 1. !
000015                  TAX                             ;                                   +---------------+
000016  SLDR010         STA         CZPAGE,X
000017                  STA         CXPAGE,X
000018                  STA         CSPAGE,X
000019                  STA         SZPAGE,X
000020                  STA         SXPAGE,X
000021                  STA         SSPAGE,X
000022                  DEX
000023                  BNE         SLDR010
000024  *                                   ; SETUP SOS CALL ENVIRONMENT (WRITE PROTECT=OFF)
000025                  LDA         #$30                ; E:=( 0.0.1.1:0.0.0.0 )
000026                  STA         E.REG               ;    ( 1.I.S.R:W.P.R.R )
000027  *
000028                  LDX         #$FB                ; CONSOLE 1.0 MODIFIES STACK DURING D.INIT CALL
000029                  TXS
000030                  LDA         #<CZPAGE            ; ZREG:=CALLER'S Z PAGE
000031                  STA         Z.REG
000032  *                                   ; +--------------------------------+
000033                  JSR         SOSLDR1             ; ! PROCESS KRNL/INTERP/DRVR FILES !
000034  *                                   ; +--------------------------------+
000035                  LDA         E.REG
000036                  AND         #$10                ; SETUP SOS CALL ENVIRONMENT (WRITE PROTECT=ON)
000037                  ORA         #$28                ; E:=( 0.0.1.X:1.0.0.0 )
000038                  STA         E.REG               ;    ( 1.I.S.R:W.P.R.R )
000039  *
000040                  LDX         #$FF                ; STACK.REG:=$FF
000041                  TXS
000042                  LDA         #<CZPAGE            ; ZREG:=CALLER'S Z PAGE
000043                  STA         Z.REG
000044  *                                                                   +---------------+
000045                  LDA         SYSBANK             ; BREG:=SYSBANK             ! SEE FIGURE 4. !
000046                  STA         B.REG               ;                           +---------------+
000047                  JMP         (I.BASE.P)          ; SOS LOAD COMPLETE - JMP TO INTERPRETER
000048  *
000049  *THE END.
000050                  REP         100
000051                  PAGE
000052                  REP         100
000053  *
000054  * MOVE ( IN:   SRC.P
000055  *        IN:   DST.P
000056  *        IN:   A="BANK"
000057  *        IN:   CNT      )
000058  *
000059  *        LOCAL:  END
000060  * (MOVES SRC.P..SRC.P+CNT-1 TO DST.P..DST.P+CNT-1)              "CNT PARM IS DESTROYED"
000061                  REP         100
000062  MOVE            EQU         *
000063                  TAX
000064                  LDA         B.REG               ; SAVE BANK REGISTER
000065                  PHA
000066                  STX         B.REG               ; BREG:=A
000067                  LDA         CNT+1               ; IF CNT <> 0
000068                  ORA         CNT                 ;    THEN
000069                  BEQ         MOVE.EXIT
000070                  LDA         CNT                 ;       CNT:=CNT-1
000071                  BNE         MOVE010
000072                  DEC         CNT+1
000073  MOVE010         DEC         CNT
000074                  CLC                             ;       SRC.P:=SRC.P+PAGE.CNT
000075                  LDA         SRC.P+1
000076                  ADC         CNT+1
```



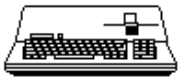
```
000077                  STA         SRC.P+1
000078                  LDA         DST.P+1             ;       DST.P:=DST.P+PAGE.CNT
000079                  ADC         CNT+1
000080                  STA         DST.P+1
000081                  INC         CNT+1               ;       PAGE.CNT:=PAGE.CNT+1
000082                  LDY         CNT                 ;       Y:=BYTE.CNT
000083                  BEQ         MOVE020             ;       IF Y=0 THEN M2
000084  *
000085  MOVE.PAGE       LDA         (SRC.P),Y           ;M1:    DO
000086                  STA         (DST.P),Y           ;          (DST.P),Y:=(SRC.P),Y
000087                  DEY                             ;          Y:=Y-1
000088                  BNE         MOVE.PAGE           ;       UNTIL  Y=0
000089  MOVE020         LDA         (SRC.P),Y           ;M2:    (DST.P),Y:=(SRC.P),Y
000090                  STA         (DST.P),Y
000091                  DEY                             ;       Y:=Y-1
000092                  DEC         SRC.P+1             ;       SRC.P:=SRC.P-256
000093                  DEC         DST.P+1             ;       DST.P:=DST.P-256
000094                  DEC         CNT+1               ;       PAGE.CNT:=PAGE.CNT-1
000095                  BNE         MOVE.PAGE           ;       IF PAGE.CNT <> 0 THEN M1
000096  *
000097                  INC         SRC.P+1             ; RESTORE SRC.P
000098                  INC         DST.P+1             ;    "    DST.P
000099  *
000100  MOVE.EXIT       PLA                             ; RESTORE BANK REGISTER
000101                  STA         B.REG
000102                  RTS
000103                  PAGE
000104                  REP         100
000105  *
000106  * LINK ( IN:   DST.P
000107  *        IN:   DSTBANK
000108  *        IN:   PREVBANK
000109  *        IN:   FIRST.ADIB
000110  *        I/O:  SDT.TBL
000111  *        I/O:  BLKDLST
000112  *        OUT:  LINKED DRIVER MODULE )
000113  *
000114  *        OWN:  LINK.P
000115  * (LINKS FIRST DIB TO PREVIOUS DRIVER'S LAST "ACTIVE" DIB, AND ADDS SDT ENTRY)
000116                  REP         100
000117  LINK            EQU         *
000118                  CLC                             ; FIRST.ADIB:=0:DST.P+FIRST.ADIB
000119                  LDA         DST.P
000120                  ADC         FIRST.ADIB
000121                  STA         FIRST.ADIB
000122                  LDA         DST.P+1
000123                  ADC         FIRST.ADIB+1
000124                  STA         FIRST.ADIB+1
000125                  LDA         #0
000126                  STA         CXPAGE+FIRST.ADIB+1
000127                  LDA         PREVBANK            ; BREG:=PREVBANK
000128                  STA         B.REG
000129                  LDY         #0                  ; (LINK.P):=FIRST.ADIB
000130                  LDA         FIRST.ADIB
000131                  STA         (LINK.P),Y
000132                  INY
000133                  LDA         FIRST.ADIB+1
000134                  STA         (LINK.P),Y
000135                  LDA         DSTBANK             ; BREG:=DSTBANK
000136                  STA         B.REG
000137                  LDA         FIRST.ADIB          ; LINK.P:=FIRST.ADIB
000138                  STA         LINK.P
000139                  LDA         FIRST.ADIB+1
000140                  STA         LINK.P+1
000141  WALKLINKS       JSR         ALLOC.DEV           ; ALLOC.DEV(LINK.P BREG.IN, SDT.TBL BLKDLST.IO)
000142  LINK010         LDY         #0                  ; WHILE (LINK.P) <> 0 AND (LINK.P) <> LINK.P
000143                  LDA         (LINK.P),Y
000144                  INY
000145                  ORA         (LINK.P),Y
000146                  BEQ         LINK100
000147                  LDA         (LINK.P),Y
000148                  CMP         LINK.P+1
000149                  BNE         LINK030
000150                  DEY
000151                  LDA         (LINK.P),Y
000152                  CMP         LINK.P
000153                  BEQ         LINK100
000154  LINK030         LDY         #0                  ;    DO  LINK.P:=(LINK.P)
000155                  LDA         (LINK.P),Y
000156                  TAX
000157                  INY
```



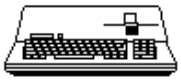
```
000158                  LDA         (LINK.P),Y
000159                  STX         LINK.P
000160                  STA         LINK.P+1
000161                  JSR         ALLOC.DEV           ;     "  ALLOC.DEV(LINK.P BREG.IN, SDT.TBL BLKDLST.IO)
000162                  JMP         LINK010
000163  *
000164  LINK100         LDY         #0                  ; (LINK.P):=0
000165                  TYA
000166                  STA         (LINK.P),Y
000167                  INY
000168                  STA         (LINK.P),Y
000169                  DEY                             ; BREG:=0
000170                  STY         B.REG
000171                  RTS
000172  *
000173  *
000174  *
000175  *
000176  * LINK.INIT ( IN:   A=# DRIVES
000177  *             IN:   DIB1..4
000178  *             I/O:  SDT.TBL
000179  *             I/O:  BLKDLST    )
000180  *
000181  LINK.INIT       EQU         *
000182                  JSR         SET.DRIVES          ; SET.DRIVES(A=#DRIVES.IN, DIB1..4.IN)
000183                  LDA         #0
000184                  STA         MAX.DNUM            ; MAXDNUM:=0
000185                  STA         BLKDLST             ; BLKDLST:=0
000186                  STA         CXPAGE+LINK.P+1     ; LINK.P:=0:DIB1
000187                  LDA         #>DIB1
000188                  STA         LINK.P
000189                  LDA         #<DIB1
000190                  STA         LINK.P+1
000191                  JMP         WALKLINKS
000192                  PAGE
000193                  REP         100
000194  *
000195  * ALLOC.DEV ( IN:   LINK.P
000196  *             IN:   B.REG
000197  *             I/O:  SDT.TBL                                                 (SYSTEM DEVICE TABLE)
000198  *                      IN:   SDT.SIZE  = CONSTANT
000199  *                      IN:   DIB.ENTRY = CONSTANT                    DEV   DIB   ADR  BANK  UNIT
000200  *                      IN:   DIB.UNIT  = CONSTANT                        !-----!-----!-----!-----!
000201  *                      IN:   DIB.DTYPE = CONSTANT                     1  !     !     !     !     !
000202  *                      I/O:  MAX.DNUM                                 2  !     !     !     !     !
000203  *                      OUT:  SDT.BANK                                 .  !     !     !     !     !
000204  *                      OUT:  SDT.DIB                                  .  !     !     !     !     !
000205  *                      OUT:  SDT.ADR                                  .  !-----!-----!-----!-----!
000206  *                      OUT:  SDT.UNIT                               MAX.DNUM
000207  *             I/O:  BLKDLST
000208  *                      IN:   BLKD.SIZE = CONSTANT
000209  * (ADDS A NEW ENTRY TO THE DEVICE MANAGER'S SYSTEM DEVICE TABLE (SDT))
000210                  REP         100
000211  ALLOC.DEV       EQU         *
000212                  INC         MAX.DNUM            ; MAX.DNUM:=MAX.DNUM+1
000213                  LDX         MAX.DNUM            ; IF MAX.DNUM >= SDT.SIZE
000214                  CPX         #>SDT.SIZE          ;    THEN
000215                  BCC         ADEV010
000216                  LDX         #ERR8X              ;       ERROR("TOO MANY DEVICES")
000217                  LDY         #ERR8L
000218                  JSR         ERROR
000219  ADEV010         LDA         B.REG               ; SDT.BANK,X:=BREG
000220                  STA         SDT.BANK,X
000221                  CLC                             ; SDT.DIB,X:=LINK.P+4
000222                  LDA         LINK.P
000223                  ADC         #4
000224                  STA         SDT.DIBL,X
000225                  LDA         LINK.P+1
000226                  ADC         #0
000227                  STA         SDT.DIBH,X
000228                  SEC                             ; SDT.ADR,X:=(LINK.P),DIB.ENTRY-1
000229                  LDY         #DIB.ENTRY
000230                  LDA         (LINK.P),Y
000231                  SBC         #1
000232                  STA         SDT.ADRL,X
000233                  INY
000234                  LDA         (LINK.P),Y
000235                  SBC         #0
000236                  STA         SDT.ADRH,X
000237                  LDY         #DIB.UNIT           ; SDT.UNIT,X:=(LINK.P),DIB.UNIT
000238                  LDA         (LINK.P),Y
```



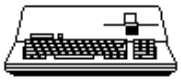
```
000239                  STA         SDT.UNIT,X
000240                  LDY         #DIB.DTYPE          ; IF (LINK.P),DIB.DTYPE = "BLOCK DEVICE"
000241                  LDA         (LINK.P),Y
000242                  BPL         ADEV.EXIT
000243                  TXA                             ;    THEN
000244                  INC         BLKDLST             ;       BLKDLST:=BLKDLST+1
000245                  LDX         BLKDLST             ;       IF BLKDLST >= BLKD.SIZE
000246                  CPX         #>BLKD.SIZE         ;          THEN
000247                  BCC         ADEV020
000248                  LDX         #ERR9X              ;             ERROR("TOO MANY BLOCK DEVICES")
000249                  LDY         #ERR9L
000250                  JSR         ERROR
000251  ADEV020         STA         BLKDLST,X           ;       BLKDLST,X:=MAX.DNUM
000252  ADEV.EXIT       RTS                             ; RETURN
000253                  PAGE
000254                  REP         100
000255  *
000256  * SOSLDR1 ()
000257  *
000258  * (PROCESSES KERNEL/INTERPRETER/DRIVER FILES)
000259                  REP         100
000260  SOSLDR1         EQU         *
000261                  LDX         #$1F                ; COPY ROM'S DISK CORE ROUTINE ZPAGE VARS TO SOS ZPAGE
000262  LDR010          LDA         $380,X
000263                  STA         SZPAGE,X
000264                  DEX
000265                  BPL         LDR010
000266                  REP         100
000267  * PROCESS KERNEL FILE
000268                  REP         100
000269  *
000270  * MOVE AND INITIALIZE SOS GLOBALS
000271  *
000272                  LDA         #>LDR.ADR           ; WORK.P:=0:LDR.ADR
000273                  STA         WORK.P
000274                  LDA         #<LDR.ADR
000275                  STA         WORK.P+1
000276                  JSR         ADVANCE             ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
000277  *
000278                  LDA         B.REG               ; MOVE(SRC.P DST.P A=BREG CNT.IN)
000279                  JSR         MOVE
000280  *
000281                  LDA         B.REG               ; SYSBANK:=BREG
000282                  AND         #$0F
000283                  STA         SYSBANK
000284                  ASL         A                   ; MEMSIZ:=SYSBANK*2+4 "16K CHUNKS"
000285                  CLC
000286                  ADC         #4
000287                  STA         MEMSIZE             ; AND, MEMSIZE (SIZE IN 16K BYTE "CHUNKS")
000288  *
000289  * MOVE KERNAL CODE
000290  *
000291                  JSR         ADVANCE             ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
000292  *
000293                  LDA         DST.P               ; K.BASE:=DST.P
000294                  STA         K.BASE
000295                  LDA         DST.P+1
000296                  STA         K.BASE+1
000297                  LDA         B.REG               ; MOVE(SRC.P DST.P A=BREG CNT.IN)
000298                  JSR         MOVE
000299  *
000300  * MOVE LOADER TO BANK 0 AND SWITCH FROM SYSTEM BANK TO BANK 0
000301  *
000302                  LDA         #>$2000             ; MOVE(SRC.P=0:2000 DST.P=8F:2000 A=BREG CNT=LDR.END-$2000)
000303                  STA         SRC.P
000304                  STA         DST.P
000305                  LDA         #<$2000
000306                  STA         SRC.P+1
000307                  STA         DST.P+1
000308                  LDA         #$8F
000309                  STA         CXPAGE+DST.P+1
000310                  LDA         #>LDREND-$2000
000311                  STA         CNT
000312                  LDA         #<LDREND-$2000
000313                  STA         CNT+1
000314                  LDA         B.REG
000315                  JSR         MOVE
000316                  LDA         #0                  ; BREG:=0
000317                  STA         B.REG
000318  *
000319  * INITIALIZE SDT TABLE, KERNEL AND PRINT WELCOME MESSAGE
```



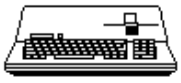
```
000320  *
000321                  LDA     K.DRIVES            ; LINK.INIT(A=K.DRIVES DIB1..4.IN, SDT.TBL BLKDLST.IO)
000322                  JSR     LINK.INIT
000323                  JSR     INIT.KRNL           ; INIT.KRNL()
000324                  JSR     WELCOME             ; WELCOME()
000325  *
000326                  LDA     E.REG               ; ENABLE ROM BANK
000327                  ORA     #$03
000328                  STA     E.REG
000329                  LDA     ROM.ADR             ; IF MONITOR ROM <> NEW
000330                  CMP     #ROM.ID             ;    THEN
000331                  BEQ     LDR020
000332                  LDX     #ERR7X              ;       ERROR("ROM ERROR:  PLEASE NOTIFY YOUR DEALER")
000333                  LDY     #ERR7L
000334                  JSR     ERROR
000335  LDR020          LDA     E.REG               ; DISABLE ROM BANK
000336                  AND     #$F6
000337                  STA     E.REG
000338                  REP     100
000339  * PROCESS INTERPRETER FILE
000340                  REP     100
000341  *
000342  * OPEN SOS INTERPRETER FILE (DEFAULT='SOS.INTERP')
000343  *
000344                  LDY     I.PATH              ; OPEN(PATHNAME:=I.PATH
000345  LDR030          LDA     I.PATH,Y            ;      REFNUM=OPEN.REF
000346                  STA     PATH,Y              ;      SYSBUF.P:=80:LDREND-2000 )
000347                  DEY
000348                  BPL     LDR030
000349  *
000350                  LDA     #>LDREND-$2000
000351                  STA     SYSBUF.P
000352                  LDA     #<LDREND-$2000
000353                  STA     SYSBUF.P+1
000354                  LDA     #$80
000355                  STA     CXPAGE+SYSBUF.P+1
000356  *
000357  *
000358                  BRK
000359                  DFB     OPEN
000360                  DW      OPEN.PARMS
000361                  BEQ     LDR040
000362                  LDX     #ERR1X              ; ERROR("INTERPRETER FILE NOT FOUND")
000363                  LDY     #ERR1L
000364                  JSR     ERROR
000365  LDR040          LDA     OPEN.REF
000366                  STA     READ.REF
000367                  STA     CLOSE.REF
000368  *
000369  * READ IN ENTIRE INTERPRETER FILE
000370  *
000371                  LDA     #$80                ; READ(REFNUM=READ.REF
000372                  STA     CXPAGE+RDBUF.P+1    ;      RDBUF.P:=80:FILE
000373                  LDA     #>FILE              ;      BYTES=$FFFF-FILE+1
000374                  STA     RDBUF.P             ;      BYTESRD=I.BYTESRD )
000375                  LDA     #<FILE
000376                  STA     RDBUF.P+1
000377  *
000378                  BRK
000379                  DFB     READ
000380                  DW      READ.PARMS
000381                  BEQ     LDR050
000382                  LDX     #ERR0X              ; ERROR("I/O ERROR")
000383                  LDY     #ERR0L
000384                  JSR     ERROR
000385  *                                                                       +---------------+
000386  * CLOSE INTERPRETER FILE AND CHECK LABEL                                ! SEE FIGURE 2. !
000387  *                                                                       +---------------+
000388  LDR050          BRK                         ; CLOSE(REFNUM=CLOSE.REF)
000389                  DFB     CLOSE
000390                  DW      CLOSE.PARMS
000391                  LDY     #7                  ; CHECK LABEL
000392  LDR051          LDA     (RDBUF.P),Y
000393                  CMP     I.LABEL,Y
000394                  BNE     LDR052
000395                  DEY
000396                  BPL     LDR051
000397                  BMI     LDR053
000398  LDR052          LDX     #ERR2X              ; ERROR("INVALID INTERPRETER FILE")
000399                  LDY     #ERR2L
000400                  JSR     ERROR
```



```
000401  *
000402  * MOVE INTERPRETER CODE
000403  *
000404  LDR053          LDA       #>I.HDR.CNT-2       ; WORK.P:=80:I.HDR.CNT-2
000405                  STA       WORK.P
000406                  LDA       #<I.HDR.CNT-2
000407                  STA       WORK.P+1
000408                  LDA       #$80
000409                  STA       CXPAGE+WORK.P+1
000410  *
000411                  JSR       ADVANCE             ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
000412  *
000413                  LDA       DST.P               ; I.BASE.P:=0:DST.P
000414                  STA       I.BASE.P
000415                  LDA       DST.P+1
000416                  STA       I.BASE.P+1
000417                  LDA       #0
000418                  STA       CXPAGE+I.BASE.P+1
000419  *
000420                  CLC                           ; IF DST.P+CNT > K.BASE THEN ERROR
000421                  LDA       CNT
000422                  ADC       DST.P
000423                  TAX
000424                  LDA       CNT+1
000425                  ADC       DST.P+1
000426                  CPX       K.BASE
000427                  SBC       K.BASE+1
000428                  BEQ       LDR070
000429                  BCC       LDR070
000430                  LDX       #ERR3X              ; ERROR("INCOMPATIBLE INTERPRETER")
000431                  LDY       #ERR3L
000432                  JSR       ERROR
000433  *
000434  LDR070          LDA       SYSBANK             ; MOVE(SRC.P=RDBUF.P DST.P A=SYSBANK CNT.IN)
000435                  JSR       MOVE
000436                  REP       100
000437  * PROCESS DRIVER FILE
000438                  REP       100
000439  *
000440  * OPEN SOS DRIVER FILE (DEFAULT='SOS.DRIVER')
000441  *
000442                  LDY       D.PATH              ; OPEN(PATHNAME:=D.PATH
000443  LDR080          LDA       D.PATH,Y            ;      REFNUM=OPEN.REF
000444                  STA       PATH,Y              ;      SYSBUF.P:=80:LDREND-2000 )
000445                  DEY
000446                  BPL       LDR080
000447  *
000448                  BRK
000449                  DFB       OPEN
000450                  DW        OPEN.PARMS
000451                  BEQ       LDR090
000452                  LDX       #ERR4X              ; ERROR("DRIVER FILE NOT FOUND")
000453                  LDY       #ERR4L
000454                  JSR       ERROR
000455  LDR090          LDA       OPEN.REF
000456                  STA       READ.REF
000457                  STA       CLOSE.REF
000458  *
000459  * READ IN ENTIRE DRIVER FILE INTO BANK 0
000460  *
000461                  BRK                           ; READ(REFNUM=READ.REF
000462                  DFB       READ                ;      RDBUF.P:=80:FILE
000463                  DW        READ.PARMS          ;      BYTES=$FFFF-FILE+1
000464  *                                     ;      BYTESRD=D.BYTESRD )
000465                  BEQ       LDR100
000466                  LDX       #ERR0X              ; ERROR("I/O ERROR")
000467                  LDY       #ERR0L
000468                  JSR       ERROR
000469  *                                                                    +---------------+
000470  * CLOSE THE DRIVER FILE AND CHECK LABEL                               ! SEE FIGURE 3. !
000471  *                                                                    +---------------+
000472  LDR100          BRK                           ; CLOSE(REFNUM=CLOSE.REF)
000473                  DFB       CLOSE
000474                  DW        CLOSE.PARMS
000475                  LDY       #$7                 ; CHECK LABEL
000476  LDR101          LDA       (RDBUF.P),Y
000477                  CMP       D.LABEL,Y
000478                  BNE       LDR102
000479                  DEY
000480                  BPL       LDR101
000481                  BMI       LDR103
```

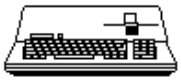


```
000482  LDR102          LDX         #ERR5X              ; ERROR("INVALID DRIVER FILE")
000483                  LDY         #ERR5L
000484                  JSR         ERROR
000485  *
000486  * MOVE CHARACTER SET TABLE
000487  *
000488  LDR103          LDA         #>D.CHRSET          ; MOVE(SRC.P=D.CHRSET DST.P=$C00 A=0 CNT=$400)
000489                  STA         SRC.P
000490                  LDA         #<D.CHRSET
000491                  STA         SRC.P+1
000492                  LDA         #>$C00
000493                  STA         DST.P
000494                  LDA         #<$C00
000495                  STA         DST.P+1
000496                  LDA         #>$400
000497                  STA         CNT
000498                  LDA         #<$400
000499                  STA         CNT+1
000500                  LDA         #0
000501                  JSR         MOVE
000502  *
000503  * MOVE KEYBOARD TABLE
000504  *
000505                  LDA         #>D.KYBD            ; MOVE(SRC.P=D.KYBD DST.P=$1700 A=0 CNT=$100.IN)
000506                  STA         SRC.P
000507                  LDA         #<D.KYBD
000508                  STA         SRC.P+1
000509                  LDA         #>$1700
000510                  STA         DST.P
000511                  LDA         #<$1700
000512                  STA         DST.P+1
000513                  LDA         #>$100
000514                  STA         CNT
000515                  LDA         #<$100
000516                  STA         CNT+1
000517                  LDA         #0
000518                  JSR         MOVE
000519  *
000520  * RE-INITIALIZE SDT TABLE
000521  *
000522                  LDY         #>D.DRIVES-D.FILE   ; LINK.INIT(A=D.DRIVES DIB1..4.IN, SDT.TBL BLKDLST.IO)
000523                  LDA         (RDBUF.P),Y
000524                  JSR         LINK.INIT
000525  *
000526                  LDA         #0                  ; DST.P:=0:I.BASE.P/256*256
000527                  STA         CXPAGE+DST.P+1
000528                  STA         DST.P
000529                  LDA         I.BASE.P+1
000530                  STA         DST.P+1
000531                  CMP         #$A0                ; IF DST.P>=$A000 THEN DST.P:=$A000
000532                  BCC         LDR105
000533                  LDA         #$A0
000534                  STA         DST.P+1
000535  LDR105          LDA         SYSBANK             ; DSTBANK:=SYSBANK
000536                  STA         DSTBANK
000537                  JSR         REVERSE             ; REVERSE(D.HDR.CNT.IN, WORK.P.OUT)
000538  *
000539  * RELOCATE AND MOVE DRIVERS
000540  *
000541  NEXTDRIVER      JSR         DADVANCE            ; "NO DRIVERS LEFT":=DADVANCE(WORK.P.IO SRC.P CNT REL.P.OUT)
000542                  BCS         LDR140
000543                  JSR         FLAGS               ; "INACTIVE":=FLAGS(SRC.P.IN, PG.ALIGN FIRST.ADIB.OUT)
000544                  BVS         NEXTDRIVER
000545                  JSR         GETMEM              ; GETMEM(PG.ALIGN CNT.IN, DST.P DSTBANK DSEGLIST.IO, PREVBANK.OUT)
000546                  JSR         RELOC               ; RELOC(SRC.P REL.P DST.P.IN)
000547  *
000548                  LDA         DSTBANK             ; IF DSTBANK < 0 OR DST.P < SRC.P THEN ERROR
000549                  BMI         LDR120
000550                  LDA         CXPAGE+SRC.P+1      ;    (CONVERT SRC.P TO BANK SWITCHED ADDRESS)
000551                  AND         #$7F
000552                  STA         TEMP.BANK
000553                  LDA         SRC.P+1
000554                  BPL         LDR110
000555                  INC         TEMP.BANK
000556  LDR110          AND         #$7F
000557                  CLC
000558                  ADC         #<$2000
000559                  STA         TEMP.ADRH
000560                  LDA         DST.P               ;    (NOW COMPARE)
000561                  CMP         SRC.P
000562                  LDA         DST.P+1
```

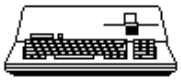


```
000563                  SBC         TEMP.ADRH
000564                  LDA         DSTBANK
000565                  SBC         TEMP.BANK
000566                  BCS         LDR130
000567  LDR120          LDX         #ERR6X              ;    ERROR("DRIVER FILE TOO LARGE")
000568                  LDY         #ERR6L
000569                  JSR         ERROR
000570  *
000571  LDR130          LDA         DSTBANK             ; MOVE(SRC.P DST.P A=DSTBANK CNT.IN)
000572                  JSR         MOVE
000573                  JSR         LINK                ; LINK(DST.P DSTBANK PREVBANK FIRST.ADIB.IN, SDT.TBL BLKDLST.IO)
000574                  JMP         NEXTDRIVER
000575                  REP         100
000576  * SETUP USER ENVIRONMENT
000577                  REP         100
000578  *
000579  * RE-INITIALIZE KERNEL/DRIVERS, ALLOCATE SYSTEM SEGMENTS
000580  *
000581  LDR140          JSR         INIT.KRNL           ; INIT.KRNL()
000582                  JSR         ALLOC.SEG           ; ALLOC.SEG(K.BASE I.BASE.P SYSBANK.IN)
000583                  JSR         ALLOC.DSEG          ; ALLOC.DSEG(DSEGLIST.IN)
000584  *
000585  * SET PREFIX TO THE BOOT VOLUME
000586  *
000587                  LDA         #0                  ; TURN VIDEO OFF - PREVENTS CHAR "GROWTH" DURING DOWNLOAD
000588                  STA         SCRNMODE
000589                  BRK                             ; SET.PREFIX(PREFIXPATH=".D1")
000590                  DFB         SETPREFIX
000591                  DW          PREFX.PARMS
000592  *
000593  * LAUNCH CHARACTER SET DOWNLOAD (CONSOLE) AND CLEAR SCREEN
000594  *
000595                  CLI                             ; BEGIN CHARACTER SET DOWNLOAD (CONSOLE)
000596  *
000597                  LDA         #0                  ; CLEAR TEXT SCREENS
000598                  STA         CXPAGE+SRC.P+1
000599                  STA         CXPAGE+DST.P+1
000600                  LDA         #$04
000601                  STA         SRC.P+1
000602                  STA         DST.P+1
000603                  LDA         #$00
000604                  STA         SRC.P
000605                  LDA         #$80
000606                  STA         DST.P
000607                  LDA         #$A0
000608                  LDX         #8
000609  CLEAR0          LDY         #$77
000610  CLEAR1          STA         (SRC.P),Y
000611                  STA         (DST.P),Y
000612                  DEY
000613                  BPL         CLEAR1
000614                  INC         SRC.P+1             ; NEXT PAGE
000615                  INC         DST.P+1             ; NEXT PAGE
000616                  DEX
000617                  BNE         CLEAR0
000618  *
000619  WAIT            INC         SRC.P               ; WAIT FOR DOWNLOAD TO COMPLETE
000620                  BNE         WAIT
000621                  INX
000622                  BNE         WAIT
000623  *
000624                  LDA         #$80                ; TURN VIDEO ON
000625                  STA         SCRNMODE
000626                  RTS
000627                  REP         100
000628
000629                  CHN         SOSLDR.E.SRC
000630
000631  ********************************************************************************
000632  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.D.SRC
000633  ********************************************************************************
000634
000635
```

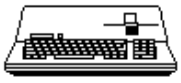
End of File -- Lines: 635 Characters: 19090



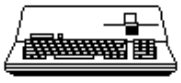
```
================================================================================
FILE: "SOS.SOSLDR.E.SRC.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.E.SRC
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP         100
000008  *
000009  * SET.DRIVES ( IN:   A=# DRIVES
000010  *              IN:   DIB1..4    )
000011  * (INITIALIZES DIB LINKS IN KERNEL'S FLOPPY DRIVER)
000012                  REP         100
000013  *
000014  SET.DRIVES      EQU         *
000015                  TAY                             ; SAVE # OF DRIVES
000016                  LDA         #>DIB2              ; DIB1:=ADR(DIB2)
000017                  STA         DIB1
000018                  LDA         #<DIB2
000019                  STA         DIB1+1
000020                  LDA         #>DIB3              ; DIB2:=ADR(DIB3)
000021                  STA         DIB2
000022                  LDA         #<DIB3
000023                  STA         DIB2+1
000024                  LDA         #>DIB4              ; DIB3:=ADR(DIB4)
000025                  STA         DIB3
000026                  LDA         #<DIB4
000027                  STA         DIB3+1
000028  *
000029                  LDA         #0                  ; CASE (Y=# OF DRIVES)
000030                  CPY         #2
000031                  BCC         STDR010
000032                  BEQ         STDR020
000033                  CPY         #4
000034                  BCC         STDR030
000035                  BCS         STDR040
000036  *
000037  STDR010         STA         DIB1                ;    1:  DIB1:=0
000038                  STA         DIB1+1
000039                  RTS
000040  *
000041  STDR020         STA         DIB2                ;    2:  DIB2:=0
000042                  STA         DIB2+1
000043                  RTS
000044  *
000045  STDR030         STA         DIB3                ;    3:  DIB3:=0
000046                  STA         DIB3+1
000047                  RTS
000048  *
000049  STDR040         STA         DIB4                ;    4:  DIB4:=0
000050                  STA         DIB4+1
000051                  RTS                             ; RETURN
000052                  PAGE
000053                  REP         100
000054  *
000055  * INIT.KRNL ()
000056  *
000057  * (CALLS KERNEL INITIALIZATION MODULES)
000058                  REP         100
000059  *
000060  INIT.KRNL       EQU         *
000061                  LDA         E.REG               ; SWITCH IN I/O BANK AND SELECT PRIMARY STACK
000062                  ORA         #$44                ; E:=( 0.1.1.X:0.1.0.0 )
000063                  STA         E.REG               ;    ( 1.I.S.R:W.P.R.R )
000064  *
000065                  LDA         #<SZPAGE            ; SWITCH TO SOS ZPAGE
000066                  STA         Z.REG
000067  *
000068                  JSR         INT.INIT            ; CALL KERNEL INITIALIZATION ROUTINES
000069                  JSR         EVQ.INIT
000070                  JSR         BFM.INIT2
000071                  BCS         INITK.ERR
000072                  JSR         DMGR.INIT
000073                  JSR         CFMGR.INIT
000074                  JSR         MMGR.INIT
000075                  JSR         BMGR.INIT
000076                  JSR         BFM.INIT
```



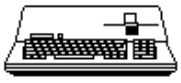
```
000077                  JSR         CLK.INIT
000078  *
000079                  LDA         E.REG               ; SWITCH OUT I/O BANK AND RETURN TO ALTERNATE STACK
000080                  AND         #$BB                ; E:=( 0.0.1.X:0.0.0.0 )
000081                  STA         E.REG               ;    ( 1.I.S.R:W.P.R.R )
000082  *
000083                  LDA         #<CZPAGE            ; SWITCH BACK TO USER ZPAGE
000084                  STA         Z.REG
000085  *
000086                  RTS                             ; RETURN
000087  *
000088  *
000089  INITK.ERR       LDX         #ERR0X              ; ERROR("I/O ERROR")
000090                  LDY         #ERR0L
000091                  JMP         ERROR
000092                  PAGE
000093                  REP         100
000094  *
000095  * ADVANCE ( I/O:  WORK.P
000096  *           OUT:  SRC.P
000097  *           OUT:  DST.P
000098  *           OUT:  CNT     )
000099  * (ADVANCES WORK.P TO NEXT INTERP.KERNEL MODULE.  INITS SRC.P, DST.P, CNT FOR MOVE)
000100                  REP         100
000101  *
000102  ADVANCE         EQU         *
000103                  CLC
000104                  LDY         #2                  ; Y:=0
000105                  LDA         WORK.P              ; WORK.P:=WORK.P+(WORK.P),Y + 4
000106                  ADC         (WORK.P),Y
000107                  TAX
000108                  INY
000109                  LDA         WORK.P+1
000110                  ADC         (WORK.P),Y
000111                  PHA
000112                  TXA
000113                  ADC         #4
000114                  STA         WORK.P
000115                  PLA
000116                  ADC         #0
000117                  STA         WORK.P+1
000118                  CLC                             ; SRC.P:=X:WORK.P+4
000119                  LDA         WORK.P
000120                  ADC         #>$0004
000121                  STA         SRC.P
000122                  LDA         WORK.P+1
000123                  ADC         #<$0004
000124                  STA         SRC.P+1
000125                  LDA         CXPAGE+WORK.P+1
000126                  STA         CXPAGE+SRC.P+1
000127                  LDY         #0                  ; DST.P:=0:(WORK.P)
000128                  STY         CXPAGE+DST.P+1
000129                  LDA         (WORK.P),Y
000130                  STA         DST.P
000131                  INY
000132                  LDA         (WORK.P),Y
000133                  STA         DST.P+1
000134                  INY                             ; Y:=2
000135                  LDA         (WORK.P),Y          ; CNT:=(WORK.P),Y
000136                  STA         CNT
000137                  INY
000138                  LDA         (WORK.P),Y
000139                  STA         CNT+1
000140                  RTS                             ; RETURN
000141                  PAGE
000142                  REP         100
000143  *
000144  * REVERSE ( IN:   D.HDR.CNT
000145  *           IN:   SDT.SIZE = CONSTANT
000146  *           I/O:  DRIVER FILE,
000147  *           OUT:  WORK.P      )      )
000148  *
000149  *            LOCAL:  REV.SAVE, REV.TEMP
000150  * (REVERSES TITLE/CODE/RELOC COUNTS TO ALLOW DRIVER FILE TO BE PROCESSED FROM BACK TO FRONT)
000151                  REP         100
000152  REVERSE         EQU         *
000153                  LDA         #>D.HDR.CNT         ; WORK.P:=80:D.HDR.CNT
000154                  STA         WORK.P
000155                  LDA         #<D.HDR.CNT
000156                  STA         WORK.P+1
000157                  LDA         #$80
```



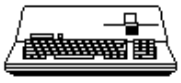
```
000158                  STA         CXPAGE+WORK.P+1
000159                  CLC                             ; WORK.P:=WORK.P+(WORK.P)+2
000160                  LDY         #0
000161                  LDA         WORK.P
000162                  ADC         (WORK.P),Y
000163                  TAX
000164                  INY
000165                  LDA         WORK.P+1
000166                  ADC         (WORK.P),Y
000167                  PHA
000168                  TXA
000169                  ADC         #2
000170                  STA         WORK.P
000171                  PLA
000172                  ADC         #0
000173                  STA         WORK.P+1
000174                  LDA         (WORK.P),Y          ; IF (WORK.P)=$FFFF
000175                  DEY
000176                  AND         (WORK.P),Y          ;    THEN
000177                  CMP         #$FF
000178                  BNE         REV010
000179                  LDX         #ERR10X             ;       ERROR("EMPTY DRIVER FILE")
000180                  LDY         #ERR10L
000181                  JSR         ERROR
000182  REV010          LDA         #$FF
000183                  STA         REV.SAVE
000184                  STA         REV.SAVE+1
000185  *
000186  REV020          LDA         REV.SAVE            ;R1: STACK:=REV.SAVE
000187                  PHA
000188                  LDA         REV.SAVE+1
000189                  PHA
000190                  LDY         #0                  ;    REV.SAVE:=(WORK.P)
000191                  LDA         (WORK.P),Y
000192                  STA         REV.SAVE
000193                  INY
000194                  LDA         (WORK.P),Y
000195                  STA         REV.SAVE+1
000196                  PLA                             ;    (WORK.P):=STACK
000197                  STA         (WORK.P),Y
000198                  DEY
000199                  PLA
000200                  STA         (WORK.P),Y
000201                  LDA         REV.SAVE            ;    IF REV.SAVE = $FFFF THEN EXIT
000202                  AND         REV.SAVE+1
000203                  CMP         #$FF
000204                  BEQ         REV.EXIT
000205  REV030          BIT         REV.SAVE+1          ;    IF REV.SAVE >= $8000 THEN ERROR
000206                  BMI         REV040
000207                  CLC                             ;    WORK.P:=WORK.P+REV.SAVE+2
000208                  LDA         WORK.P
000209                  ADC         REV.SAVE
000210                  TAX
000211                  LDA         WORK.P+1
000212                  ADC         REV.SAVE+1
000213                  PHA
000214                  BCS         REV040
000215                  TXA
000216                  ADC         #2
000217                  STA         WORK.P
000218                  PLA
000219                  ADC         #0
000220                  STA         WORK.P+1
000221                  BCC         REV020              ;    IF C=FALSE THEN R1
000222  REV040          LDX         #ERR5X              ;                ELSE ERROR("INVALID DRIVER FILE")
000223                  LDY         #ERR5L
000224                  JSR         ERROR
000225  *
000226  REV.EXIT        RTS                             ; RETURN
000227                  PAGE
000228                  REP         100
000229  *
000230  * DADVANCE ( I/O:  WORK.P
000231  *            OUT:  C="NO DRIVERS LEFT"
000232  *            OUT:  SRC.P
000233  *            OUT:  CNT
000234  *            OUT:  REL.P )
000235  * (ADVANCES WORK.P TO NEXT DRIVER MODULE.  INITS SRC.P, CNT, REL.P FOR RELOCATION AND MOVE)
000236                  REP         100
000237  DADVANCE        EQU         *
000238                  LDY         #0                  ; IF (WORK.P)=$FFFF THEN EXIT "NO DRIVERS LEFT IN FILE"
```



```
000239                  LDA         (WORK.P),Y
000240                  INY
000241                  AND         (WORK.P),Y
000242                  CMP         #$FF
000243                  BNE         DADV010
000244                  SEC                             ; C:="NO DRIVERS LEFT"
000245                  RTS                             ; RETURN
000246  *
000247  *
000248  DADV010         LDA         WORK.P              ; REL.P:=X:WORK.P
000249                  STA         REL.P
000250                  LDA         WORK.P+1
000251                  STA         REL.P+1
000252                  LDA         CXPAGE+WORK.P+1
000253                  STA         CXPAGE+REL.P+1
000254  *
000255                  JSR         DADD                ; ADVANCE TO CODE COUNT FIELD
000256  *
000257                  LDY         #0                  ; CNT:=(WORK.P)
000258                  LDA         (WORK.P),Y
000259                  STA         CNT
000260                  INY
000261                  LDA         (WORK.P),Y
000262                  STA         CNT+1
000263  *
000264                  JSR         DADD                ; ADVANCE TO TITLE CNT FIELD
000265  *
000266                  CLC                             ; SRC.P:=X:WORK.P+2
000267                  LDA         WORK.P
000268                  ADC         #2
000269                  STA         SRC.P
000270                  LDA         WORK.P+1
000271                  ADC         #0
000272                  STA         SRC.P+1
000273                  LDA         CXPAGE+WORK.P+1
000274                  STA         CXPAGE+SRC.P+1
000275  *
000276                  JSR         DADD                ; ADVANCE TO RELOC FIELD OF NEXT DRIVER
000277                  CLC                             ; C:="DRIVERS LEFT"
000278                  RTS                             ; RETURN
000279                  PAGE
000280                  REP         100
000281  *
000282  * DADD ( I/O:   WORK.P )
000283  *
000284  * (ADVANCES WORK.P TO NEXT FIELD IN DRIVER MODULE)
000285                  REP         100
000286  DADD            EQU         *
000287                  SEC                             ; WORK.P:=WORK.P-(WORK.P)-2
000288                  LDY         #0
000289                  LDA         WORK.P
000290                  SBC         (WORK.P),Y
000291                  TAX
000292                  INY
000293                  LDA         WORK.P+1
000294                  SBC         (WORK.P),Y
000295                  PHA
000296                  TXA
000297                  SBC         #2
000298                  STA         WORK.P
000299                  PLA
000300                  SBC         #0
000301                  STA         WORK.P+1
000302                  RTS                             ; RETURN
000303                  PAGE
000304                  REP         100
000305  *
000306  * FLAGS ( IN:   SRC.P
000307  *         OUT:  PG.ALIGN
000308  *         OUT:  FIRST.ADIB
000309  *         OUT:  OV="ALL DIBS INACTIVE" )
000310  *
000311  *         LOCAL:  PREV.ADIB.P, DIB.P
000312  * (PROCESSES "INACTIVE" & "PAGE ALIGN" FLAGS IN DRIVER MODULE'S DIBS"
000313                  REP         100
000314  FLAGS           EQU         *
000315                  SEC                             ; C="FIRST DIB"
000316  FLAG010         JSR         NEXT.DIB            ; NEXT.DIB(SRC.P.IN, DIB.P PG.ALIGN C OV.OUT)
000317                  BVC         FLAG015             ; IF OV <> "INACTIVE" THEN ACTIVE DIB FOUND
000318                  BCC         FLAG010             ; IF C <> "LAST DIB" THEN CHECK NEXT DIB
000319                  RTS                             ; RETURN  (OV:="ALL DIBS INACTIVE")
```

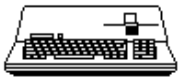


```
000320  *
000321  FLAG015         PHP                             ; PUSH STATUS
000322                  SEC                             ; FIRST.ADIB:=DIB.P-SRC.P
000323                  LDA         DIB.P
000324                  SBC         SRC.P
000325                  STA         FIRST.ADIB
000326                  LDA         DIB.P+1
000327                  SBC         SRC.P+1
000328                  STA         FIRST.ADIB+1
000329                  LDA         DIB.P               ; PREV.ADIB.P:=X:DIB.P
000330                  STA         PREV.ADIB.P
000331                  LDA         DIB.P+1
000332                  STA         PREV.ADIB.P+1
000333                  LDA         CXPAGE+DIB.P+1
000334                  STA         CXPAGE+PREV.ADIB.P+1
000335                  PLP                             ; PULL STATUS
000336                  BCS         FLAG100             ; IF C="LAST DIB" THEN EXIT
000337  *
000338  FLAG020         JSR         NEXT.DIB            ; NEXT.DIB(SRC.P.IN, DIB.P PG.ALIGN C OV.OUT)
000339                  PHP                             ; PUSH STATUS
000340                  LDY         #0                  ; IF OV="INACTIVE DIB"
000341                  BVC         FLAG025
000342                  SEC                             ;    THEN
000343                  LDA         PREV.ADIB.P         ;       (PREV.ADIB.P):=PREV.ADIB.P-SRC.P
000344                  SBC         SRC.P
000345                  STA         (PREV.ADIB.P),Y
000346                  INY
000347                  LDA         PREV.ADIB.P+1
000348                  SBC         SRC.P+1
000349                  STA         (PREV.ADIB.P),Y
000350                  JMP         FLAG050
000351  *
000352  FLAG025         SEC                             ;    ELSE
000353                  LDA         DIB.P               ;       (PREV.ADIB.P):=DIB.P-SRC.P
000354                  SBC         SRC.P
000355                  STA         (PREV.ADIB.P),Y
000356                  INY
000357                  LDA         DIB.P+1
000358                  TAX
000359                  SBC         SRC.P+1
000360                  STA         (PREV.ADIB.P),Y
000361                  STX         PREV.ADIB.P+1       ;       PREV.ADIB.P:=DIB.P
000362                  LDA         DIB.P
000363                  STA         PREV.ADIB.P
000364  FLAG050         PLP                             ; PULL STATUS
000365                  BCC         FLAG020             ; IF C <> "LAST DIB" THEN PROCESS NEXT DIB
000366  *
000367  FLAG100         CLV                             ; OV:="ACTIVE DIBS"
000368                  RTS                             ; RETURN
000369                  PAGE
000370                  REP         100
000371  *
000372  * NEXT.DIB ( IN:   C="FIRST DIB"
000373  *            IN:   SRC.P
000374  *            OUT:  DIB.P
000375  *            OUT:  PG.ALIGN
000376  *            OUT:  C="LAST DIB"
000377  *            OUT:  OV="INACTIVE DIB" )
000378  *
000379  *            LOCAL:  DIB.FLAGS, DIB.DCB = CONSTANT
000380  * (ADVANCES TO NEXT DIB IN DRIVER MODULE)
000381                  REP         100
000382  NEXT.DIB        EQU         *
000383                  LDY         #0
000384                  BCC         NXTD010             ; IF C = "FIRST DIB"
000385                  STY         PG.ALIGN            ;    THEN
000386                  STY         PG.ALIGN+1          ;       PG.ALIGN:=0
000387                  LDA         SRC.P               ;       DIB.P:=X:SRC.P
000388                  STA         DIB.P
000389                  LDA         SRC.P+1
000390                  STA         DIB.P+1
000391                  LDA         CXPAGE+SRC.P+1
000392                  STA         CXPAGE+DIB.P+1
000393                  JMP         NXTD020
000394  NXTD010         LDA         SRC.P               ;    ELSE
000395                  ADC         (DIB.P),Y           ;       DIB.P:=SRC.P+(DIB.P)
000396                  TAX
000397                  INY
000398                  LDA         SRC.P+1
000399                  ADC         (DIB.P),Y
000400                  STA         DIB.P+1
```

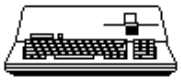


```
000401                  STX         DIB.P
000402  *
000403  NXTD020         LDY         #DIB.FLAGS          ; IF (DIB.P),DIB.FLAGS.BIT7 = "INACTIVE"
000404                  LDA         (DIB.P),Y
000405                  BMI         NXTD030
000406                  BIT         NXTD999             ;    THEN
000407                  BVS         NXTD040             ;       OV:="INACTIVE"
000408  *                                               ELSE
000409  NXTD030         AND         #$40                ;       IF (DIB.P),DIB.FLAGS.BIT6 = "PAGE ALIGN"
000410                  BEQ         NXTD040
000411                  CLC                             ;          THEN
000412                  LDA         #DIB.DCB+2          ;             PAGE.ALIGN:=DIB.DCB+2+(SRC.P),DIB.DCB
000413                  TAY
000414                  DEY
000415                  DEY
000416                  ADC         (SRC.P),Y
000417                  STA         PG.ALIGN
000418                  INY
000419                  LDA         #0
000420                  ADC         (SRC.P),Y
000421                  STA         PG.ALIGN+1
000422                  CLV                             ;       OV:="ACTIVE"
000423  *
000424  NXTD040         LDY         #0                  ; IF (DIB.P) = 0
000425                  LDA         (DIB.P),Y
000426                  INY
000427                  ORA         (DIB.P),Y
000428                  BNE         NXTD998
000429                  SEC                             ;    THEN  C:="LAST DIB"
000430                  BCS         NXTD999
000431  NXTD998         CLC                             ;    ELSE  C:=NOT "LAST DIB"
000432  NXTD999         RTS                             ; RETURN
000433                  REP         100
000434
000435                  CHN         SOSLDR.F.SRC
000436
000437                  RTS                             ; RETURN
000438
000439  ********************************************************************
000440  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.E.SRC
000441  ********************************************************************
000442
```

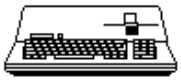
End of File -- Lines: 442 Characters: 12097



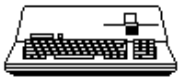
```
================================================================================
FILE: "SOS.SOSLDR.F.SRC.TEXT"
================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.F.SRC
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP         100
000008  *
000009  * GETMEM ( IN:   PG.ALIGN
000010  *          IN:   CNT
000011  *          I/O:  DST.P
000012  *          I/O:  DSTBANK
000013  *          I/O:  DSEGLIST
000014  *          OUT:  PREVBANK )
000015  *
000016  *          LOCAL:  PREVDST
000017  * (COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT)
000018                  REP         100
000019  GETMEM          EQU         *
000020                  LDA         DSTBANK             ; PREVBANK:=DSTBANK
000021                  STA         PREVBANK
000022                  LDA         DST.P               ; PREVDST:=DST.P
000023                  STA         PREVDST
000024                  LDA         DST.P+1
000025                  STA         PREVDST+1
000026                  JSR         NEWDST              ; NEWDST(PG.ALIGN.IN, PREVDST.IN, CNT.IN, DST.P.OUT)
000027  *
000028                  LDA         DST.P+1             ; IF DST.P >= $2000
000029                  CMP         #$20
000030                  BCC         GETM010
000031                  SEC                             ;    THEN
000032                  LDA         PREVDST+1           ;       A=PAGES:=PREVDST-DST.P
000033                  SBC         DST.P+1
000034                  CLC
000035                  JSR         BUILD.DSEG          ;       BUILD.DSEG(C="NEXT BANK".IN, A=PAGES.IN, DSEGLIST.IO)
000036                  JMP         GETM.EXIT
000037  *                                           ELSE
000038  GETM010         DEC         DSTBANK             ;       DSTBANK:=DSTBANK-1
000039                  LDA         #>$A000             ;       PREVDST:=$A000
000040                  STA         PREVDST
000041                  LDA         #<$A000
000042                  STA         PREVDST+1
000043                  JSR         NEWDST              ;       NEWDST(PG.ALIGN.IN, PREVDST.IN, CNT.IN, DST.P.OUT)
000044                  SEC                             ;       A="PAGES":=PREVDST-DST.P
000045                  LDA         PREVDST+1
000046                  SBC         DST.P+1
000047                  SEC
000048                  JSR         BUILD.DSEG          ;       BUILD.DSEG(C="NEXTBANK".IN, A="PAGES".IN, DSEGLIST.IO)
000049  *
000050  GETM.EXIT       RTS                             ; RETURN
000051                  PAGE
000052                  REP         100
000053  *
000054  * NEWDST ( IN:   PG.ALIGN
000055  *          IN:   PREVDST
000056  *          IN:   CNT
000057  *          I/O:  DST.P      )
000058  * (COMPUTES DESTINATION BASE ADDRESS, ALIGNING ON PAGE BOUNDARY IF REQUESTED)
000059                  REP         100
000060  NEWDST          EQU         *
000061                  SEC                             ; IF (PREVDST-$2000) < CNT
000062                  LDA         PREVDST
000063                  SBC         #>$2000
000064                  TAX
000065                  LDA         PREVDST+1
000066                  SBC         #<$2000
000067                  CPX         CNT
000068                  SBC         CNT+1
000069                  BCS         NEWD010
000070                  LDA         #0                  ;    THEN
000071                  STA         DST.P               ;      DST.P:=0
000072                  STA         DST.P+1
000073                  BEQ         NEWD.EXIT
000074  NEWD010         SEC                             ;    ELSE
000075                  LDA         PREVDST             ;       DST.P:=PREVDST-CNT
000076                  SBC         CNT
```



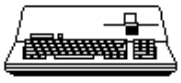
```
000077                  STA         DST.P
000078                  LDA         PREVDST+1
000079                  SBC         CNT+1
000080                  STA         DST.P+1
000081                  LDA         PG.ALIGN            ;       IF PG.ALIGN <> 0
000082                  ORA         PG.ALIGN+1          ;          THEN
000083                  BEQ         NEWD.EXIT
000084                  SEC                             ;             DST.P:=(DST.P/256*256)-PG.ALIGN
000085                  LDA         #0
000086                  SBC         PG.ALIGN
000087                  STA         DST.P
000088                  LDA         DST.P+1
000089                  SBC         PG.ALIGN+1
000090                  STA         DST.P+1
000091  NEWD.EXIT       RTS                             ; RETURN
000092                  PAGE
000093                  REP         100
000094  *
000095  * BUILD.DSEG ( IN:   C="NEXTBANK"
000096  *              IN:   A="PAGES"
000097  *              I/O:  DSEGLIST     )
000098  * (COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT)
000099                  REP         100
000100  BUILD.DSEG      EQU         *
000101                  PHA
000102                  BCS         BLDS010             ; IF ("NEXTBANK"=TRUE OR DSEGX=$FF)
000103                  LDA         DSEGX               ;    THEN
000104                  BPL         BLDS020
000105  BLDS010         INC         DSEGX               ;       DSEGX:=DSEGX+1
000106  BLDS020         LDX         DSEGX
000107                  CLC                             ; DSEGLIST(DSEGX):=DSEGLIST(DSEGX)+"PAGES"
000108                  PLA
000109                  ADC         DSEGLIST,X
000110                  STA         DSEGLIST,X
000111                  RTS                             ; RETURN
000112  *
000113  *
000114  *
000115  DSEGX           DFB         $FF                 ; DRIVER SEGMENT LIST TABLE
000116  DSEGLIST        DFB         $0                  ; # PAGES FOR 1ST DRIVER SEGMENT   (BANK N  )
000117                  DFB         $0                  ;        "    2ND       "          (BANK N-1)
000118                  DFB         $0                  ;        "    3RD       "          (BANK N-2)
000119                  DFB         $0                  ;        "    4TH       "          (BANK N-3)
000120                  PAGE
000121                  REP         100
000122  *
000123  * RELOC ( IN:   SRC.P
000124  *         IN:   REL.P
000125  *         IN:   DST.P
000126  *         OUT:  RELOCATED DRIVER MODULE )
000127  *
000128  *         LOCAL:  REL.END, CODE.P
000129  * (RELOCATES DRIVER MODULE'S CODE FIELD USING RELOCATION FIELD)
000130                  REP         100
000131  RELOC           EQU         *
000132                  SEC                             ; REL.END:=REL.P-(REL.P)
000133                  LDY         #0
000134                  LDA         REL.P
000135                  SBC         (REL.P),Y
000136                  STA         REL.END
000137                  INY
000138                  LDA         REL.P+1
000139                  SBC         (REL.P),Y
000140                  STA         REL.END+1
000141  REL.LOOP        SEC                             ; REL.P:=REL.P-2
000142                  LDA         REL.P
000143                  SBC         #2
000144                  STA         REL.P
000145                  LDA         REL.P+1
000146                  SBC         #0
000147                  STA         REL.P+1
000148                  LDA         REL.P               ; IF REL.P < REL.END THEN EXIT
000149                  CMP         REL.END
000150                  LDA         REL.P+1
000151                  SBC         REL.END+1
000152                  BCC         REL.EXIT
000153                  LDY         #0                  ; CODE.P:=X:SRC.P+(REL.P)
000154                  CLC
000155                  LDA         SRC.P
000156                  ADC         (REL.P),Y
000157                  STA         CODE.P
```



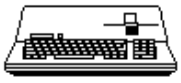
```
000158                  INY
000159                  LDA         SRC.P+1
000160                  ADC         (REL.P),Y
000161                  STA         CODE.P+1
000162                  LDA         CXPAGE+SRC.P+1
000163                  STA         CXPAGE+CODE.P+1
000164                  LDY         #0                  ; (CODE.P):=(CODE.P)+DST.P
000165                  CLC
000166                  LDA         (CODE.P),Y
000167                  ADC         DST.P
000168                  STA         (CODE.P),Y
000169                  INY
000170                  LDA         (CODE.P),Y
000171                  ADC         DST.P+1
000172                  STA         (CODE.P),Y
000173                  JMP         REL.LOOP            ; GOTO REL.LOOP
000174  *
000175  REL.EXIT        RTS                             ; RETURN
000176                  PAGE
000177                  REP         100
000178  *
000179  * ALLOC.SEG ( IN:   K.BASE
000180  *             IN:   I.BASE.P
000181  *             IN:   SYSBANK )
000182  *         I.BASE.P
000183  *         D.BASE.PG
000184  * (ALLOCATES SEGMENTS FOR KERNEL, INTERPRETER AND SYSTEM WORK AREA)
000185                  REP         100
000186  ALLOC.SEG       EQU         *
000187                  BRK                             ; REQ.SEG(BASE=(F,0), LIMIT=(F,1D), SEGID=0, SEGNUM)
000188                  DFB         REQSEG
000189                  DW          SEGMENT
000190  *
000191                  LDA         #$10                ; SET BASE/LIMIT BANKS
000192                  STA         SEGBASE
000193                  STA         SEGLIM
000194                  LDA         #0                  ; AND INIT BASE PAGE
000195                  STA         SEGBASE+1
000196  *
000197                  LDX         K.BASE+1            ; KERNEL SEGMENT, ID=1
000198                  JSR         RSEG
000199  *
000200                  LDX         I.BASE.P+1          ; INTERPRETER SEGMENT, ID=2
000201                  JSR         RSEG
000202                  RTS
000203                  PAGE
000204                  REP         100
000205  *
000206  * RSEG ( IN:  X=BASE.PAGE OF SEGMENT )
000207  *
000208                  REP         100
000209  RSEG            EQU         *
000210                  INC         SEGID               ; SEGID:=SEGID+1
000211                  LDY         SEGBASE+1           ; LIMIT.PAGE:=BASE.PAGE-1
000212                  DEY
000213                  STY         SEGLIM+1
000214                  STX         SEGBASE+1           ; BASE.PAGE:=X
000215  *
000216                  CPX         #$A0                ; IF BASE>=$A0 OR LIMIT<$A0 THEN
000217                  BCS         RSEG010             ;    THEN
000218                  LDA         SEGLIM+1            ;       REQUEST ONLY ONE SEGMENT
000219                  CMP         #$A0
000220                  BCC         RSEG010
000221  *
000222                  TXA                             ;    ELSE
000223                  PHA                             ;       REQUEST TWO SEGMENTS
000224                  LDX         #$A0
000225                  STX         SEGBASE+1
000226  *
000227                  BRK                             ;       REQ.SEG(BASE, LIMIT, SEGID, SEGNUM)
000228                  DFB         REQSEG
000229                  DW          SEGMENT
000230  *
000231                  PLA
000232                  STA         SEGBASE+1
000233                  LDA         #$9F
000234                  STA         SEGLIM+1
000235                  LDA         SYSBANK
000236                  STA         SEGBASE
000237                  STA         SEGLIM
000238  *
```



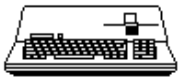
```
000239  *
000240  RSEG010         BRK                             ; REQ.SEG(BASE, LIMIT, SEGID, SEGNUM)
000241                  DFB         REQSEG
000242                  DW          SEGMENT
000243  *
000244                  RTS                             ; RETURN
000245                  PAGE
000246                  REP         100
000247  *
000248  * ALLOC.DSEG ( IN:   DSEGLIST )
000249  *
000250  * (ALLOCATES SEGMENTS FOR DRIVER MODULES"
000251                  REP         100
000252  ALLOC.DSEG      EQU         *
000253                  INC         DSEGX               ; DSEGX:=DSEGX+1
000254                  BNE         ALDS010             ; IF DSEGX=0
000255                  LDX         #ERR5X              ;    THEN ERROR("INVALID DRIVER FILE")
000256                  LDY         #ERR5L
000257                  JSR         ERROR
000258  *
000259  ALDS010         LDY         #$FF                ; Y:=-1
000260  ALDS020         INY                             ; WHILE (Y:=Y+1) < DSEGX
000261                  CPY         DSEGX               ;    DO
000262                  BCS         ALDS.EXIT
000263                  LDA         DSEGLIST,Y          ;       PAGECT:=DSEGLIST(Y)
000264                  STA         SEGPGCNT
000265                  BRK                             ;       FINDSEG (SRCHMODE=0.IN, SEGID=3.IN
000266                  DFB         FINDSEG             ;               PAGECT=DSEGLIST(Y)
000267                  DW          SEGMENT1            ;               BASE.OUT, LIMIT.OUT)
000268                  JMP         ALDS020
000269  *
000270  ALDS.EXIT       RTS                             ; RETURN
000271                  PAGE
000272                  REP         100
000273  *
000274  * ERROR (IN:  X=MESSAGE INDEX
000275  *        IN:  Y=MESSAGE LENGTH
000276  * (DISPLAYS ERROR MESSAGE, SOUNDS BELL AND LOOPS UNTIL CONTROL/RESET PRESSED)
000277                  REP         100
000278  ERROR           EQU         *
000279                  STY         ETEMP               ; CENTER MSG (Y:=LEN/2+LEN)
000280                  SEC
000281                  LDA         #40
000282                  SBC         ETEMP
000283                  LSR         A
000284                  CLC
000285                  ADC         ETEMP
000286                  TAY
000287  *
000288  PRNT010         LDA         ERR,X               ; MOVE MESSAGE TO SCREEN MEMORY
000289                  STA         EMSGADR-1,Y
000290                  DEX
000291                  DEY
000292                  DEC         ETEMP
000293                  BNE         PRNT010
000294  *
000295                  LDA         #$73                ; E:=( 0.1.1.1:0.0.1.1 )
000296                  STA         E.REG               ;    ( 1.I.S.R:W.P.R.S )
000297                  LDA         $C040               ; SOUND BELL
000298                  JMP         *                   ; LOOP UNTIL REBOOT (CTRL/RESET)
000299                  PAGE
000300                  REP         100
000301  *
000302  * ERROR MESSAGES
000303  *
000304                  REP         100
000305  EMSGADR         EQU         $7A8
000306  *
000307  ERR             EQU         *
000308  ERR0            ASC         "I/O ERROR"
000309  ERR0L           EQU         *-ERR0
000310  ERR0X           EQU         *-ERR-1
000311  ERR1            ASC         "INTERPRETER FILE NOT FOUND"
000312  ERR1L           EQU         *-ERR1
000313  ERR1X           EQU         *-ERR-1
000314  ERR2            ASC         "INVALID INTERPRETER FILE"
000315  ERR2L           EQU         *-ERR2
000316  ERR2X           EQU         *-ERR-1
000317  ERR3            ASC         "INCOMPATIBLE INTERPRETER"
000318  ERR3L           EQU         *-ERR3
000319  ERR3X           EQU         *-ERR-1
```



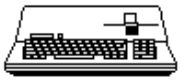
```
000320  ERR4            ASC         "DRIVER FILE NOT FOUND"
000321  ERR4L           EQU         *-ERR4
000322  ERR4X           EQU         *-ERR-1
000323  ERR5            ASC         "INVALID DRIVER FILE"
000324  ERR5L           EQU         *-ERR5
000325  ERR5X           EQU         *-ERR-1
000326  ERR6            ASC         "DRIVER FILE TOO LARGE"
000327  ERR6L           EQU         *-ERR6
000328  ERR6X           EQU         *-ERR-1
000329  ERR7            ASC         "ROM ERROR:  PLEASE NOTIFY YOUR DEALER"
000330  ERR7L           EQU         *-ERR7
000331  ERR7X           EQU         *-ERR-1
000332  ERR8            ASC         "TOO MANY DEVICES"
000333  ERR8L           EQU         *-ERR8
000334  ERR8X           EQU         *-ERR-1
000335  ERR9            ASC         "TOO MANY BLOCK DEVICES"
000336  ERR9L           EQU         *-ERR9
000337  ERR9X           EQU         *-ERR-1
000338  ERR10           ASC         "EMPTY DRIVER FILE"
000339  ERR10L          EQU         *-ERR10
000340  ERR10X          EQU         *-ERR-1
000341                  PAGE
000342                  REP         100
000343  *
000344  * WELCOME ()
000345  *
000346  * (PRINTS WELCOME MESSAGE - "APPLE ///", VERSION, DATE/TIME, COPYRIGHT)
000347                  REP         100
000348  WELCOME         EQU         *
000349  *
000350  *  PRINT "APPLE III" MESSAGE
000351  *
000352                  LDY         #AMSGL
000353  WAM010          LDA         AMSG-1,Y
000354                  STA         AMSGADR-1,Y
000355                  DEY
000356                  BNE         WAM010
000357  *
000358  *  PRINT SOS VERSION MESSAGE
000359  *
000360                  CLC
000361                  LDA         #40
000362                  ADC         #>SOSVERL
000363                  LSR         A
000364                  TAX
000365                  LDY         #>SOSVERL
000366  WSM010          LDA         SOSVER-1,Y
000367                  ORA         #$80
000368                  STA         SMSGADR-1,X
000369                  DEX
000370                  DEY
000371                  BNE         WSM010
000372  *
000373  *  PRINT DATE AND TIME MESSAGE
000374  *
000375                  BRK                             ; GET.TIME(TIME.OUT)
000376                  DFB         GETTIME
000377                  DW          DTPARMS
000378  *
000379                  LDA         DATETIME+8          ;SET UP WEEKDAY
000380                  AND         #$0F
000381                  BEQ         WDM040              ;NO CLOCK
000382                  STA         WTEMP
000383                  ASL         A
000384                  ADC         WTEMP
000385                  TAX
000386                  LDY         #3
000387  WDM010          LDA         DAYNAME-1,X
000388                  STA         DMSG-1,Y
000389                  DEX
000390                  DEY
000391                  BNE         WDM010
000392  *
000393                  LDA         DATETIME+7          ;SET UP DATE
000394                  LDX         DATETIME+6
000395                  STA         DMSG+6
000396                  STX         DMSG+5
000397  *
000398                  LDA         DATETIME+5          ;SET UP MONTH
000399                  AND         #$0F
000400                  LDX         DATETIME+4
```



```
000401                  CPX         #$31
000402                  BCC         WDM020
000403                  ADC         #9
000404  WDM020          STA         WTEMP
000405                  ASL         A
000406                  ADC         WTEMP
000407                  TAX
000408                  LDY         #3
000409  WDM030          LDA         MONNAME-1,X
000410                  STA         DMSG+7,Y
000411                  DEX
000412                  DEY
000413                  BNE         WDM030
000414  *
000415                  LDA         DATETIME+3          ;SET UP YEAR
000416                  LDX         DATETIME+2
000417                  STA         DMSG+13
000418                  STX         DMSG+12
000419  *
000420                  LDA         DATETIME+10         ;SET UP HOUR
000421                  LDX         DATETIME+09
000422                  STA         DMSG+17
000423                  STX         DMSG+16
000424  *
000425                  LDA         DATETIME+12         ;SET UP MINUTE
000426                  LDX         DATETIME+11
000427                  STA         DMSG+20
000428                  STX         DMSG+19
000429  *
000430                  LDY         #DMSGL              ;PRINT DATE & TIME
000431  WDM050          LDA         DMSG-1,Y
000432                  ORA         #$80
000433                  STA         DMSGADR-1,Y
000434                  DEY
000435                  BNE         WDM050
000436  *
000437  *  PRINT COPYRIGHT MESSAGE
000438  *
000439  WDM040          LDY         #CMSGL
000440  WCM010          LDA         CMSG-1,Y
000441                  STA         CMSGADR-1,Y
000442                  DEY
000443                  BNE         WCM010
000444                  RTS
000445                  PAGE
000446                  REP         100
000447  *
000448  * WELCOME () - DATA DECLARATIONS
000449  *
000450                  REP         100
000451                  MSB         ON
000452  AMSG            ASC         "APPLE ///"
000453  AMSGL           EQU         *-AMSG
000454  AMSGADR         EQU         40-AMSGL/2+$4A8
000455                  MSB         OFF
000456  SMSGADR         EQU         $5A8
000457  DMSG            ASC         "DAY, DD-MON-YY  HH:MM"
000458  DMSGL           EQU         *-DMSG
000459  DMSGADR         EQU         40-DMSGL/2+$6A8
000460  DAYNAME         ASC         "SUNMONTUEWEDTHUFRISAT"
000461  MONNAME         ASC         "JANFEBMARAPRMAYJUN"
000462                  ASC         "JULAUGSEPOCTNOVDEC"
000463                  MSB         ON
000464  CMSG            ASC         "(C)1980,1981,1982 BY APPLE COMPUTER INC."
000465  CMSGL           EQU         *-CMSG
000466  CMSGADR         EQU         40-CMSGL/2+$7D0
000467                  MSB         OFF
000468                  PAGE
000469                  REP         100
000470  *
000471  * SOS SYSTEM CALLS (1)
000472  *
000473                  REP         100
000474  * OPEN (PATHNAME.IN, REFNUM.OUT, OPENLIST.IN, OPENCNT.IN)  ** (ACCESS.IN, PAGES.IN, SYSBUF.IN)
000475                  REP         100
000476  OPEN            EQU         $C8
000477  *
000478  OPEN.PARMS      DFB         $4
000479                  DW          PATH
000480  OPEN.REF        DFB         $0
000481                  DW          OPEN.LIST
```

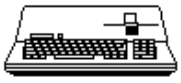


```
000482                  DFB         $4
000483  OPEN.LIST       DFB         $0,$4               ; PAGES:=4
000484                  DW          SYSBUF.P
000485  PATH            DS          $40                 ; PATHNAME BUFFER
000486  I.LABEL         ASC         "SOS NTRP"          ; FILE LABELS
000487  D.LABEL         ASC         "SOS DRVR"
000488                  REP         100
000489  * READ (REFNUM.IN, BUFFER.IN, BYTES.IN, BYTESREAD.OUT)
000490                  REP         100
000491  READ            EQU         $CA
000492  *
000493  READ.PARMS      DFB         $4
000494  READ.REF        DFB         $0
000495  READ.BUF        DW          RDBUF.P
000496  READ.BYT        DW          $FFFF-FILE+1
000497  READ.BYTRD      DW          $0
000498                  REP         100
000499  * CLOSE (REFNUM.IN)
000500                  REP         100
000501  CLOSE           EQU         $CC
000502  *
000503  CLOSE.PARMS     DFB         $1
000504  CLOSE.REF       DFB         $0
000505                  REP         100
000506  * FIND.SEG (SRCHMODE.IN, PAGES.IN, SEGID.IN, BASE.OUT, LIMIT.OUT, SEGNUM.OUT)
000507                  REP         100
000508  FINDSEG         EQU         $41
000509  *
000510  SEGMENT1        DFB         $6                  ; FIND.SEG(SRCHMODE, SEGID, PAGECT, BASE, LIMIT, SEGNUM)
000511  SEGSRCH         DFB         $0,$3
000512  SEGPGCNT        DW          $0000
000513                  DW          $0
000514                  DW          $0
000515                  DFB         $0
000516                  PAGE
000517                  REP         100
000518  *
000519  * SOS SYSTEM CALLS (2)
000520  *
000521                  REP         100
000522                  REP         100
000523  * REQUEST.SEG (BASE.IN, LIMIT.IN, SEGID.IN, SEGNUM.OUT)
000524                  REP         100
000525  REQSEG          EQU         $40
000526  *
000527  SEGMENT         DFB         $4                  ; REQUEST SEG PARM LIST
000528  SEGBASE         DFB         $F,$0
000529  SEGLIM          DFB         $F,$1D
000530  SEGID           DFB         $0,$0
000531                  REP         100
000532  * SET.PREFIX (PREFIXPATH.IN)
000533                  REP         100
000534  SETPREFIX       EQU         $C6
000535  PREFX.PARMS     DFB         $1
000536                  DW          PREFX.PATH
000537  PREFX.PATH      DFB         $3
000538                  ASC         '.D1'
000539                  REP         100
000540  * GETTIME (TIME.OUT)
000541                  REP         100
000542  GETTIME         EQU         $63
000543  *
000544  DTPARMS         DFB         1
000545                  DW          DATETIME
000546  DATETIME        ASC         "YYYYMMDDWHHMMSSMMM"
000547                  PAGE
000548                  REP         100
000549  *
000550  * END OF SOSLDR CODE
000551  *
000552                  REP         100
000553  SLOP            EQU         >$F8-*
000554                  DS          SLOP                ; +----------------------------------+
000555  INITMODULE      DS          $200                ; ! KERNEL'S INIT MODULE RESIDES HERE !
000556  LDREND          EQU         *                   ; +----------------------------------+
000557  FILE            EQU         *-$2000+$400
000558                  REP         100
000559  * SOS INTERPRETER FILE
000560                  REP         100
000561  I.FILE          EQU         FILE
000562  I.HDR.CNT       EQU         I.FILE+$8
```

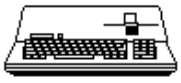


```
000563                  REP         100
000564  * SOS DRIVER FILE
000565                  REP         100
000566  D.FILE          EQU         FILE
000567  D.HDR.CNT       EQU         D.FILE+$8
000568  D.DRIVES        EQU         D.HDR.CNT+$2
000569  D.CHRSET        EQU         D.DRIVES+$2+$10
000570  D.KYBD          EQU         D.CHRSET+$10+$400
000571                  REP         100
000572
000573                  LST         ON
000574  ZZEND           EQU         *
000575  ZZLEN           EQU         ZZEND-ZZORG
000576  *
000577  NE              ZZLEN-LENLODR
000578                  FAIL        2,"SOSORG            FILE IS INCORRECT FOR SOS LOADER"
000579                  FIN
000580  *
000581
000582  ********************************************************************
000583  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.F.SRC
000584  ********************************************************************
000585
```

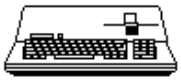
End of File -- Lines: 585 Characters: 15585



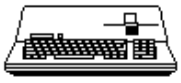
```
================================================================================================
FILE: "SOS.SOSLDR.SRC.TEXT"
================================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.SRC
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1 SOS LOADER"
000007                  REL
000008                  ORG        $1E00
000009  ZZORG           EQU        *
000010                  MSB        OFF
000011                  REP        100
000012  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000013  *                      ALL RIGHTS RESERVED
000014                  REP        100
000015  *
000016  *        SOS KERNEL LOAD & MEMORY POINTS
000017  *
000018  *  MODULE      START  END   I/O  ROM  SOS BLOAD    SIZE
000019  *-------------------------------------------------------
000020  *  SOSLDR      1E00 - 28F7            2000         0CF8
000021  *  INIT        28F8 - 2AA9            2AF8        [01B2]
000022  *  SYSGLOB     18FC - 1A03            2CF8
000023  *
000024  *  BFM.INIT2 + BITMAPS
000025  *              B800 - BBFF            2E00         03FF
000026  *  BFM         BC00 - DE62            3200         2263
000027  *  <PATCH>     DE63 - DE6A            5463         0008
000028  *
000029  *  OPRMSG      DE6B - E48A   X        546B         015A
000030  *  IPL         DFC5 - E48F   X    X   55C5         04CB
000031  *  UMGR        E490 - E89D   X    X   5A8B         040E
000032  *
000033  *  DISK3       E899 - EE03   X    X   5E99         056B
000034  *  SYSERR      EE04 - EED8   X        64D9         00D5
000035  *  DEVMGR      EED9 - F05D            64D9         0185
000036  *
000037  *  SCMGR       F05E - F2F3            665E         0296
000038  *  FMGR        F2F4 - F354            68F4         0061
000039  *  CFMGGR      F355 - F551            6955         01FD
000040  *
000041  *  BUFMGR      F552 - F86D            6B52         031C
000042  *  MEMMGR      F86E - FFBE            6E6E         0751
000043  *  <END>       FFBE
000044  *
000045                  REP        100
000046  * SOS LOADER  (VERSION = 1.1O   )
000047  *             (DATE    = 8/04/81)
000048  *
000049  * SOURCE FILES:  SOSLDR.SRC,   SOSLDR.A.SRC, SOSLDR.B.SRC, SOSLDR.C.SRC,
000050  *                              SOSLDR.D.SRC, SOSLDR.E.SRC, SOSLDR.F.SRC
000051  *
000052  * FUNCTION:
000053  *    MOVES AND INITIALIZES SOS KERNEL, READS INTERPRETER FROM DISK, READS CHARACTER SET TABLE,
000054  *    KEYBOARD TABLE AND DRIVERS FROM DISK, INITIALIZES ALL DRIVERS AND THEN JUMPS TO INTERPRETER
000055  *    ENTRY POINT.
000056  *
000057  * CALLED BY:
000058  *    SOSBOOT 7.0 WITH KERNEL FILE LOADED AT $I:1E00.9FFF(MAX)
000059  *    WHERE: $I=INTERPRETER BANK (HIGHEST BANK IN SYSTEM)
000060  *
000061  * CALLS:
000062  *    INTERPRETER ENTRY POINT (FIRST BYTE OF INTERPRETER CODE)
000063  *
000064  * DOCUMENTS:
000065  *    SOS ERS APPENDICES - XX/XX/81
000066  *    APPLE III I/O SYSTEM PROGRAMMERS GUIDE - DEC-15-80
000067  *
000068  * CONSTRAINTS:
000069  *    INTERPRETER FILE:  READ INTO BANK 0 BEGINNING AT $80:LDREND+$400(=BUFSIZE).
000070  *                       INTERPRETER CODE DOES NOT CONTAIN RELOCATION INFORMATION.
000071  *                       MAX = 38K  ($I:2000..B7FF)
000072  *                       MIN = .25K ($I:B700..B7FF)
000073  *
000074  *    DRIVER FILE:  READ INTO BANK 0 BEGINNING AT $80:LDREND+$400(=BUFSIZE).
000075  *                  DRIVER MODULES ARE RELOCATED AND MOVED TO THE HIGHEST AVAILABLE 32K BANK USING
000076  *                  A "FIRST FIT" ALGORITHM.  MODULES ARE REMOVED FROM THE FILE BEGINNING AT THE BACK
```



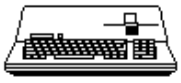
```
000077  *                  AND WORKING TOWARD THE FRONT.  A DRIVER MODULE CANNOT SPAN A BANK BOUNDARY.
000078  *
000079  *                  DRIVER FILE:  MAX = 60K  (APPROX)       DRIVER MODULE:  MAX = 32K-1
000080  *                                MIN = .25K                                MIN < .25K
000081  *
000082  *
000083  * DATA STRUCTURES:
000084  *    SOS.KERNEL FILE FORMAT
000085  *    SOS.INTERP FILE FORMAT
000086  *    SOS.DRIVER FILE FORMAT
000087  *
000088                  REP        100
000089                  PAGE
000090                  REP        100
000091  *
000092  * NOTATION:
000093  *
000094  *    A, X, Y           ::= 6502 REGISTERS
000095  *
000096  *    C, OV             ::= CARRY, OVERFLOW FLAGS IN 6502 STATUS (P) REGISTER
000097  *    E, Z, B           ::= ENVIRONMENT, ZERO PAGE, BANK REGISTERS (SYSTEM CONTROL REGISTERS)
000098  *
000099  *    (1.I.S.R:W.P.R.R) ::= ENVIRONMENT REGISTER FLAGS.  FROM LEFT TO RIGHT BITS 7..0
000100  *                          (1MHZ, I/O ENABLE, SCREEN ENABLE, RESET ENABLE,
000101  *                           WRITE PROTECT, PRIMARY STACK, ROM1, ROM ENABLE)
000102  *
000103  *    "POSITIVE LOGIC"  ::= ALL LOGIC USED IS POSITIVE LOGIC.  FOR EXAMPLE, C="NO DRIVERS LEFT"
000104  *                          INDICATES THAT NO DRIVERS ARE LEFT WHEN CARRY = SET, AND THAT ONE OR
000105  *                          MORE DRIVERS ARE LEFT WHEN CARRY = CLEAR.
000106  *
000107  *    TRUE,FALSE        ::= TRUE = SET = ON, WHILE FALSE = CLEAR = OFF.
000108  *
000109                  REP        100
000110  *
000111  * ABBREVIATIONS:
000112  *
000113  *    DIB               ::= DEVICE INFORMATION BLOCK.  DEFINES A UNIQUE DEVICE THAT CAN BE LINKED
000114  *                          INTO THE SYSTEM DEVICE TABLE.  EACH DRIVER MODULE CONTAINS ONE OR MORE
000115  *                          DIBS (DEVICES) EACH OF WHICH CAN BE "ACTIVE" OR "INACTIVE".
000116  *
000117  *    ADIB              ::= "ACTIVE DIB"
000118  *
000119  *   <VARNAME>.P        ::=  POINTER.  A 3 BYTE ZERO PAGE POINTER.  DON'T FORGET THE X BYTE!
000120  *
000121  *    SDT               ::= SYSTEM DEVICE TABLE.  CONTAINS THE ENTRY POINT AND DIB ADDRESS OF EACH
000122  *                          DEVICE CONFIGURED INTO THE SYSTEM, (USED BY THE DEVICE MANAGER).
000123                  REP        100
000124
000125                  CHN        SOSLDR.A.SRC
000126
000127  ********************************************************************************
000128  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.SRC
000129  ********************************************************************************
000130
000131

End of File -- Lines: 131 Characters: 4487
```



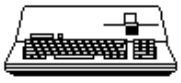
```
================================================================================================
FILE: "SOS.SOSORG.TEXT"
================================================================================================

000001  ************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSORG
000003  ************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  REP     100
000007  *  SOS KERNEL MODULE ORIGINS
000008  ORGLODR         EQU     $1E00           ; ORIGIN OF SOS LOADER
000009  ORGINIT         EQU     $28F8           ; ORIGIN OF INIT
000010  ORGGLOB         EQU     $18FC           ; ORIGIN OF SYSGLOB
000011  ORGBFMI         EQU     $B800           ; ORIGIN OF BFM.INIT2 & BITMAPS
000012  ORGBFM          EQU     $BC00           ; ORIGIN OF BFM
000013  ORGPATCH        EQU     $DE66           ; ORIGIN OF PATCH AREA
000014  ORGOMSG         EQU     $DE66           ; ORIGIN OF OPRMSG
000015  ORGIPL          EQU     $DFC0           ; ORIGIN OF IPL
000016  ORGUMGR         EQU     $E48B           ; ORIGIN OF UMGR
000017  ORGDISK3        EQU     $E899           ; ORIGIN OF DISK3
000018  ORGSERR         EQU     $EE04           ; ORIGIN OF SYSERR
000019  ORGDMGR         EQU     $EED9           ; ORIGIN OF DEVMGR
000020  ORGSCMGR        EQU     $F05E           ; ORIGIN OF SCMGR
000021  ORGFMGR         EQU     $F2F4           ; ORIGIN OF FMGR
000022  ORGCFM          EQU     $F355           ; ORIGIN OF CFMGR
000023  ORGBUFMG        EQU     $F552           ; ORIGIN OF BUFMGR
000024  ORGMEMMG        EQU     $F86E           ; ORIGIN OF MEMMGR
000025  ORGEND          EQU     $FFBF           ; END MARKER
000026                  REP     100
000027  *  LENGTH OF SOS MODULES  -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
000028  LENLODR         EQU     ORGINIT-ORGLODR     ; LENGTH OF SOS LOADER
000029  LENINIT         EQU     $01B2               ; LENGTH OF INIT
000030  LENBFMI         EQU     ORGBFM-ORGBFMI      ; LENGTH OF BFM.INIT2 & BITMAPS
000031  LENBFM          EQU     ORGPATCH-ORGBFM     ; LENGTH OF BFM
000032  LENPATCH        EQU     ORGOMSG-ORGPATCH    ; LENGTH OF PATCH AREA
000033  LENOMSG         EQU     ORGIPL-ORGOMSG      ; LENGTH OF OPRMSG
000034  LENIPL          EQU     ORGUMGR-ORGIPL      ; LENGTH OF IPL
000035  LENUMGR         EQU     ORGDISK3-ORGUMGR    ; LENGTH OF UMGR
000036  LENDISK3        EQU     ORGSERR-ORGDISK3    ; LENGTH OF DISK3
000037  LENSERR         EQU     ORGDMGR-ORGSERR     ; LENGTH OF SYSERR
000038  LENDMGR         EQU     ORGSCMGR-ORGDMGR    ; LENGTH OF DEVMGR
000039  LENSCMGR        EQU     ORGFMGR-ORGSCMGR    ; LENGTH OF SCMGR
000040  LENFMGR         EQU     ORGCFM-ORGFMGR      ; LENGTH OF FMGR
000041  LENCFM          EQU     ORGBUFMG-ORGCFM     ; ORIGIN OF CFMGR
000042  LENBUFMG        EQU     ORGMEMMG-ORGBUFMG   ; LENGTH OF BUFMGR
000043  LENMEMMG        EQU     ORGEND-ORGMEMMG     ; LENGTH OF MEMMGR
000044                  REP     100
000045  *       SOS BLOAD ADDRESSES
000046  BLALODR         EQU     $2000               ; BLOAD ADDRESS OF SOS LOADER
000047  BLAINIT         EQU     BLALODR+LENLODR     ; BLOAD ADDRESS OF INIT
000048  BLAGLOB         EQU     $2CF8               ; BLOAD ADDRESS OF SYSGLOB
000049  BLABFMI         EQU     $2E00               ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
000050  BLABFM          EQU     $3200               ; BLOAD ADDRESS OF BFM
000051  BLAPATCH        EQU     BLABFM+LENBFM       ; BLOAD ADDRESS OF PATCH AREA
000052  BLAOMSG         EQU     BLAPATCH+LENPATCH   ; BLOAD ADDRESS OF OPRMSG
000053  BLAIPL          EQU     BLAOMSG+LENOMSG     ; BLOAD ADDRESS OF IPL
000054  BLAUMGR         EQU     BLAIPL+LENIPL       ; BLOAD ADDRESS OF UMGR
000055  BLADISK3        EQU     BLAUMGR+LENUMGR     ; BLOAD ADDRESS OF DISK3
000056  BLASERR         EQU     BLADISK3+LENDISK3   ; BLOAD ADDRESS OF SYSERR
000057  BLADMGR         EQU     BLASERR+LENSERR     ; BLOAD ADDRESS OF DEVMGR
000058  BLASCMGR        EQU     BLADMGR+LENDMGR     ; BLOAD ADDRESS OF SCMGR
000059  BLAFMGR         EQU     BLASCMGR+LENSCMGR   ; BLOAD ADDRESS OF FMGR
000060  BLACFM          EQU     BLAFMGR+LENFMGR     ; BLOAD ADDRESS OF CFMGR
000061  BLABUFMG        EQU     BLACFM+LENCFM       ; BLOAD ADDRESS OF BUFMGR
000062  BLAMEMMG        EQU     BLABUFMG+LENBUFMG   ; BLOAD ADDRESS OF MEMMGR
000063                  REP     100
000064
000065  ************************************************************************
000066  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSORG
000067  ************************************************************************
000068

End of File -- Lines: 68 Characters: 3776
```



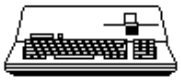
```
================================================================================
FILE: "SOS.SWAPOUT.IN.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SWAPOUT.IN
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  SWAPOUT         EQU        *
000007  *
000008  * SWAP OUT A VOLUME LOGGED ON A DEVICE
000009  * INPUT ARGUMENT: DEVICE NUMBER "A"
000010  * (STORED AS "DEVNUM")
000011  * OUTPUT ARGUMENT: NONE
000012  * CONDITION CODE: CARRY SET USER DID NOT COMPLY WITH REQUEST
000013  *
000014  * SAVE VCBPTR, FCBPTR, DEVNUM ON STACK
000015  * 1) FIND UNSWAPPED VOLUME IN VCB
000016  * 2) IF DIRTY BIT MAP FOR THIS VOLUME THEN DO
000017  *   IF NOT ONLINE, REQUEST USER TO INSERT
000018  *   IF REQUEST DENIED, UNCONDITIONALLY  CLOSE ALL FILES ON THIS VOLUME AND RTS
000019  *   IF ONLINE, UPDATE AND RELEASE BIT MAP
000020  * DOEND
000021  * 3) SWAP IT (MARK VCBSWAP FIELD $80, MARK ALL FILES ON THIS VOLUME WITH SWAP MARK $8X WHERE X=VCB ENTRY)
000022  * "VCB ENTRY" DEFINED AS: HIGH ORDER NIBBLE OF LOW ORDER BYTE OF ENTRIES VCB ADDRESS
000023  * RESTORE VCBPTR, FCBPTR
000024  * RTS
000025  *
000026                  TAX                             ; SAVE DEVICE NUMBER
000027                  JSR        SAVECBS
000028                  STX        DEVNUM               ; PERMANENTLY
000029  SWAPOUTX        JSR        DEVVCB               ; FIND MATCHING UNSWAPPED ACTIVE VCB ENTRY (BY DEVNUM)
000030                  BCS        SORTS                ; NO FIND--RETURN WITHOUT ERROR
000031                  LDY        #VCBSTAT
000032                  LDA        (VCBPTR),Y           ; GET STATUS OF FILES ON THIS VOLUME
000033                  BPL        UNLOG                ; IF NO OPEN FILES, JUST THROW VOLUME AWAY
000034                  LDA        DEVNUM               ; DIRTY BM EXIST ON THIS VOLUME?
000035                  LDX        #0
000036                  CMP        BMADEV,X             ; IN BIT MAP "A"?
000037                  BEQ        FDIRBM               ; BRANCH IF YES
000038                  LDX        #6                   ; BIT MAP HEADER TABLE SIZE
000039                  CMP        BMADEV,X             ; IN BIT MAP "B"?
000040                  BEQ        FDIRBM               ; BRANCH IF YES
000041                  JMP        MARKSWAP             ; NO NEED TO WRITE BIT MAP
000042  FDIRBM          LDA        BMASTAT,X            ; IS BIT MAP DIRTY?
000043                  BPL        MARKSWAP             ; BRANCH IF NOT
000044  GETVOL          JSR        VERFYVOL             ; IS THE CORRECT VOLUME ON LINE NOW?
000045                  BCC        VONLINE              ; BRANCH IF YES
000046                  JSR        USRREQ               ; OTHERWISE, REQUEST USER INSERTION
000047                  BCC        GETVOL               ; AND VERIFY IT AGAIN
000048                  JSR        CLOSEU               ; USER SAID "NO": UNCONDITIONALLY CLOSE VOLUME
000049                  JSR        RESTCBS
000050                  SEC
000051                  RTS                             ; ERROR RETURN TO CALLER
000052  VONLINE         LDX        DEVNUM               ; UPDATE THE
000053                  JSR        UPBMAP               ; DIRTY BIT MAP
000054  MARKSWAP        LDA        VCBPTR               ; CALCULATE
000055                  LSR        A                    ; SWAP BYTE
000056                  LSR        A                    ; AND
000057                  LSR        A                    ; MARK ALL FILES
000058                  LSR        A                    ; BELONGING TO THIS VOLUME
000059                  SEC                             ; AS SWAPPED OUT
000060                  ORA        #$80
000061                  PHA                             ; SAVE SWAP BYTE
000062                  JSR        FCBSCAN
000063                  PLA                             ; MARK VCBSWAP
000064                  LDY        #VCBSWAP             ; BYTE
000065                  STA        (VCBPTR),Y
000066  SORTS           JSR        RESTCBS              ; RESTORE FCBPTR, VCBPTR, DEVNUM
000067                  CLC
000068                  RTS                             ; SUCCESSFUL SWAP OUT
000069  UNLOG           LDA        #0
000070                  STA        VCB,X                ; UNLOG VOLUME
000071                  BEQ        SORTS                ; SWAP THE EASY WAY! (BRANCH ALWAYS)
000072  *
000073  *
000074  *
000075  SWAPIN          EQU        *
000076  *
```



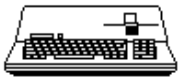
```
000077  * UNSWAP A VOLUME AND ALL ITS FILES
000078  *
000079  * INPUT ARGUMENT: VOLUME NAME (VCBPTR)
000080  * OUTPUT ARGUMENT: NONE
000081  * CONDITION CODE: CARRY SET : USER DID NOT COMPLY WITH REQUEST
000082  *
000083  * SAVE VCBPTR, FCBPTR ON STACK
000084  * 1) FIND SWAPPED VOLUME IN VCB, IF NOT FOUND, THEN RTS.
000085  * 2) IF ANOTHER UNSWAPPED VOLUME ON DEVICE, THEN SWAP IT
000086  * 3) VERIFY UNSWAPPED VOLUME, IF NOT OK THEN REQUEST INSERTION
000087  * 4) UNMARK VCB'S AND FCB'S
000088  * RTS
000089                  JSR         SAVECBS             ; SAVE FCB, VCB POINTERS, DEVNUM
000090                  LDY         #VCBNML             ; MAKE SURE VOLUME
000091                  LDA         (VCBPTR),Y          ; IS AT LEAST OPEN
000092                  BEQ         USRTS               ; BRANCH IF NOT RIGHT BACK TO CALLER
000093                  LDY         #VCBSWAP            ; SEE IF
000094                  LDA         (VCBPTR),Y          ; CURRENTLY SWAPPED
000095                  BEQ         USRTS               ; IF NOT, RETURN IMMEDIATELY TO CALLER
000096                  LDY         #VCBDEV             ; SAVE DEVICE NUMBER
000097                  LDA         (VCBPTR),Y
000098                  STA         DEVNUM
000099                  PHA                             ; SAVE DEVNUM AGAIN (SWAPOUTX TRASHES DEVNUM ON RETURN)
000100                  JSR         SWAPOUTX            ; AND MAKE SURE ANY CURRENT ACTIVE VOLUME IS SWAPPED OUT (NOTICE ENTRY
POINT)
000101                  PLA                             ; RECALL CURRENT DEVICE NUMBER
000102                  STA         DEVNUM              ; AND SAVE IT TO ITS PROPER PLACE
000103  SI1             JSR         VERFYVOL            ; VERIFY THE CURRENT VOLUME MOUNTED
000104                  BCC         UNMARK              ; IF THE RIGHT ONE, GO MARK IT AS UNSWAPPED
000105                  JSR         USRREQ              ; ELSE REQUEST USER TO INSERT
000106                  BCC         SI1                 ; USER SAID 'OK'
000107                  JSR         CLOSEU              ; OTHERWISE UNCONDITIONALLY CLOSE
000108                  JSR         RESTCBS
000109                  SEC
000110                  RTS
000111  UNMARK          LDY         #VCBSWAP            ; FETCH
000112                  LDA         (VCBPTR),Y          ; VOLUME
000113                  PHA                             ; SWAP BYTE
000114                  LDA         #0                  ; BUT CLEAR
000115                  STA         (VCBPTR),Y          ; VOLUME SWAP
000116                  PLA
000117                  CLC                             ; "UNSWAPPED"
000118                  JSR         FCBSCAN
000119                  LDA         DEVNUM              ; MAKE SURE BIT MAPS
000120                  JSR         CLEARBMS            ; ARE MARKED AS INVALID ON THIS DEVICE
000121  USRTS           JSR         RESTCBS             ; RESTORE VCB, FCB PTRS
000122                  CLC                             ; NO ERRORS
000123                  RTS
000124  *
000125  SAVEPTRS        DS          5                   ; A RARE EMBEDDED TEMP SAVE AREA, USED ONLY BY ...
000126  *
000127  *
000128  SAVECBS         EQU         *                   ; SAVE FCBPTR, VCBPTR IN A TEMP SAVE AREA
000129                  LDA         VCBPTR
000130                  STA         SAVEPTRS
000131                  LDA         VCBPTR+1
000132                  STA         SAVEPTRS+1
000133                  LDA         FCBPTR
000134                  STA         SAVEPTRS+2
000135                  LDA         FCBPTR+1
000136                  STA         SAVEPTRS+3
000137                  LDA         DEVNUM
000138                  STA         SAVEPTRS+4
000139                  RTS
000140  *
000141  RESTCBS         EQU         *                   ; RESTORE FCBPTR, VCBPTR
000142  * NOTICE THERE EXISTS A SEQUENCE OF CALLS (SWAPIN, WHICH MAY CALL SWAPOUT) THAT JSR'S TO SAVECBS ONCE BUT JSR'S RESTCBS
TWICE.
000143                  LDA         SAVEPTRS
000144                  STA         VCBPTR
000145                  LDA         SAVEPTRS+1
000146                  STA         VCBPTR+1
000147                  LDA         SAVEPTRS+2
000148                  STA         FCBPTR
000149                  LDA         SAVEPTRS+3
000150                  STA         FCBPTR+1
000151                  LDA         SAVEPTRS+4
000152                  STA         DEVNUM
000153                  RTS
000154  *
000155  *
```



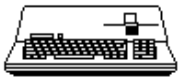
```
000156  * MARK ALL FILES BELONGING TO A VOLUME
000157  * AS SWAPPED-IN OR SWAPPED-OUT.
000158  *
000159  * INPUT ARGS: DEVNUM -- DEVICE NUMBER OF MOUNTED VOLUME
000160  *             A REGISTER - SWAP BYTE
000161  *             CARRY -- CARRY FLAG SET MEANS SWAP OUT; ELSE SWAP IN
000162  *
000163  * OUTPUT ARGS: NONE
000164  * GLOBALS AFFECTED: FCB, FCBPTR
000165  * REGISTER STATUS: SCRAMBLED
000166  *
000167  FCBSCAN         EQU        *                   ; MARK FILES BELONGING TO VOLUME AS SWAPPED OR UNSWAPPED
000168  *
000169                  TAX                            ; SAVE SWAP BYTE
000170                  LDY        FCBADDRH            ; POINT TO
000171                  STY        FCBPTR+1            ; BEGINNING TO FCB
000172                  LDY        #0
000173                  STY        FCBPTR
000174                  BCS        FCBOUT              ; SWAP OUT A VOLUMES FILES
000175  FCBIN           EQU        *                   ; SWAPIN A VOLUMES FILES
000176                  JSR        FCBFETCH            ; GET NEXT ACTIVE FCB CANDIDATE
000177                  BCS        FCBRTS              ; NO MORE FILES TO PROCESS
000178                  LDY        #FCBSWAP
000179                  TXA
000180                  CMP        (FCBPTR),Y          ; SWAP BYTES MATCH?
000181                  BNE        FCBIN1              ; BRANCH IF NOT
000182                  LDA        #0
000183                  STA        (FCBPTR),Y          ; MARK FILE AS SWAPPED IN
000184  FCBIN1          JSR        NEXTFCB             ; ADVANCE FCB POINTER
000185                  BCS        FCBRTS              ; NO MORE TO LOOK AT
000186                  JMP        FCBIN               ; AND LOOK AT NEXT FILE
000187  *
000188  FCBOUT          EQU        *                   ; SWAPPED OUT A VOLUMES FILES
000189                  JSR        FCBFETCH            ; GET NEXT ACTIVE FILE IN FCB
000190                  BCS        FCBRTS              ; NO MORE FILES -- RETURN TO USER
000191                  LDY        #FCBSWAP            ; COMPARE
000192                  LDA        (FCBPTR),Y
000193                  BNE        FCBOUT1             ; ALREADY SWAPPED OUT
000194                  TXA
000195                  STA        (FCBPTR),Y          ; MARK AS SWAPPED
000196  FCBOUT1         JSR        NEXTFCB             ; ADVANCE FCB POINTER
000197                  BCS        FCBRTS
000198                  JMP        FCBOUT              ; SWAP OUT NEXT FILE
000199  *
000200  FCBRTS          RTS
000201  FCBFETCH        EQU        *                   ; GET NEXT ACTIVE FILE FROM FCB
000202  * X REGISTER MUST NOT BE DISTURBED
000203  * USES FCBPTR
000204                  LDY        #FCBDEVN            ; MAKE
000205                  LDA        (FCBPTR),Y          ; SURE DEVICE
000206                  CMP        DEVNUM              ; MATCHES
000207                  BNE        NEXTFCB
000208                  LDY        #FCBREFN            ; MAKE SURE FILE IS ACTIVE
000209                  LDA        (FCBPTR),Y
000210                  BEQ        NEXTFCB             ; BRANCH IF NOT
000211                  CLC
000212                  RTS                            ; RETURN WITH CARRY CLEAR SHOWING AN ACTIVE FILE
000213  NEXTFCB         LDA        FCBPTR
000214                  CLC
000215                  ADC        #$20                ; FCB ENTRY SIZE
000216                  STA        FCBPTR
000217                  BCC        FCBFETCH            ; BRANCH IF NO PAGE CROSS
000218                  LDA        FCBPTR+1
000219                  INC        FCBPTR+1            ; SECOND PAGE
000220                  CMP        FCBADDRH
000221                  BEQ        FCBFETCH            ; LOOK AT PAGE TWO
000222  NEXTEND         SEC
000223                  RTS                            ; SHOW NO MORE FILES TO LOOK AT
000224  USRREQ          EQU        *                   ; OPERATOR CONSOLE MESSAGE INTERFACE
000225  * PRODUCES A MESSAGE REQUESTING
000226  * THE SYSTEM OPERATOR TO MOUNT THE VOLUME
000227  * SPECIFIED BY "VCBPTR" ON DEVICE SPECIFIED
000228  * BY DEVNUM.  THIS MODULE INSISTS
000229  * UPON THE CORRECT OPERATOR ACTION
000230  * UPON THREE FAILURES TO COMPLY,
000231  * THE MODULE WILL SIGNIFY FAILURE WITH
000232  * CARRY SET.  IF THE CORRECT ACTION IS TAKEN,
000233  * CARRY WILL BE RETURNED CLEAR
000234  *
000235  * INPUT ARGS: VOLUME NAME (VCBPTR)
000236  *             DEVICE NUMBER (DEVNUM)
```



```
000237  *
000238  * OUTPUT ARGS: CC = OPERATOR COMPLIED WITH REQUESTED ACTION
000239  *              CS = OPERATOR COULDN'T/DIDN'T COMPLY
000240  *
000241  * GLOBALS AFFECTED: NONE
000242  *
000243  * STATUS OF REGISTERS: UNCERTAIN
000244  *
000245  VNML            EQU       ZPGTEMP             ; VOLUME NAME LENGTH
000246                  LDY       #VCBNML             ; IF ILLEGAL VCB
000247                  LDA       (VCBPTR),Y          ; GET OUT QUICK
000248                  BEQ       NEXTEND             ; BRANCH TO SEC RTS
000249                  LDX       #$E                 ; LENGTH OF NAMED AREA-1
000250                  LDA       #$0                 ; NULLS
000251  UR1             STA       MDEV,X              ; BOTH CLEAR
000252                  STA       MVOL,X              ; IN ONE LOOP
000253                  DEX
000254                  BPL       UR1
000255  *
000256  * DO A D-INFO TO FETCH THE DEVICE NAME
000257  *
000258                  LDA       #5                  ; DO ALL
000259                  STA       $C0                 ; NECESSARY
000260                  LDA       DEVNUM              ; HOUSKEEPING
000261                  STA       $C1                 ; TO SET UP
000262                  LDA       #>MDEV-1            ; A DEVICE MANAGER CALL
000263                  STA       $C2
000264                  LDA       #<MDEV-1
000265                  STA       $C3
000266                  LDA       #$8F                ; EXTEND BYTE
000267                  STA       $14C3
000268                  LDA       #0
000269                  STA       $14C2
000270                  STA       $C4
000271                  STA       $C5
000272                  STA       $C6                 ; ZERO SUPERFLUOUS PARMS
000273                  STA       URDERR              ; RESET FAILURE COUNT
000274                  JSR       RPEATIO0            ; GET INFO FROM BOBS CODE
000275                  LDA       #$20                ; "SPACE" RESTORED
000276                  STA       MDEV-1              ; RESTORED
000277                  LDY       #VCBNML
000278                  LDA       (VCBPTR),Y          ; LENGTH OF VOLUME NAME
000279                  STA       VNML                ; SAVED FOR WORK
000280                  LDA       #0
000281                  TAX
000282                  LDY       #VCBNAM             ; POINT TO BEGINNING OF VOLUME NAME
000283  UR2             LDA       (VCBPTR),Y
000284                  STA       MVOL,X
000285                  INX
000286                  INY                           ; VOLUME NAME MOVED
000287                  DEC       VNML                ; TO MESSAGE BUFFER
000288                  BNE       UR2                 ; CHARACTER BY CHARACTER
000289  URDU            LDX       #>UMB               ; PASS THE AREA'S ADDR
000290                  LDY       #<UMB               ; IN X AND Y REGS,LOW, HIGH)
000291                  JSR       OPMSGRPLY           ; HAVE MESSAGE SYSTEM PRINT IT
000292                  JSR       VERFYVOL            ; DID THE USER COMPLY?
000293                  BCS       URDU1               ; BRANCH IF NOT
000294                  RTS                           ; EXIT--CARRY IS CLEAR
000295  URDU1           INC       URDERR              ; COLLECT USER ERRORS
000296                  LDA       URDERR
000297                  CMP       #3                  ; ONLY THREE TRIES ALLOWED
000298                  BCC       URDU                ; RETRY MESSAGE IF LESS THAN THREE TRIES
000299                  RTS                           ; OTHERWISE RETURN WITH CARRY SET
000300  *
000301  *
000302  *
000303  *
000304  *
000305  * CLOSE UNCONDITIONAL
000306  *
000307  * (USER HAS REPLIED 'N' TO A VOLUME MOUNT REQUEST
000308  * CLOSE ALL FILES ON VOLUME/UNLOG VOLUME
000309  *
000310  * INPUT ARGUMENT: (VCBPTR)
000311  * OUTPUT ARGUMENT: NONE
000312  *
000313  CLOSEU          EQU       *
000314  VSWA            EQU       ZPGTEMP             ; THE 'SWAP BYTE' STORED HERE
000315                  LDY       #VCBDEV             ; FETCH
000316                  LDA       (VCBPTR),Y          ; THE DEVICE NUMBER
000317                  STA       DEVNUM              ; OF THIS VOLUME & SAVE IT
```

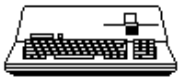


```
000318                  LDY         #VCBSWAP              ; FETCH THE
000319                  LDA         (VCBPTR),Y            ; SWAP BYTE
000320                  STA         VSWA                  ; SAVE FOR REFERENCE, TOO
000321                  LDA         #0
000322                  LDY         #VCBNML               ; UNLOG THE VOLUME
000323                  STA         (VCBPTR),Y            ; BY SETTING LEN OF VOL NAME TO ZERO
000324                  LDY         #VCBSWAP
000325                  STA         (VCBPTR),Y            ; TURN OFF SWAP FLAG
000326                  LDY         FCBADDRH              ; SET UP FCB SCAN FROM BEGINNING OF FCB
000327                  STY         FCBPTR+1
000328                  LDY         #0
000329                  STY         FCBPTR
000330  VFCBLOP         LDY         #FCBDEVN              ; FETCH
000331                  LDA         (FCBPTR),Y            ; THE DEVICE
000332                  CMP         DEVNUM                ; NUMBER AND SEE IF A MATCH
000333                  BNE         VFCBNXT               ; BRANCH IF NO MATCH
000334                  LDY         #FCBREFN              ; SEE EVEN IF FILE OPEN
000335                  LDA         (FCBPTR),Y
000336                  BEQ         VFCBNXT               ; BRANCH IF NOT
000337                  LDY         #FCBSWAP              ; CHECK TO SEE IF ATTACHED
000338                  LDA         (FCBPTR),Y            ; TO SAME VOLUME
000339                  CMP         VSWA
000340                  BNE         VFCBNXT               ; BRANCH IF NOT
000341                  LDY         #FCBBUFN              ; RELEASE
000342                  LDA         (FCBPTR),Y            ; ANY
000343                  JSR         RELBUF                ; BUFFERS ASSOCIATED
000344                  LDY         #FCBSWAP              ; AND CLEAR
000345                  LDA         #0                    ; THE SWAP BYTE
000346                  STA         (FCBPTR),Y
000347                  LDY         #FCBREFN              ; AND FINALLY
000348                  STA         (FCBPTR),Y            ; SAY 'CLOSED'
000349  VFCBNXT         LDA         FCBPTR
000350                  CLC
000351                  ADC         #$20                  ; FCB ENTRY SIZE
000352                  STA         FCBPTR
000353                  BCC         VFCBLOP
000354                  LDA         FCBPTR+1
000355                  INC         FCBPTR+1              ; LOOK AT SECOND PAGE
000356                  CMP         FCBADDRH
000357                  BEQ         VFCBLOP               ; CHECK PAGE TWO OF FCB
000358                  RTS                               ; RETURN TO USER W/O ERROR
000359  *
000360  FCBUSED         EQU         *                     ; MARK AS FCB AS DIRTY SO
000361  * THE DIRECTORY WILL BE FLUSHED ON 'FLUSH'
000362                  STY         ZPGTEMP
000363                  PHA                               ; SAVE REGS
000364                  LDY         #FCBDIRTY
000365                  LDA         (FCBPTR),Y            ; FETCH CURRENT FCBDIRTY BYTE
000366                  ORA         #FCBMOD               ; MARK FCB AS DIRTY
000367                  STA         (FCBPTR),Y            ; SAVE IT BACK
000368                  PLA
000369                  LDY         ZPGTEMP               ; AND RESTORE REGS
000370                  RTS
000371  *
000372  URDERR          DS          1                     ; ERROR COUNT FOR USRREQ
000373  *
000374  *
000375  UMB             EQU         *
000376                  DFB         $49,$6E,$73,$65,$72,$74,$20
000377                  DFB         $76,$6F,$6C,$75,$6D,$65
000378                  DFB         $3A,$20               ; "INSERT VOLUME: "
000379  MVOL            DS          15
000380                  DFB         $0D                   ; CR LINE TERMINATOR
000381                  DFB         $20,$20,$20,$20,$69,$6E,$20
000382                  DFB         $64,$65,$76,$69,$63,$65
000383                  DFB         $3A,$20               ; "    IN DEVICE: "
000384  MDEV            DS          15
000385                  DFB         $0D                   ; CR LINE TERMINATOR
000386                  DFB         $74,$68,$65,$6E,$20,$70,$72
000387                  DFB         $65,$73,$73,$20,$74,$68,$65,$20
000388                  DFB         $41,$4C,$50,$48,$41,$20,$4C
000389                  DFB         $4F,$43,$4B,$20,$6B,$65,$79
000390                  DFB         $20,$74,$77,$69,$63,$65
000391  * "THEN PRESS THE ALPHA LOCK KEY TWICE"
000392  * FOLLOWED WITH $FF MESSAGE TERMINATOR (HIGH BIT SIGNIFICANT)
000393                  DFB         $FF                   ; MESSAGE TERMINATOR (HIGH BIT)
000394  *
000395  ZZLEN           EQU         *-ZZORG
000396  ZZEND           EQU         *
000397                  IFNE        ZZLEN-LENBFM
000398                  FAIL        2,"SOSORG             FILE IS INCORRECT FORMBFM"
```

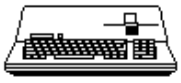


```
000399                  FIN
000400
000401  ********************************************************************
000402  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SWAPOUT.IN
000403  ********************************************************************
000404
```

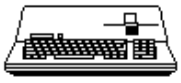
End of File -- Lines: 404 Characters: 15080



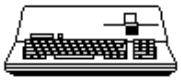
```
================================================================================
FILE: "SOS.SYSERR.SRC.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SYSERR.SRC
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1 SYSTEM ERROR ROUTINES"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGSERR
000010  ZZORG           EQU        *
000011                  MSB        OFF
000012                  REP        60
000013  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000014  *                      ALL RIGHTS RESERVED
000015                  REP        60
000016  *
000017  * SYSTEM ERROR ROUTINES  (VERSION = 1.1O   )
000018  *                        (DATE    = 12/02/81)
000019  *
000020  * THIS MODULE CONTAINS THE SYSTEM ERROR AND SYSTEM FAILURE ROUTINES.
000021  *
000022                  REP        60
000023  *
000024                  ENTRY      SYSERR
000025                  ENTRY      SYSDEATH
000026  *
000027                  EXTRN      SERR
000028                  EXTRN      SDEATH.REGS
000029                  EXTRN      SCRNMODE
000030                  PAGE
000031                  REP        60
000032  *
000033  * DATA DECLARATIONS
000034  *
000035                  REP        60
000036  *
000037  E.REG           EQU        $FFDF
000038  Z.REG           EQU        $FFD0
000039  B.REG           EQU        $FFEF
000040  *
000041  S.SAVE          EQU        $09                 ; REGISTER SAVE AREA
000042  PCH.SAVE        EQU        $08
000043  PCL.SAVE        EQU        $07
000044  P.SAVE          EQU        $06
000045  A.SAVE          EQU        $05
000046  X.SAVE          EQU        $04
000047  Y.SAVE          EQU        $03
000048  E.SAVE          EQU        $02
000049  Z.SAVE          EQU        $01
000050  B.SAVE          EQU        $00
000051  *
000052  NMI.VECTOR      EQU        $FFFA
000053  *
000054  TXT.CLR         EQU        $C050
000055  MIX.CLR         EQU        $C052
000056  HIRES.CLR       EQU        $C056
000057  *
000058  PG2.CLR         EQU        $C054
000059  *
000060  MSGBASE         EQU        $7E4
000061  MSGBASE2        EQU        $BE4
000062  MSG             ASC        ' SYSTEM FAILURE = $'
000063  MSGLEN          EQU        *-MSG
000064                  PAGE
000065                  REP        60
000066  *
000067  * SYSTEM ERROR ROUTINE
000068  *
000069  * THIS ROUTINE IS CALLED WHEN AN ERROR CONDITION HAS BEEN
000070  * ENCOUNTERED.  THE ERROR NUMBER IS PASSED IN THE A REG
000071  * AND THE CALL TO THIS ROUTINE MUST ALWAYS BE A JSR.
000072  *
000073                  REP        60
000074  SYSERR          EQU        *
000075  *
000076                  STA        SERR
```



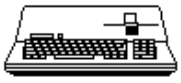
```
000077                  PLA
000078                  STA         SDEATH.REGS+PCL.SAVE
000079                  PLA
000080                  STA         SDEATH.REGS+PCH.SAVE
000081                  SEC
000082                  LDA         SERR
000083                  BNE         SERR.EXIT
000084                  CLC
000085  SERR.EXIT       RTS                             ; RETURNS ONE LEVEL BEYOND CALLER
000086                  PAGE
000087                  REP         60
000088  *
000089  * SYSTEM DEATH ROUTINE
000090  *
000091  * CALLED TO IMMEDIATELY TERMINATE EXECUTION OF THE MACHINE
000092  * BECAUSE A FATAL ERROR HAS BEEN DETECTED BY THE OPERATING
000093  * SYSTEM.  THE ERROR CODE IS PASSED IN THE A REG.  THE
000094  * CALL TO THIS ROUTINE MUST ALWAYS BE A JSR.
000095  *
000096                  REP         60
000097  SYSDEATH        EQU         *
000098  *
000099                  STA         SDEATH.REGS+A.SAVE  ; SAVE REGISTERS
000100                  STX         SDEATH.REGS+X.SAVE
000101                  STY         SDEATH.REGS+Y.SAVE
000102                  PHP
000103                  PLA
000104                  STA         SDEATH.REGS+P.SAVE
000105                  TSX
000106                  STX         SDEATH.REGS+S.SAVE
000107                  LDA         E.REG
000108                  STA         SDEATH.REGS+E.SAVE
000109                  LDA         Z.REG
000110                  STA         SDEATH.REGS+Z.SAVE
000111                  LDA         B.REG
000112                  STA         SDEATH.REGS+B.SAVE
000113                  PLA
000114                  STA         SDEATH.REGS+PCL.SAVE
000115                  PLA
000116                  STA         SDEATH.REGS+PCH.SAVE
000117  *
000118                  SEI                             ; TURN OFF INTERRUPTS
000119                  CLD
000120  *
000121                  LDX         #0                  ; SAVE SYSTEM STACK PAGE IN PAGE $17
000122  SD005           LDA         $100,X
000123                  STA         $1700,X
000124                  DEX
000125                  BNE         SD005
000126  *
000127                  LDA         $C059               ; ENSURE SILENTYPE PORT SHUT DOWN
000128                  LDA         $C0DD
000129                  LDA         $C0DF
000130                  LDA         $C05F
000131                  LDA         $C05A
000132  *
000133                  LDA         $C040               ; SOUND BELL
000134  *
000135                  LDA         #$74                ; ENSURE RESET LOCK OFF & RAM SWITCHED IN.
000136                  STA         E.REG
000137  *
000138                  LDA         TXT.CLR             ; SWITCH TO 40 COL B&W DISPLAY MODE
000139                  LDA         MIX.CLR
000140                  LDA         HIRES.CLR
000141                  LDA         PG2.CLR             ; & SELECT PAGE 1
000142  *
000143                  LDA         #$02
000144                  BIT         SCRNMODE
000145                  BVS         SD015               ; IF GRAPHICS MODE THEN KEEP 40 COL MODE
000146                  BEQ         SD015               ; IF 40 COL MODE THEN KEEP
000147                  LDA         MIX.CLR+1           ; ELSE SWITCH TO 80 COL DISPLAY MODE
000148  *
000149                  LDX         #MSGLEN+1           ; ENSURE BKGRND SET TO INVERSE SPACES
000150                  LDA         #$20                ; SPACE CHAR W/INVERSE
000151  SD010           STA         MSGBASE2-1,X
000152                  DEX
000153                  BPL         SD010
000154  *
000155  SD015           LDX         #0                  ; MOVE MSG TO TEXT SCREEN
000156  SD020           LDA         MSG,X
000157                  STA         MSGBASE-1,X
```



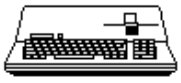
```
000158                  INX
000159                  CPX         #MSGLEN
000160                  BNE         SD020
000161  *
000162                  LDA         SDEATH.REGS+A.SAVE   ; DISPLAY ERROR CODE (2 HEX DIGITS)
000163                  CLC
000164                  LSR         A
000165                  LSR         A
000166                  LSR         A
000167                  LSR         A
000168                  JSR         PRINT                ; FIRST DIGIT
000169                  INX
000170                  LDA         SDEATH.REGS+A.SAVE
000171                  AND         #$0F
000172                  JSR         PRINT                ; SECOND DIGIT
000173  *
000174                  LDA         #>SD100
000175                  STA         NMI.VECTOR
000176                  LDA         #<SD100
000177                  STA         NMI.VECTOR+1
000178  *
000179  *
000180                  JMP         *                    ; HANG UNTIL REBOOT (CTRL/RESET)
000181                  REP         60
000182  SD100           RTI                              ; NMI VECTOR POINT HERE TO MASK THEM OUT
000183  *
000184  *
000185  * PRINT SUBROUTINE
000186  *
000187  PRINT           EQU         *
000188                  CMP         #$A
000189                  BCS         PRNT100
000190                  ADC         #$30                 ; "0"-"9"
000191                  BCC         PRNT110              ; ALWAYS TAKEN
000192  PRNT100         ADC         #$36                 ; "A"-"F"
000193  PRNT110         STA         MSGBASE-1,X
000194                  RTS
000195  *
000196                  LST         ON
000197  ZZEND           EQU         *
000198  ZZLEN           EQU         ZZEND-ZZORG
000199                  IFNE        ZZLEN-LENSERR
000200                  FAIL        2,"SOSORG            FILE IS INCORRECT FOR SYSERR"
000201                  FIN
000202
000203  ********************************************************************************
000204  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SYSERR.SRC
000205  ********************************************************************************
000206

End of File -- Lines: 206 Characters: 5264
```



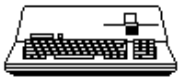
```
================================================================================
FILE: "SOS.SYSGLOB.SRC.TEXT"
================================================================================

000001  ***************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: SYSGLOB.SRC
000003  ***************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1  GLOBAL EQUATES"
000007                  REL
000008                  ORG        $18FC
000009                  MSB        OFF
000010                  REP        60
000011  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000012  *                    ALL RIGHTS RESERVED
000013                  REP        60
000014  *
000015  *  SOS SYSTEM GLOBAL DATA & EQUATES
000016  *
000017  *  THIS MODULE CONTAINS THE SOS JUMP TABLE, AND ALL GLOBAL
000018  *  DATA AND EQUATES.  THE JUMP TABLE, AND ALL DATA THAT IS
000019  *  TO BE REFERENCED BY DEVICE HANDLERS, ARE ASSIGNED FIXED
000020  *  ADDRESSES AT THE BEGINNING OF MEMORY PAGE $19.  DATA
000021  *  THAT IS ONLY REFERENCED BY SOS BEGINS $1980, BUT MAY BE
000022  *  MOVED WHENEVER SOS IS RELINKED.
000023  *
000024                  REP        60
000025  *
000026                  EXTRN      ALLOCSIR
000027                  EXTRN      DEALCSIR
000028                  EXTRN      NMIDSBL
000029                  EXTRN      NMIENBL
000030                  EXTRN      QUEEVENT
000031                  EXTRN      SELC800
000032                  EXTRN      SYSDEATH
000033                  EXTRN      SYSERR
000034                  EXTRN      REQBUF
000035                  EXTRN      GETBUFADR
000036                  EXTRN      RELBUF
000037                  EXTRN      NMIDBUG
000038                  EXTRN      NMICONT
000039                  EXTRN      COLDSTRT
000040  *
000041  *
000042                  ENTRY      MEMSIZE
000043                  ENTRY      SYSBANK
000044                  ENTRY      SUSPFLSH
000045                  ENTRY      NMIFLAG
000046                  ENTRY      SCRNMODE
000047                  ENTRY      GRSIZE
000048  *
000049                  ENTRY      SERR
000050                  ENTRY      DBUGBRK
000051                  ENTRY      KYBDNMI
000052                  ENTRY      NMISPSV
000053                  ENTRY      SDEATH.REGS
000054  *
000055                  ENTRY      SOSVER
000056                  ENTRY      SOSVERL
000057  *
000058                  ENTRY      SZPAGE
000059                  ENTRY      SXPAGE
000060                  ENTRY      SSPAGE
000061  *
000062                  ENTRY      CZPAGE
000063                  ENTRY      CXPAGE
000064                  ENTRY      CSPAGE
000065                  ENTRY      CEVPRI
000066  *
000067                  ENTRY      SIRTEMP
000068                  ENTRY      SIRARGSIZ
000069                  ENTRY      IRQCNTR
000070                  ENTRY      NMICNTR
000071                  ENTRY      QEVTEMP
000072                  ENTRY      QEV.THIS
000073                  ENTRY      QEV.LAST
000074  *
000075                  ENTRY      BADBRK
000076                  ENTRY      BADINT1
```



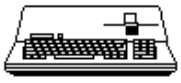
```
000077                  ENTRY     BADINT2
000078                  ENTRY     NMIHANG
000079                  ENTRY     EVQOVFL
000080                  ENTRY     STKOVFL
000081                  ENTRY     BADSYSCALL
000082                  ENTRY     DEV.OVFLOW
000083                  ENTRY     MEM2SML
000084                  ENTRY     VCBERR
000085                  ENTRY     FCBERR
000086                  ENTRY     ALCERR
000087                  ENTRY     DIRERR
000088                  ENTRY     TOOLONG
000089                  ENTRY     BADBUFNUM
000090                  ENTRY     BADBUFSIZ
000091                  ENTRY     BITMAPADR
000092  *
000093                  ENTRY     BADSCNUM
000094                  ENTRY     BADCZPAGE
000095                  ENTRY     BADXBYTE
000096                  ENTRY     BADSCPCNT
000097                  ENTRY     BADSCBNDS
000098  *
000099                  ENTRY     NODNAME
000100                  ENTRY     BADDNUM
000101  *
000102                  ENTRY     BADPATH
000103                  ENTRY     CFCBFULL
000104                  ENTRY     FCBFULL
000105                  ENTRY     BADREFNUM
000106                  ENTRY     PATHNOTFND
000107                  ENTRY     VNFERR
000108                  ENTRY     FNFERR
000109                  ENTRY     DUPERR
000110                  ENTRY     OVRERR
000111                  ENTRY     DIRFULL
000112                  ENTRY     CPTERR
000113                  ENTRY     TYPERR
000114                  ENTRY     EOFERR
000115                  ENTRY     POSNERR
000116                  ENTRY     ACCSERR
000117                  ENTRY     BTSERR
000118                  ENTRY     FILBUSY
000119                  ENTRY     NOTSOS
000120                  ENTRY     BADLSTCNT
000121                  ENTRY     OUTOFMEM
000122                  ENTRY     BUFTBLFULL
000123                  ENTRY     BADSYSBUF
000124                  ENTRY     DUPVOL
000125                  ENTRY     NOTBLKDEV
000126                  ENTRY     LVLERR
000127  *
000128                  ENTRY     BADJMODE
000129  *
000130                  ENTRY     BADBKPG
000131                  ENTRY     SEGRQDN
000132                  ENTRY     SEGTBLFULL
000133                  ENTRY     BADSEGNUM
000134                  ENTRY     SEGNOTFND
000135                  ENTRY     BADSRCHMODE
000136                  ENTRY     BADCHGMODE
000137                  ENTRY     BADPGCNT
000138  *
000139                  ENTRY     XREQCODE
000140                  ENTRY     XCTLCODE
000141                  ENTRY     XCTLPARM
000142                  ENTRY     XNOTOPEN
000143                  ENTRY     XNOTAVAIL
000144                  ENTRY     XNORESRC
000145                  ENTRY     XBADOP
000146                  ENTRY     XIOERROR
000147                  ENTRY     XNODRIVE
000148                  ENTRY     XNOWRITE
000149                  ENTRY     XBYTECNT
000150                  ENTRY     XBLKNUM
000151                  ENTRY     XDISKSW
000152                  ENTRY     BACKMASK            ; MASK BYTE FOR BACKUP BIT.
000153  *
000154                  ENTRY     E1908               ; DISK DRIVER IS READING/WRITING (SET) ELSE NOT (RESET)
000155  *
000156                  PAGE
000157                  DW        SYSGLOB             ;SYSGLOB TARGET ADDRESS
```



```
000158                  DW          $0100               ;  AND LENGTH
000159  *
000160  *  SYSTEM GLOBAL DATA
000161  *     (ACCESSIBLE TO SOS AND DEVICE HANDLERS)
000162  *
000163  SYSGLOB         EQU         *
000164  *
000165  MEMSIZE         DFB         $08                 ;MEMORY SIZE = 128K
000166  SYSBANK         DFB         $02                 ;SYSTEM BANK = 2
000167  SUSPFLSH        DFB         $00                 ;SYSOUT SUSPEND/FLUSH FLAG
000168  NMIFLAG         DFB         $00                 ;NMI PENDING FLAG
000169                  DW          NMIEXIT             ;DEFAULT NMI VECTOR
000170  SCRNMODE        DFB         $80                 ;CURRENT SCREEN MODE
000171  GRSIZE          DFB         $00
000172  *
000173  *
000174  *  SOS JUMP TABLE
000175  *
000176                  DS          SYSGLOB+$10-*,$00   ; USED BY THE MOUSE DRIVER
000177  USERNMI         JMP         NMIEXIT             ;KEYBOARD NMI VECTOR
000178                  JMP         ALLOCSIR            ;ALLOCATE A SIR
000179                  JMP         DEALCSIR            ;DEALLOCATE A SIR
000180                  JMP         NMIDSBL             ;DISABLE NMI
000181                  JMP         NMIENBL             ;ENABLE NMI
000182                  JMP         QUEEVENT            ;QUEUE AN EVENT
000183                  JMP         SELC800             ;SELECT I/O EXPANSION ROM
000184                  JMP         SYSDEATH            ;SYSTEM DEATH
000185                  JMP         SYSERR              ;SOS ERROR
000186                  JMP         REQBUF              ;REQUEST BUFFER
000187                  JMP         GETBUFADR           ;GET BUFFER'S ADDRESS
000188                  JMP         RELBUF              ;RELEASE BUFFER
000189                  JMP         CLRBMASK            ;VECTOR TO CLRBMASK
000190                  PAGE
000191  *
000192  *  SOS DATA AND EQUATES
000193  *     (ACCESSIBLE ONLY TO SOS)
000194  *
000195                  DS          SYSGLOB+$80-*,$00
000196  SERR            DFB         $00                 ;SYSTEM ERROR CODE
000197  *
000198  DBUGBRK         NOP                             ;TO ENABLE DEBUG BREAK POINTS,
000199                  PLA                             ;  PATCH THESE BYTES TO
000200                  PLA                             ;  JMP TO THE DEBUGGER
000201                  RTS
000202  *
000203  KYBDNMI         JMP         USERNMI
000204                  JMP         NMIDBUG
000205  NMISPSV         DFB         0
000206                  JMP         NMICONT
000207  NMIEXIT         RTS
000208  *
000209  *
000210  SOSVER          ASC         "SOS 1.3   01-DEC-82"
000211  SOSVERL         EQU         *-SOSVER
000212  *
000213                  ASC         "(C) 1980, 1981 BY APPLE COMPUTER INC."
000214  *
000215  E1908           EQU         $1908               ; ALLOCATED TO STEPHEN SMITH (MOUSE DRIVER)
000216  * ABOVE SET AND RESET IN DISK DRIVER
000217  SZPAGE          EQU         $1800               ;SYSTEM ZERO PAGE
000218  SXPAGE          EQU         $1400               ;SYSTEM EXTEND PAGE
000219  SSPAGE          EQU         $0100               ;SYSTEM STACK PAGE
000220  *
000221  CZPAGE          EQU         $1A00               ;CALLER'S ZERO PAGE
000222  CXPAGE          EQU         $1600               ;CALLER'S EXTEND PAGE
000223  CSPAGE          EQU         $1B00               ;CALLER'S STACK PAGE
000224  CEVPRI          DFB         $00                 ;CALLER'S EVENT PRIORITY
000225  *
000226  SIRTEMP         DFB         $00                 ;TEMP FOR ALLOCSIR & DEALCSIR
000227  SIRARGSIZ       DFB         $00                 ;ARGUMENT COUNT FOR ALLOCSIR & DEALCSIR
000228  IRQCNTR         DW          $0000               ;FALSE IRQ COUNTER
000229  NMICNTR         DW          $0000               ;COUNTER FOR NMILOCK
000230  QEVTEMP         DFB         $00                 ;TEMP FOR QUEEVENT
000231  QEV.THIS        DFB         $00                 ;POINTER FOR QUEEVENT
000232  QEV.LAST        DFB         $00                 ;POINTER FOR QUEEVENT
000233  *
000234  SOSQUIT         DS          COLDSTRT
000235  BACKMASK        DFB         BACKBIT             ; MASK USED BY BFM TO UPDATE BACKUP BIT
000236  *
000237  * TO CLEAR THE BACKUP BIT, A PROGRAM MUST JSR TO CLRBMASK THRU 1934 THEN DO A
000238  * SET-FILE-INFO WITH NO INTERVENING SOS CALLS.  ANY SOS CALL WILL
```

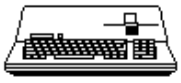


```
000239  * SET BACKMASK AGAIN.  THIS FEATURE IS INTENTIONALLY LEFT UNDOCUMENTED.
000240  *
000241  CLRBMASK        AND     #BACKBIT            ; PURIFY
000242                  STA     BACKMASK            ; SET THE MASK
000243                  RTS                         ; AND BACK TO THE CALLER
000244                  PAGE
000245  *
000246  *  SYSTEM DEATH REGISTER SAVE AREA
000247  * (SYSTEM STACK MOVED TO $1700-$17FF)
000248  *
000249                  DS      SYSGLOB+$F6-*,$00
000250  SDEATH.REGS     EQU     *
000251                  DFB     $00                 ;BANK
000252                  DFB     $00                 ;ZERO PAGE
000253                  DFB     $00                 ;ENVIRONMENT
000254                  DFB     $00                 ;Y
000255                  DFB     $00                 ;X
000256                  DFB     $00                 ;A
000257                  DFB     $00                 ;STATUS
000258                  DW      $00                 ;PROGRAM COUNTER
000259                  DFB     $00                 ;STACK POINTER
000260  *
000261  *  SYSTEM DEATH ERROR NUMBERS
000262  *
000263  BADBRK          EQU     $01                 ;BRK FROM SOS
000264  BADINT1         EQU     $02                 ;INTERRUPT NOT FOUND
000265  BADINT2         EQU     $03                 ;BAD ZERO PAGE ALLOCATION
000266  NMIHANG         EQU     $04                 ;UNABLE TO LOCK NMI
000267  EVQOVFL         EQU     $05                 ;EVENT QUEUE OVERFLOW
000268  STKOVFL         EQU     $06                 ;STACK OVERFLOW
000269  *
000270  BADSYSCALL      EQU     $07                 ;DMGR DETECTED INVALID REQUEST CODE
000271  DEV.OVFLOW      EQU     $08                 ;DMGR - TOO MANY DEVICE HANDLERS
000272  MEM2SML         EQU     $09                 ;MEMORY SIZE < 64K
000273  VCBERR          EQU     $0A                 ;VOLUME CONTROL BLOCK NOT USABLE (BFMGR)
000274  FCBERR          EQU     $0B                 ;FILE CONTROL BLOCK CRASHED
000275  ALCERR          EQU     $0C                 ;ALLOCATION BLOCKS INVALID
000276  TOOLONG         EQU     $0E                 ;PATHNAME BUFFER OVERFLOW
000277  BADBUFNUM       EQU     $0F                 ;INVALID BUFFER NUMBER
000278  BADBUFSIZ       EQU     $10                 ;INVALID BUFFER SIZE (=0 OR >16K)
000279                  PAGE
000280  *
000281  *  SYSTEM ERROR NUMBERS
000282  *
000283  * - SYSTEM CALL MANAGER
000284  *
000285  BADSCNUM        EQU     $01                 ;BAD SYSTEM CALL NUMBER
000286  BADCZPAGE       EQU     $02                 ;BAD CALLER'S ZPAGE (MUST=$1A)
000287  BADXBYTE        EQU     $03                 ;BITS         6..4 <> 0
000288  BADSCPCNT       EQU     $04                 ;BAD SYSTEM CALL PARM COUNT
000289  BADSCBNDS       EQU     $05                 ;SYS CALL PARM ADR
000290  *
000291  * - DEVICE MANAGER
000292  *
000293  NODNAME         EQU     $10                 ;DEVICE NAME NOT FOUND
000294  BADDNUM         EQU     $11                 ;INVALID DEV.NUM PARM
000295  *
000296  * - DEVICE HANDLERS (STANDARD ERRORS)
000297  *
000298  XREQCODE        EQU     $20                 ;INVALID REQUEST CODE
000299  XCTLCODE        EQU     $21                 ;INVALID CONTROL/STATUS CODE
000300  XCTLPARM        EQU     $22                 ;INVALID CONTROL/STATUS PARM
000301  XNOTOPEN        EQU     $23                 ;DEVICE NOT OPEN
000302  XNOTAVAIL       EQU     $24                 ;DEVICE NOT AVAILABLE
000303  XNORESRC        EQU     $25                 ;UNABLE TO OBTAIN RESOURCE
000304  XBADOP          EQU     $26                 ;INVALID OPERATION
000305  XIOERROR        EQU     $27                 ;I/O ERROR
000306  *
000307  XNODRIVE        EQU     $28                 ;NO DRIVE CONNECTED
000308  XNOWRITE        EQU     $2B                 ;DEVICE WRITE PROTECTED
000309  XBYTECNT        EQU     $2C                 ;BYTE COUNT <> A MULTIPLE OF 512
000310  XBLKNUM         EQU     $2D                 ;BLOCK NUMBER TOO LARGE
000311  XDISKSW         EQU     $2E                 ;DISK MEDIA HAS BEEN SWITCHED
000312  *
000313  * - NOTE: ERROR CODES $30-$3F HAVE BEEN RESERVED FOR DEVICE
000314  *   HANDLER SPECIFIC ERRORS
000315  *
000316  *
000317  * - FILE MANAGER
000318  *
000319  BADPATH         EQU     $40                 ;PATHNAME, INVALID SYNTAX
```

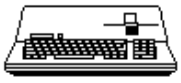


```
000320  CFCBFULL        EQU       $41               ;CHAR FILE CTRL BLOCK TABLE FULL
000321  FCBFULL         EQU       $42               ;BLOCK FILE CTRL BLOCK TABLE FULL
000322  BADREFNUM       EQU       $43               ;INVALID REF.NUM PARM
000323  PATHNOTFND      EQU       $44               ;PATHNAME NOT FOUND
000324  VNFERR          EQU       $45               ;VOLUME NOT FOUND
000325  FNFERR          EQU       $46               ;FILE NOT FOUND
000326  DUPERR          EQU       $47               ;DUPLICATE FILE NAME ERROR
000327  OVRERR          EQU       $48               ;NOT ENOUGH DISK SPACE FOR PREALLOCATION
000328  DIRFULL         EQU       $49               ;DIRECTORY FULL ERROR
000329  CPTERR          EQU       $4A               ;FILE INCOMPATIBLE SOS VERSION
000330  TYPERR          EQU       $4B               ;NOT CURRENTLY SUPPORTED FILE TYPE
000331  EOFERR          EQU       $4C               ;POSITION ATTEMPTED BEYOND END OF FILE
000332  POSNERR         EQU       $4D               ;ILLEGAL POSITION (L.T. 0 OR G.T. $FFFFFF)
000333  ACCSERR         EQU       $4E               ;FILE ACCESS R/W REQUEST CONFLICTS WITH ATTRIBUTES
000334  BTSERR          EQU       $4F               ;USER SUPPLIED BUFFER TOO SMALL
000335  FILBUSY         EQU       $50               ;EITHER WRITE WAS REQUESTED OR WRITE ACCESS ALREADY ALLOCATED
000336  DIRERR          EQU       $51               ;DIRECTORY ERROR
000337  NOTSOS          EQU       $52               ;NOT A SOS DISKETTE
000338  BADLSTCNT       EQU       $53               ;INVALID VALUE IN LIST PARAMETER
000339  OUTOFMEM        EQU       $54               ;OUT OF FREE MEMORY FOR BUFFER
000340  BUFTBLFULL      EQU       $55               ;BUFFER TABLE FULL
000341  BADSYSBUF       EQU       $56               ;INVALID SYSBUF PARAMETER
000342  DUPVOL          EQU       $57               ;SON A BITCH GOT TWO VOLUMES OF SAME ROOT NAME!!!
000343  NOTBLKDEV       EQU       $58
000344  LVLERR          EQU       $59               ;INVALID FILE LEVEL
000345  BITMAPADR       EQU       $5A
000346  BACKBIT         EQU       $20               ; MASK FOR BACKUP BIT
000347  *
000348  * - UTILITY MANAGER
000349  *
000350  BADJMODE        EQU       $70               ;INVALID JOYSTICK REQUEST
000351  *
000352  * - MEMORY MANAGER
000353  *
000354  BADBKPG         EQU       $E0               ;INVALID BANK/PAGE PAIR
000355  SEGRQDN         EQU       $E1               ;SEGMENT REQUEST DENIED
000356  SEGTBLFULL      EQU       $E2               ;SEGMENT TABLE FULL
000357  BADSEGNUM       EQU       $E3               ;INVALID SEGMENT NUMBER
000358  SEGNOTFND       EQU       $E4               ;SEGMENT NOT FOUND
000359  BADSRCHMODE     EQU       $E5               ;INVALID SEARCH MODE PARM
000360  BADCHGMODE      EQU       $E6               ;INVALID CHANGE MODE PARM
000361  BADPGCNT        EQU       $E7               ;INVALID PAGE COUNT PARM
000362                  ORG       SYSGLOB+$100
000363                  DW        $B800             ;KERNEL TARGET ADDRESS
000364                  DW        $47C0             ;  AND LENGTH
000365
000366  ************************************************************************
000367  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SYSGLOB.SRC
000368  ************************************************************************
000369
000370
```

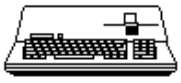
End of File -- Lines: 370 Characters: 12524



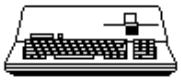
```
================================================================================
FILE: "SOS.TCOMP.SOS.TEXT"
================================================================================

000001  ****************************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: TCOMP.SOS
000003  ****************************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006  :TABS 17,23,40
000007  SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
000008  SL4:DR1:ASM UMGR.SRC,UMGR.OBJ,6,1
000009  SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
000010  SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
000011  SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
000012  SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
000013  SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
000014  SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
000015  SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
000016  SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
000017
000018  ****************************************************************************
000019  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: TCOMP.SOS
000020  ****************************************************************************

End of File -- Lines: 20 Characters: 824
```



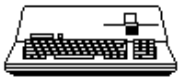
```
================================================================================
FILE: "SOS.UMGR.SRC.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: UMGR.SRC
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  SBTL       "SOS 1.1  UTILITY MANAGER"
000007                  REL
000008                  INCLUDE    SOSORG,6,1,254
000009                  ORG        ORGUMGR
000010  ZZORG           EQU        *
000011                  MSB        OFF
000012                  REP        60
000013  *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
000014  *                      ALL RIGHTS RESERVED
000015                  REP        60
000016  *  UTILITY MANAGER
000017  *
000018  *  THIS MODULE HANDLES THE FOLLOWING SOS CALLS:
000019  *    SET.FENCE,   GET.FENCE
000020  *    SET.TIME,    GET.TIME
000021  *    JOYSTICK,    COLDSTRT
000022  *
000023  *  IN ADDITION, IT CONTAINS THE ROUITNE DATETIME WHICH
000024  *  PROVIDES THE DATE AND TIME FOR THE BLOCK FILE MANAGER.
000025  *
000026                  REP        60
000027  *
000028                  ENTRY      UMGR
000029                  ENTRY      DATETIME
000030                  ENTRY      BCDBIN
000031                  ENTRY      COLDSTRT
000032  *
000033                  ENTRY      PCLOCK
000034  *
000035                  EXTRN      SYSBANK
000036                  EXTRN      CEVPRI
000037                  EXTRN      SYSERR
000038                  EXTRN      BADSCNUM
000039                  EXTRN      BADJMODE
000040                  EXTRN      XNORESRC
000041                  EXTRN      ALLOCSIR
000042                  EXTRN      DEALCSIR
000043  *
000044  U.TPARMX        EQU        $C0
000045  U.REQCODE       EQU        U.TPARMX
000046  PRIORITY        EQU        U.TPARMX+1
000047  J.MODE          EQU        U.TPARMX+1
000048  J.VALUE         EQU        U.TPARMX+2
000049  TIME            EQU        U.TPARMX+1
000050  MEMORY          EQU        U.TPARMX+1
000051  *
000052  BITON2          EQU        $04
000053  BITON5          EQU        $20
000054  BITON6          EQU        $40
000055  BITON7          EQU        $80
000056  BITOFF5         EQU        $DF
000057  *
000058  Z.REG           EQU        $FFD0
000059  E.REG           EQU        $FFDF
000060  B.REG           EQU        $FFEF
000061                  PAGE
000062                  REP        35
000063  *
000064  * UTILITY SWITCH
000065  *
000066                  REP        35
000067  *
000068  *
000069  UMGR            EQU        *
000070                  LDA        E.REG               ;SELECT $C000 I/O SPACE
000071                  ORA        #BITON6
000072                  STA        E.REG
000073  *
000074                  LDA        U.REQCODE
000075                  CMP        #USWCNT
000076                  BCS        UMGRERR
```



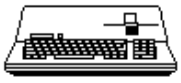
```
000077                  ASL         A
000078                  TAX
000079                  LDA         USWTBL+1,X
000080                  PHA
000081                  LDA         USWTBL,X
000082                  PHA
000083                  RTS
000084  *
000085  UMGRERR         LDA         #>BADSCNUM
000086                  JSR         SYSERR
000087  *
000088  *  UTILITY SWITCH TABLE
000089  *
000090  USWTBL          EQU         *
000091                  DW          SET.FENCE-1
000092                  DW          GET.FENCE-1
000093                  DW          SET.TIME-1
000094                  DW          GET.TIME-1
000095                  DW          JOYSTICK-1
000096                  DW          COLDSTRT-1
000097  USWCNT          EQU         *-USWTBL/2
000098                  PAGE
000099                  REP         60
000100  *
000101  * SET.FENCE(IN.PRIORITY) SYSTEM CALL
000102  *
000103  * GET.FENCE(OUT.PRIORITY) SYSTEM CALL
000104  *
000105  * THESE TWO CALLS ALLOW THE CALLER TO EITHER RETRIEVE OR SET
000106  * THE CURRENT SYSTEM EVENT PRIORITY THRESHOLD.  BY RAISING
000107  * THE FENCE, A USER MAY INHIBIT THE EXECUTION OF EVENTS WHOSE
000108  * PRIORITY IS EQUAL TO OR LESS THAN THE VALUE OF THE SYSTEM
000109  * FENCE.
000110  *
000111                  REP         60
000112  *
000113  *
000114  SET.FENCE       EQU         *
000115                  LDA         PRIORITY
000116                  STA         CEVPRI
000117                  RTS                             ; NORMAL EXIT
000118  *
000119  *
000120  GET.FENCE       EQU         *
000121                  LDA         CEVPRI
000122                  LDY         #0
000123                  STA         (PRIORITY),Y
000124                  RTS                             ; NORMAL EXIT
000125                  PAGE
000126                  REP         60
000127  *
000128  *  SET.TIME(IN.TIME)
000129  *  GET.TIME(OUT.TIME)
000130  *
000131  *  THESE SYSTEM CALLS ALLOW THE USER TO SET AND READ THE
000132  *  SYSTEM'S CLOCK.  THE TIME IS EXPRESSED AS AN EIGHTEEN
000133  *  DIGIT ASCII STRING IN THE FORM "YYYYMMDDWHHMMSSMMM".
000134  *
000135  *    YYYY  YEAR       [1900-1999]
000136  *      MM  MONTH        [01-12]
000137  *      DD  DAY          [01-31]
000138  *       W  WEEKDAY       [1-7]  1 => SUNDAY
000139  *      HH  HOUR         [00-23]
000140  *      MM  MINUTE       [00-59]
000141  *      SS  SECOND       [00-59]
000142  *     MMM  MILLISECOND [000-999]
000143  *
000144  *  THE CLOCK CHIP AUTOMATICALLY MAINTAINS THE TIME AND
000145  *  DATE FROM MILLISECONDS TO MONTHS.  IT DOES NOT MAINTAIN
000146  *  THE YEAR, HOWEVER, NOR DOES IT RECOGNIZE 29 FEBRUARY
000147  *  IN LEAP YEARS.  THE SOFTWARE SETS THE DAY AND MONTH
000148  *  LATCHES TO THE DON'T CARE STATE AND USES THE REMAINING
000149  *  EIGHT BITS TO HOLD A TWO DIGIT BCD YEAR.  THE CLOCK
000150  *  MUST BE RESET AT THE BEGINNING OF EACH YEAR AND ON
000151  *  29 FEBRUARY IN LEAP YEARS.
000152  *
000153  *  SET.TIME ASSUMES THAT THE DATE IS VALID AND CORRECT.
000154  *  THE CENTURY IS IGNORED AND MILLISECONDS ARE ALWAYS SET
000155  *  TO ZERO.  GET.TIME ALWAYS SETS THE CENTURY TO 19.
000156  *
000157                  REP         60
```



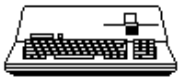
```
000158  *
000159  *
000160  *  TEMPORARY ZERO PAGE
000161  *
000162  PCLK            EQU       $D0                 ;POINTER TO SAVED PCLOCK
000163  WKDAY           EQU       $D2
000164  CKSUM           EQU       $D3
000165  CLKTEMP         EQU       $18D4               ;THROUGH $18DD - ABSOLUTE
000166  *
000167  *  CLOCK LOCAL DATA
000168  *
000169  PCLOCK          DS        $0A                 ;PSEUDO CLOCK REGISTERS
000170  RETRY           DS        $01
000171  *
000172  *  CLOCK HARDWARE ADDRESSES
000173  *
000174  CLOCK           EQU       $C070
000175  CSEC            EQU       $02
000176  CMIN            EQU       $03
000177  CMON            EQU       $07
000178  LDAY            EQU       $0E
000179  CRESET          EQU       $12
000180  STATUS          EQU       $14
000181  *
000182  WKMON           DFB       8,11,11,7,9,12
000183                  DFB       7,10,13,8,11,13
000184  *
000185  *
000186  SET.TIME        EQU       *
000187                  LDX       #$00
000188                  LDY       #$12
000189                  LDA       #'0'
000190                  BNE       STIM011
000191  *
000192  STIM010         INX
000193                  LDA       (TIME),Y            ;CONVERT TIME FROM
000194  STIM011         AND       #$0F                ;  ASCII TO BCD AND
000195                  STA       PCLOCK,X            ;  TRANSFER TO PCLOCK
000196                  DEY
000197                  CPY       #$07
000198                  BEQ       STIM010
000199                  LDA       (TIME),Y
000200                  ASL       A
000201                  ASL       A
000202                  ASL       A
000203                  ASL       A
000204                  ORA       PCLOCK,X
000205                  STA       PCLOCK,X
000206                  DEY
000207                  BPL       STIM010
000208  *
000209                  LDA       PCLOCK+7            ;CALCULATE WEEKDAY
000210                  JSR       BCDBIN
000211                  TAX
000212                  LDA       PCLOCK+8
000213                  JSR       BCDBIN
000214                  TAY
000215                  LSR       A
000216                  LSR       A
000217                  STA       WKDAY
000218                  TYA
000219                  AND       #$03
000220                  BNE       STIM015
000221                  CPX       #3
000222                  BCS       STIM015             ; <SRS 82.162>
000223                  DEY
000224  STIM015         CLC
000225                  TYA
000226                  ADC       WKDAY
000227                  ADC       WKMON-1,X
000228                  STA       WKDAY
000229                  LDA       PCLOCK+6
000230                  JSR       BCDBIN
000231                  CLC
000232                  ADC       WKDAY
000233                  SEC
000234  STIM016         SBC       #7
000235                  CMP       #8
000236                  BCS       STIM016
000237                  STA       PCLOCK+5
000238  *
```



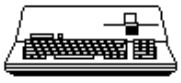
```
000239                  LDA     #$D0
000240                  STA     PCLK                ;POINT (PCLK) TO 8F:FFD0
000241                  LDA     #$FF
000242                  STA     PCLK+1
000243                  LDA     #$8F
000244                  STA     $1401+PCLK
000245                  LDA     #$A5
000246                  STA     CKSUM               ;INITIALIZE CHECKSUM
000247                  LDY     #$00
000248  *
000249  STIM020         LDA     PCLOCK,Y            ;SAVE PCLOCK
000250                  STA     (PCLK),Y            ;  BEHIND 6522
000251                  EOR     CKSUM
000252                  STA     CKSUM
000253                  INY
000254                  CPY     #$0A
000255                  BCC     STIM020
000256                  STA     (PCLK),Y            ;SAVE CHECKSUM
000257  *
000258                  LDA     Z.REG
000259                  PHA                         ;SAVE ZERO PAGE
000260                  LDA     E.REG
000261                  PHA                         ;SAVE ENVIRONMENT
000262                  ORA     #BITON7             ;  AND SET 1 MHZ
000263                  STA     E.REG
000264  *
000265                  LDY     #STATUS
000266                  STY     Z.REG
000267                  LDA     CLOCK               ;DOES CLOCK EXIST?
000268                  BMI     STIM050             ;  NO
000269  *
000270                  LDX     #CRESET
000271                  STX     Z.REG
000272                  LDA     #$FF                ;RESET ALL COUNTERS
000273                  STA     CLOCK
000274                  STA     CLOCK
000275  *
000276                  LDX     #CSEC-1
000277  STIM030         INX
000278                  PHP
000279                  SEI                         ;DISABLE INTERRUPTS
000280  STIM040         STX     Z.REG
000281                  LDA     CLOCK               ;(DUMMY READ FOR STATUS)
000282                  LDA     PCLOCK,X
000283                  STA     CLOCK               ;SET CLOCK COUNTER
000284                  LDA     CLOCK               ;(DUMMY READ FOR STATUS)
000285                  STY     Z.REG
000286                  LDA     CLOCK               ;CHECK STATUS BIT
000287                  BNE     STIM040
000288                  PLP                         ;RESTORE INTERRUPTS
000289                  CPX     #CMON
000290                  BCC     STIM030
000291  *
000292                  LDX     #LDAY
000293                  STX     Z.REG
000294                  LDA     PCLOCK+8
000295                  ORA     #$CC                ;STUFF YEAR INTO DAY
000296                  STA     CLOCK               ;  AND MONTH LATCHES
000297                  INC     Z.REG
000298                  LDA     PCLOCK+8
000299                  LSR     A
000300                  LSR     A
000301                  ORA     #$CC
000302                  STA     CLOCK
000303  *
000304  STIM050         PLA
000305                  STA     E.REG               ;RESTORE ENVIRONMENT
000306                  PLA
000307                  STA     Z.REG               ;  AND ZERO PAGE
000308                  RTS
000309                  PAGE
000310  GET.TIME        EQU     *
000311                  LDA     Z.REG               ;SAVE ZERO PAGE
000312                  PHA
000313                  LDA     E.REG               ;SAVE ENVIRONMENT
000314                  PHA
000315                  ORA     #BITON7
000316                  STA     E.REG               ;SET 1 MHZ
000317  *
000318                  LDY     #STATUS
000319                  STY     Z.REG
```



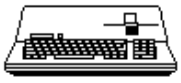
```
000320                  LDA     CLOCK           ;DOES CLOCK EXIST?
000321                  BMI     GTIM050         ;  NO
000322  *
000323                  LDA     #$10            ;ALLOW $10 RETRYS
000324                  STA     RETRY
000325  GTIM010         LDX     #CMON+1
000326                  PHP
000327                  SEI                     ;DISABLE INTERRUPTS
000328  *
000329  GTIM020         DEX
000330                  BMI     GTIM030         ;ALL DONE
000331                  STX     Z.REG
000332                  LDA     CLOCK           ;COPY CLOCK COUNTERS
000333                  STA     CLKTEMP,X       ;  TO TEMP REGISTERS
000334                  STY     Z.REG
000335                  LDA     CLOCK           ;CHECK STATUS BIT
000336                  BEQ     GTIM020
000337  *
000338                  PLP                     ;CLOCK READ ERROR
000339                  DEC     RETRY
000340                  BPL     GTIM010         ;TRY AGAIN
000341                  BMI     GTIM050
000342  *
000343  GTIM030         PLP                     ;RESTORE INTERRUPTS
000344                  LDX     #LDAY+1
000345                  STX     Z.REG
000346                  LDA     CLOCK           ;READ YEAR FROM DAY
000347                  SEC                     ;  AND MONTH LATCHES
000348                  ROL     A
000349                  ROL     A
000350                  DEC     Z.REG
000351                  AND     CLOCK
000352                  STA     CLKTEMP+8
000353  *
000354                  LDX     #$09
000355  GTIM040         LDA     CLKTEMP,X       ;COPY CLOCK DATA
000356                  STA     PCLOCK,X        ;  TO PSEUDO CLOCK
000357                  DEX
000358                  BPL     GTIM040
000359  *
000360  GTIM050         LDA     #$19
000361                  STA     PCLOCK+9
000362  *
000363                  PLA
000364                  STA     E.REG           ;RESTORE ENVIRONMENT
000365                  PLA
000366                  STA     Z.REG           ;  AND ZERO PAGE
000367  *
000368                  LDY     #$11
000369                  LDX     #$00
000370  GTIM060         LDA     PCLOCK,X        ;GET MOST SIGNIFICANT
000371                  LSR     A               ;  BCD DIGIT
000372                  LSR     A
000373                  LSR     A
000374                  LSR     A
000375                  ORA     #$30            ;CONVERT TO ASCII
000376                  STA     (TIME),Y
000377                  INX
000378                  DEY
000379                  BMI     GTIM080
000380  GTIM070         LDA     PCLOCK,X        ;GET LEAST SIGNIFICANT
000381                  AND     #$0F            ;  BCD DIGIT
000382                  ORA     #$30            ;CONVERT TO ASCII
000383                  STA     (TIME),Y
000384                  DEY
000385                  CPY     #$07
000386                  BNE     GTIM060
000387                  INX
000388                  BNE     GTIM070
000389  GTIM080         RTS
000390                  PAGE
000391                  REP     60
000392  *
000393  *  SUBROUTINE DATETIME
000394  *
000395  *  THIS SUBROUTINE READS THE CLOCK AND WRITES A DATE/TIME
000396  *  STAMP TO A FOUR BYTE BUFFER ON THE CALLER'S ZERO PAGE;
000397  *  THE DATA FORMAT IS SHOWN BELOW.  ON ENTRY, X MUST POINT
000398  *  TO THE BUFFER.  ON EXIT, ALL REGISTERS ARE CLOBBERED.
000399  *  IF AN ERROR OCCURS, CARRY IS SET AND THE BUFFER IS
000400  *  SET TO ZERO; OTHERWISE, CARRY IS CLEARED.
```



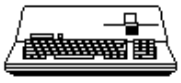
```
000401  *
000402  *   BITS: 7 6 5 4 3 2 1 0
000403  *    X+0  M M M D D D D D
000404  *    X+1  Y Y Y Y Y Y Y M
000405  *    X+2   -  MINUTE   -
000406  *    X+3   - - HOUR  - -
000407  *
000408                  REP         60
000409  *
000410  *  TEMPORARY STORAGE
000411  *
000412  OFFSET          DFB         0
000413  ERRCNT          DFB         0
000414  CLKREGS         DS          5
000415  MIN             EQU         CLKREGS+0
000416  HOUR            EQU         CLKREGS+1
000417  DAY             EQU         CLKREGS+3
000418  MON             EQU         CLKREGS+4
000419  YEAR            EQU         CLKREGS+2
000420  *
000421  *
000422  DATETIME        EQU         *
000423                  STX         OFFSET
000424                  LDA         Z.REG
000425                  PHA                             ;SAVE ZERO PAGE
000426                  LDA         E.REG
000427                  PHA                             ;  AND ENVIRONMENT
000428                  ORA         #BITON7+BITON6      ;SET 1 MHZ AND
000429                  STA         E.REG               ;  ENABLE I/O SPACE
000430  *
000431                  LDY         #STATUS
000432                  STY         Z.REG
000433                  LDA         CLOCK               ;DOES CLOCK EXIST?
000434                  BMI         DT030               ;  NO
000435  *
000436                  LDA         #8
000437                  STA         ERRCNT              ;ALLOW 8 RETRYS
000438  DT010           LDX         #CMON+1
000439                  PHP
000440                  SEI                             ;DISABLE INTERRUPTS
000441  *
000442  DT020           DEX
000443                  CPX         #CMIN
000444                  BCC         DT050
000445                  STX         Z.REG
000446                  LDA         CLOCK               ;READ THE CLOCK
000447                  STA         CLKREGS-CMIN,X
000448                  STY         Z.REG
000449                  LDA         CLOCK               ;CHECK STATUS
000450                  BEQ         DT020
000451  *
000452                  PLP                             ;CLOCK READ ERROR
000453                  DEC         ERRCNT
000454                  BPL         DT010
000455  DT030           PLA
000456                  STA         E.REG               ;RESTORE ENVIRONMENT
000457                  PLA
000458                  STA         Z.REG               ;  AND ZERO PAGE
000459                  LDX         #CMON-CMIN
000460  DT040           LDA         PCLOCK+CMIN,X
000461                  STA         CLKREGS,X
000462                  DEX
000463                  BPL         DT040
000464                  LDX         PCLOCK+8
000465                  JMP         DT060
000466  *
000467  DT050           PLP                             ;READ YEAR FROM LATCHES
000468                  LDA         #LDAY+1
000469                  STA         Z.REG
000470                  LDA         CLOCK
000471                  SEC
000472                  ROL         A
000473                  ROL         A
000474                  DEC         Z.REG
000475                  AND         CLOCK
000476                  TAX
000477  *
000478                  PLA
000479                  STA         E.REG               ;RESTORE ENVIRONMENT
000480                  PLA
000481                  STA         Z.REG               ;  AND ZERO PAGE
```



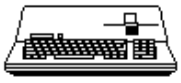
```
000482  *
000483  DT060           TXA
000484                  JSR         BCDBIN              ;CONVERT YEAR TO BINARY
000485                  STA         YEAR
000486                  LDA         MON                 ;CONVERT MONTH AND DAY
000487                  JSR         BCDBIN              ;  TO BINARY THEN
000488                  ASL         A                   ;  COMBINE WITH YEAR
000489                  ASL         A                   ;  TO FORM DATE STAMP
000490                  ASL         A
000491                  ASL         A
000492                  ASL         A
000493                  STA         MON
000494                  ROL         YEAR
000495                  LDA         DAY
000496                  JSR         BCDBIN
000497                  ORA         MON
000498                  LDX         OFFSET
000499                  STA         0,X
000500                  LDA         YEAR
000501                  STA         1,X
000502                  LDA         MIN                 ;CONVERT MINUTE
000503                  JSR         BCDBIN
000504                  STA         2,X
000505                  LDA         HOUR                ;CONVERT HOUR
000506                  JSR         BCDBIN
000507                  STA         3,X
000508                  CLC
000509                  RTS
000510                  PAGE
000511                  REP         60
000512  *
000513  *  SUBROUTINE BCDBIN
000514  *
000515  *  THIS SUBROUTINE CONVERTS A BYTE FROM BCD TO BINARY.
000516  *  THE BYTE IS PASSED AND RETURNED IN A.  THERE IS NO
000517  *  ERROR CHECKING.  Y IS DESTROYED AND X IS UNCHANGED.
000518  *
000519                  REP         60
000520  *
000521  BCDBIN          EQU         *
000522                  PHA
000523                  LSR         A                   ;ISOLATE TENS DIGIT FOR
000524                  LSR         A                   ;  INDEXING THE TABLE
000525                  LSR         A
000526                  LSR         A
000527                  TAY
000528                  PLA
000529                  AND         #$0F                ;GET UNITS
000530                  CLC
000531                  ADC         TENS,Y              ;ADD IN TENS
000532                  RTS
000533  *
000534  TENS            DFB         00,10,20,30,40,50,60,70,80,90
000535                  PAGE
000536                  REP         60
000537  *
000538  *  SOS CALL $64 -- JOYSTICK INPUT
000539  *    JOYSTICK(IN.J.MODE; OUT.J.VALUE)
000540  *
000541                  REP         60
000542  *
000543  *
000544  AD.INPUT        EQU         $D0
000545  AD.TEMP         EQU         $D1
000546  *
000547  PA.SW0          EQU         $C061               ;PORT A, SWITCH 0
000548  PA.SW1          EQU         $C063               ;PORT A, SWITCH 1
000549  PB.SW0          EQU         $C062               ;PORT B, SWITCH 0
000550  PB.SW1          EQU         $C060               ;PORT B, SWITCH 1
000551  *
000552  AD.SEL0         EQU         $C058               ;A/D SELECT CONTROLS
000553  AD.SEL1         EQU         $C05E
000554  AD.SEL2         EQU         $C05A
000555  AD.CHRG         EQU         $C05C               ;A/D RAMP CHARGE /
000556  AD.STRT         EQU         $C05D               ;    START TIMEOUT
000557  AD.FLAG         EQU         $C066               ;A/D TIMEOUT FLAG
000558  *
000559  TCHARGE         EQU         500                 ;CHARGE TIME FOR A/D
000560  TOFFSET         EQU         360                 ;OFFSET TIME TO A/D WINDOW
000561  *
000562  ANALOG          EQU         $F4A8               ;ROM ENTRY FOR ANALOG INPUT
```



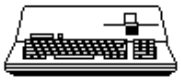
```
000563  ANLOG1          EQU       $F4AB               ;  INTERRUPT REENTRY
000564  D.T2            EQU       $FFD8               ;TIMER
000565  D.ACR           EQU       $FFDB               ;AUXILIARY CONTROL REGISTER
000566  D.IFR           EQU       $FFDD               ;INTERRUPT FLAG REGISTER
000567  *
000568  ENSEL           EQU       $C0DC
000569  ENSIO           EQU       $C0DE
000570  *
000571  *
000572  JOYSTICK        EQU       *
000573                  LDA       J.MODE              ;VALIDATE J.MODE
000574                  CMP       #$08
000575                  BCC       JS010
000576                  LDA       #>BADJMODE
000577  JS.ERR          JSR       SYSERR
000578  *
000579  JS010           JSR       AD.SETUP            ;SET UP RESOURCES
000580                  BCS       JS.ERR
000581                  LDA       J.MODE              ;READ PORT B OR PORT A?
000582                  AND       #BITON2
000583                  BNE       JS020
000584                  LDA       PB.SW0              ;PORT B
000585                  LDX       PB.SW1
000586                  LDY       #$01
000587                  BNE       JS030
000588  JS020           LDA       PA.SW0              ;PORT A
000589                  LDX       PA.SW1
000590                  LDY       #$03
000591  JS030           STY       AD.INPUT            ;SAVE INPUT SELECT
000592                  AND       #BITON7
000593                  BEQ       JS040
000594                  LDA       #$FF
000595  JS040           LDY       #$00
000596                  STA       (J.VALUE),Y         ;RETURN SWITCH 0
000597                  TXA
000598                  AND       #BITON7
000599                  BEQ       JS050
000600                  LDA       #$FF
000601  JS050           INY
000602                  STA       (J.VALUE),Y         ;RETURN SWITCH 1
000603  *
000604                  LSR       J.MODE
000605                  BCC       JS060
000606                  LDA       AD.INPUT
000607                  JSR       AD.READ             ;READ A/D
000608                  LDY       #$02
000609                  STA       (J.VALUE),Y         ;RETURN X AXIS
000610  JS060           INC       AD.INPUT
000611                  LSR       J.MODE
000612                  BCC       JS070
000613                  LDA       AD.INPUT
000614                  JSR       AD.READ             ;READ A/D
000615                  LDY       #$03
000616                  STA       (J.VALUE),Y         ;RETURN Y AXIS
000617  *
000618  JS070           JSR       AD.CLNUP            ;CLEAN UP
000619                  RTS                           ;  AND EXIT
000620                  PAGE
000621                  REP       60
000622  *
000623  *  SUBROUTINE AD.SETUP
000624  *  THIS SUBROUTINE SETS UP THE ENVIRONMENT AND RESOURCES
000625  *  FOR READING THE JOYSTICKS.  IF AN ERROR OCCURS, CARRY
000626  *  IS SET AND AN ERROR NUMBER IS RETURNED IN A.
000627  *  OTHERWISE, CARRY IS CLEARED.
000628  *
000629                  REP       60
000630  AD.SETUP        EQU       *
000631                  LDA       #JOYSIRSIZ
000632                  LDX       #>JOYSIRTBL
000633                  LDY       #<JOYSIRTBL
000634                  JSR       ALLOCSIR            ;ALLOCATE RESOURCES
000635                  BCC       ADS010
000636                  LDA       #>XNORESRC
000637                  RTS
000638  ADS010          LDA       E.REG
000639                  AND       #$7F                ;SET 2 MHZ,
000640                  ORA       #$43                ;  ENABLE ROM, & I/O SPACE
000641                  STA       E.REG
000642                  PHP
000643                  SEI
```



```
000644                  LDA         D.ACR
000645                  AND         #BITOFF5            ;SET UP TIMER
000646                  STA         D.ACR
000647                  PLP
000648                  BIT         ENSEL               ;DISABLE ENSEL
000649                  BIT         ENSIO               ;SET ENSIO FOR INPUT
000650                  RTS
000651  *
000652  JOYSIRTBL       EQU         *
000653                  DFB         $0C,0,0,0,0         ;ENSIO
000654                  DFB         $0D,0,0,0,0         ;ENSEL
000655                  DFB         $0E,0,0,0,0         ;6522 D.T2
000656  JOYSIRSIZ       EQU         *-JOYSIRTBL
000657                  REP         60
000658  *
000659  *  SUBROUTINE AD.CLNUP
000660  *  THIS SUBROUTINE RESTORES THE ENVIRONMENT AND RELEASES
000661  *  THE RESOURCES AFTER READING THE JOYSTICKS.
000662  *
000663                  REP         60
000664  AD.CLNUP        EQU         *
000665                  LDA         E.REG
000666                  AND         #$3C                ;RESTORE RAM AT $C000 & $F000
000667                  STA         E.REG
000668                  LDA         #JOYSIRSIZ
000669                  LDX         #>JOYSIRTBL
000670                  LDY         #<JOYSIRTBL
000671                  JSR         DEALCSIR            ;DEALLOCATE RESOURCES
000672                  RTS
000673                  PAGE
000674                  REP         60
000675  *
000676  *  SUBROUTINE AD.READ
000677  *  THIS SUBROUTINE READS A SPECIFIED A/D INPUT AND RETURNS
000678  *  AN 8 BIT RESULT.  IT ASSUMES THAT THE A/D RESOURCES HAVE
000679  *  BEEN ALLOCATED, THE I/O SPACE AND $F000 ROM HAVE BEEN
000680  *  SELECTED, AND THE SYSTEM IS RUNNING IN 2 MHZ MODE.
000681  *
000682  *  PARAMETERS:
000683  *    A:  A/D INPUT PORT (0-7)
000684  *
000685  *  RETURN VALUE:
000686  *    A:  RESULT (0 - 255)
000687  *    X, Y:  UNDEFINED
000688  *
000689                  REP         60
000690  *
000691  AD.READ         EQU         *
000692                  LSR         A                   ;SELECT THE APPROPRIATE
000693                  BIT         AD.SEL0             ;  A/D INPUT
000694                  BCC         ADR010
000695                  BIT         AD.SEL0+1
000696  ADR010          LSR         A
000697                  BIT         AD.SEL1
000698                  BCC         ADR020
000699                  BIT         AD.SEL1+1
000700  ADR020          LSR         A
000701                  BIT         AD.SEL2
000702                  BCC         ADR030
000703                  BIT         AD.SEL2+1
000704  ADR030          PHP
000705  *
000706  ADR040          CLI
000707                  BIT         AD.CHRG             ;CHARGE A/D CAPACITOR
000708                  LDA         #>TCHARGE
000709                  STA         D.T2
000710                  LDA         #<TCHARGE
000711                  STA         D.T2+1
000712                  LDA         #BITON5
000713  ADR050          BIT         D.IFR
000714                  BEQ         ADR050
000715  *
000716                  SEI
000717                  SEC
000718                  LDA         #>TOFFSET
000719                  STA         D.T2                ;SET UP TIMER
000720                  LDA         #<TOFFSET
000721                  BIT         AD.STRT             ;START A/D TIMEOUT
000722                  JSR         ANALOG              ;MEASURE CONVERSION TIME
000723                  BCC         ADR070
000724  *
```

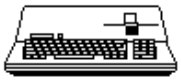


```
000725  ADR060          CLI                             ;PROCESS AN INTERRUPT
000726                  SEI
000727                  BIT         AD.FLAG             ;STILL TIMING?
000728                  BPL         ADR040              ;  NO -- START OVER
000729                  JSR         ANLOG1              ;  YES -- CONTINUE
000730                  BCS         ADR060
000731  *
000732  ADR070          PLP
000733                  EOR         #$FF                ;NORMALIZE RESULT
000734                  BMI         ADR080              ;RESULT < 0
000735                  STA         AD.TEMP
000736                  TYA
000737                  EOR         #$FF
000738                  LSR         AD.TEMP
000739                  ROR         A
000740                  LSR         AD.TEMP
000741                  ROR         A
000742                  LSR         AD.TEMP
000743                  BNE         ADR090              ;RESULT > 255
000744                  ROR         A
000745                  ADC         #0
000746                  RTS
000747  ADR080          LDA         #0
000748                  RTS
000749  ADR090          LDA         #$FF
000750                  RTS
000751                  PAGE
000752                  REP         60
000753  *
000754  *  SYSTEM COLD START
000755  *
000756  *  THIS ROUTINE IS CALLED TO TELL THE USER TO REBOOT THE
000757  *  SYSTEM.  IT CLEARS THE SCREEN, DISPLAYS A MESSAGE,
000758  *  OVERWRITES BANKED MEMORY, AND HANGS UNTIL THE USER
000759  *  PERFORMS A HARD RESET.
000760  *
000761                  REP         60
000762  *
000763  *
000764  COLDSTRT        EQU         *
000765                  SEI                             ;SHUT DOWN INTERRUPTS
000766                  LDA         #$40                ;  AND IGNORE NMI
000767                  STA         $FFCA
000768                  LDA         #$67
000769                  STA         E.REG               ;DISABLE RESET
000770                  LDA         #$00
000771                  STA         Z.REG               ;USE PAGE ZERO
000772  *
000773                  LDX         SYSBANK
000774                  LDA         #$BF
000775                  LDY         #$00
000776                  STY         MEMORY
000777  CS010           STA         MEMORY+1
000778                  STX         B.REG
000779                  LDA         #$A0
000780  CS020           STA         (MEMORY),Y          ;SET MEMORY TO BLANKS
000781                  DEY
000782                  BNE         CS020
000783                  DEC         MEMORY+1
000784                  BNE         CS020
000785                  DEX
000786                  BPL         CS010
000787  *
000788                  LDY         #6
000789  CS030           STA         $C050,Y             ;SELECT 40 COLUMN
000790                  DEY                             ;  BLACK & WHITE TEXT
000791                  BPL         CS030
000792  *
000793                  LDY         #BOOTLEN
000794  CS040           LDA         BOOTMSG-1,Y         ;PRINT BOOT MESSAGE
000795                  STA         BOOTADR-1,Y
000796                  DEY
000797                  BNE         CS040
000798  *
000799                  LDA         #$77
000800                  STA         E.REG               ;ENABLE RESET
000801                  JMP         *                   ;HANG UNTIL RESET
000802                  PAGE
000803                  MSB         ON
000804  BOOTMSG         ASC         "INSERT SYSTEM DISKETTE & REBOOT"
000805  BOOTLEN         EQU         *-BOOTMSG
```

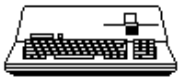


```
000806  BOOTADR         EQU         40-BOOTLEN/2+$628
000807                  MSB         OFF
000808                  LST         ON
000809  ZZEND           EQU         *
000810  ZZLEN           EQU         ZZEND-ZZORG
000811                  IFNE        ZZLEN-LENUMGR
000812                  FAIL        2,"SOSORG            FILE IS INCORRECT FOR UMBR"
000813                  FIN
000814
000815  ********************************************************************************
000816  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: UMGR.SRC
000817  ********************************************************************************
000818
```

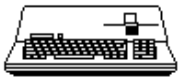
End of File -- Lines: 818 Characters: 20770



```
================================================================================
FILE: "SOS.VOLUME.TEXT"
================================================================================

000001  ********************************************************************
000002  * APPLE /// SOS 1.3 SOURCE CODE FILE: VOLUME
000003  ********************************************************************
000004  * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000005
000006                  PAGE
000007                  REP         40
000008  * NAME    : VOLUME
000009  * FUNCTION: RETURN VOLUME INFO
000010  * INPUT   : DEVICE NAME
000011  * OUTPUT  : THE INFO
000012  * VOLATILE: ALL REGS
000013                  REP         40
000014  *
000015  VOLUME          EQU         *
000016                  LDA         C.DNAMP             ; TRANSFER DEVICE NAME
000017                  STA         DVNAMP              ; NAME FOR DMGR
000018                  LDA         C.DNAMP+1
000019                  STA         DVNAMP+1
000020                  LDA         SISTER+C.DNAMP+1    ; AND XTND
000021                  STA         SISTER+DVNAMP+1
000022                  JSR         GETDNUM             ; GET DEVNUM
000023                  BCC         VOL7                ; =>SOME KINDA ERROR
000024                  RTS                             ; RETURN ERROR
000025  VOL7            BMI         VOL2                ; =>IT'S GOOD...
000026                  LDA         #NOTBLKDEV          ; NOT BLOCKED
000027                  JMP         VOLERR              ; =>RETURN THE ERROR
000028  *
000029  * UNCONDITIONALLY READ ROOT DIRECTORY:
000030  *
000031  VOL2            EQU         *
000032                  LDA         SCRTCH+1
000033                  STA         DEVNUM              ; SETUP DEV NUMBER
000034                  LDA         #2                  ; BLKNUM=2
000035                  LDX         #0
000036                  JSR         GETROT0             ; GET IT PLEASE
000037                  LDA         #VNFERR             ; ERROR CODE
000038                  BCC         VOL8                ; BRANCH IF NO ERROR ON READ
000039                  RTS                             ; =>ERROR, PASS IT ON.
000040  *
000041  VOL8            LDA         #>VCB               ; SET VCBPTR TO THE
000042                  STA         VCBPTR              ; FIRST OF THEM
000043                  LDA         #<VCB
000044                  STA         VCBPTR+1
000045  *
000046  * IS THIS VOLUME SOS OR OTHER?
000047  *
000048                  JSR         TSTSOS              ; WHICH KIND?
000049                  BCC         VLOGGED             ; =>IT'S SOS
000050                  JMP         VNOTSOS             ; =>NOT SOS
000051  *
000052  * IS THIS SOS VOLUME LOGGED IN?
000053  *
000054  VLOGGED         EQU         *
000055                  JSR         CMPVCB              ; DOES VOLNAME MATCH?
000056                  BCC         VFOUND              ; =>YES, WE KNOW ABOUT IT.
000057                  JSR         VNXTVCB             ; BUMP TO NEXT
000058                  BCC         VLOGGED             ; =>TRY 'EM ALL...
000059                  BCS         VNEW                ; =>NOT FOUND, IT'S NEW (BRANCH ALWAYS)
000060  *
000061  *
000062  * IT'S BEEN LOGGED IN BEFORE:
000063  *  IS IT SWAPPED IN OR OUT?
000064  *
000065  VFOUND          EQU         *
000066                  LDY         #VCBSWAP            ; INDEX TO IT
000067                  LDA         (VCBPTR),Y          ; SWAPPED?
000068                  BPL         VFOUND1             ; =>IN. RETURN THE INFO
000069  *
000070  * SWAPPED OUT. BEFORE WE SWAP IT
000071  *  IN, MAKE SURE IT BELONGS ON
000072  *  THIS DEVICE!
000073  *
000074                  LDY         #VCBDEV             ; INDEX TO IT
000075                  LDA         (VCBPTR),Y          ; GET ITS DEVICE
000076                  CMP         DEVNUM              ; CORRECT DEVICE?
```



```
000077                  BEQ         VSWAPIN             ; =>YES
000078                  LDA         #DUPVOL             ; IF FOR ANOTHER DEV,
000079                  JMP         VOLERR              ; THEN IT'S AN ERROR!
000080  *
000081  * NOW SWAP-IN THIS VOLUME:
000082  *
000083  VSWAPIN         EQU         *
000084                  JSR         SWAPIN              ; SWAP IT IN
000085                  JMP         VINFO               ; AND RETURN THE INFO
000086  *
000087  VFOUND1         LDY         #VCBDEV
000088                  LDA         (VCBPTR),Y          ; SAME DEVICES?
000089                  CMP         DEVNUM
000090                  BEQ         VINFO               ; YES; RETURN THE INFORMATION
000091                  LDY         #VCBSTAT
000092                  LDA         (VCBPTR),Y          ; OPEN FILES?
000093                  BPL         VFOUND2             ; BRANCH IF NOT
000094                  LDA         #DUPVOL
000095                  BNE         VOLERR              ; ELSE REPORT DUPLICATE VOLUME ERROR (BRANCH ALWAYS)
000096  VFOUND2         LDY         #VCBNML             ; MOVE THE LOGIN TO THIS NEW DEVICE
000097                  LDA         #0                  ; BY UNLOGGING THE OLD
000098                  STA         (VCBPTR),Y          ; AND LOGGING IN THE NEW (DROP INTO VNEW)
000099                  REP         40
000100  *
000101  * IT'S A BRAND NEW VOLUME.
000102  *  GUESS WE'LL HAVE TO LOG IT IN:
000103  *
000104  VNEW            EQU         *
000105                  LDA         DEVNUM              ; PASS A REG TO SWAPOUT
000106                  JSR         SWAPOUT             ; SWAP ANY ACTIVE VOL ON THIS DEVICE
000107                  BCC         VNEW1               ; BRANCH ON NO ERROR
000108                  LDA         #XIOERROR
000109                  RTS
000110  VNEW1           LDA         #>VCB               ; FIND AN EMPTY VCB
000111                  STA         VCBPTR
000112                  LDA         #<VCB
000113                  STA         VCBPTR+1
000114  VFREE           LDY         #VCBNML
000115                  LDA         (VCBPTR),Y          ; EMPTY VCB?
000116                  BEQ         VLOGIN              ; ITS FREE, USE IT
000117                  LDY         #VCBDEV
000118                  LDA         (VCBPTR),Y          ; OR ONE WITH SAME DEVICE
000119                  CMP         DEVNUM
000120                  BNE         VFREEX              ; BRANCH IF NO DEVICE MATCH
000121                  LDY         #VCBSTAT
000122                  LDA         (VCBPTR),Y          ; AND NO OPEN FILES
000123                  BPL         VLOGIN              ; BRANCH IF OK TO REUSE THIS VCB
000124                  LDA         DEVNUM              ; THEN WE MUST SWAP OUT THIS VOLUME
000125                  JSR         SWAPOUT
000126                  BCC         VFREEX              ; SWAPOUT PROCEEDED OK
000127                  LDA         #XIOERROR           ; ELSE REPORT ERROR
000128                  RTS
000129  VFREEX          JSR         VNXTVCB             ; TRY NEXT
000130                  BCC         VFREE               ; MORE TO COME
000131  * RAN OUT OF MT'S ... FIND W/O FILES
000132  VNFIL           LDY         #VCBSTAT
000133                  LDA         (VCBPTR),Y
000134                  BPL         VLOGIN
000135                  JSR         VNXTVCB
000136                  BCC         VNFIL
000137  * ALL OPEN ... REPORT VCBFULL
000138                  LDA         #FCBFULL
000139                  BNE         VOLERR
000140  VLOGIN          EQU         *
000141                  JSR         LOGVCB              ; AND LOGIN THIS ONE
000142                  REP         40
000143  *
000144  * RETURN ALL THE NICE INFO:
000145  *
000146  VINFO           EQU         *
000147                  LDA         #0
000148                  LDY         #VCBTFRE            ; FETCH VOLUME FREE BLOCK COUNT
000149                  STA         (VCBPTR),Y          ; FORCE RESCAN OF ALL
000150                  INY                             ; BITMAPS
000151                  STA         (VCBPTR),Y          ; TO MAKE SURE VCB INFO CURRENT
000152                  STA         REQL                ; FREE BLOCKS
000153                  STA         REQH
000154                  JSR         TSFRBLK
000155  *
000156                  LDX         VCBPTR              ; GET VCB INDEX
000157                  LDY         #0
```



```
000158  VINFO1          EQU         *
000159                  LDA         VCB+VCBTBLK,X       ; MOVE TOTAL
000160                  STA         (C.OUTBLK),Y        ; BLOCKS AVAIL
000161                  INX
000162                  INY
000163                  CPY         #4                  ; AND FREE ONES TOO
000164                  BNE         VINFO1
000165  *
000166                  LDY         #0                  ; NOW DO VOLNAME
000167                  LDA         (VCBPTR),Y
000168                  TAY
000169  VINFO2          EQU         *
000170                  LDA         (VCBPTR),Y
000171                  STA         (C.OUTVOL),Y
000172                  DEY
000173                  BPL         VINFO2
000174                  CLC
000175                  BCC         VOLRET              ; =>DONE
000176  *
000177  VOLERR          EQU         *
000178                  SEC
000179  VOLRET          EQU         *
000180                  RTS
000181                  PAGE
000182                  REP         40
000183  * THIS ISN'T A SOS VOLUME. MARK
000184  *  THE ACTIVE VOL THIS DEVICE
000185  *  SO THAT IT GETS CHECKED LATER:
000186  *
000187  VNOTSOS         EQU         *
000188                  LDY         #VCBDEV             ; IS VCB FOR THIS
000189                  LDA         (VCBPTR),Y          ; DEVICE?
000190                  CMP         DEVNUM
000191                  BNE         VNS2
000192                  LDY         #VCBSTAT            ; INDEX TO IT
000193                  LDA         (VCBPTR),Y          ; GET STATUS
000194                  BPL         VNS2                ; =>NOT ACTIVE.
000195                  ORA         #DSWITCH            ; SET 'SWITCHEROO'
000196                  STA         (VCBPTR),Y          ; PUT IT BACK
000197  *
000198  VNS2            EQU         *
000199                  JSR         VNXTVCB             ; GET NEXT VCB
000200                  BCC         VNOTSOS             ; =>TRY 'EM ALL.
000201  *
000202                  LDA         #NOTSOS             ; GIVE THE ERROR
000203                  BNE         VOLERR              ; (BRANCH ALWAYS)
000204                  SKP         5
000205  * NAME    : VNXTVCB
000206  * FUNCTION: BUMP VCBPTR TO NEXT VCB
000207  * INPUT   : NOTHING
000208  * OUTPUT  : VCBPTR UPDATED
000209  *         : 'BCC' IF MORE TO GO
000210  *         : 'BCS' IF DONE
000211  * VOLATILE: AC
000212  *
000213  VNXTVCB         EQU         *
000214                  LDA         VCBPTR
000215                  CLC
000216                  ADC         #VCBSIZE            ; BUMP IT
000217                  STA         VCBPTR
000218                  RTS                             ; CARRY SET IF END OF PAGE
000219
000220                  CHN         CREATE,4,1
000221
000222  ********************************************************************************
000223  * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: VOLUME
000224  ********************************************************************************
000225
```

End of File -- Lines: 225 Characters: 7048

SUMMARY:

  Total number of files : 58
  Total file lines   : 17223
  Total file characters : 546786