```
*
BFM.INIT2       EQU       *
                INC       STATE
                BMI       BFMI050
                JSR       GETK
                LDA       RETRY
                BEQ       BADNEWS
                BCC       BFMI050
                JSR       NMIDSBL
                JSR       DC
                INC       STATE
BFMI050         CLC
                RTS
BADNEWS         SEC                          ; I/O ERROR
                RTS
                PAGE
                REP       60
*
* DECODE SUBROUTINE
*
* TO ENCODE:
*    E0.E8:        - INIT KEY  & PREV.K
*    B84E:4C 64 B8 - JUMPS AROUND INTERP'S 3 BYTE OVERWRITE
*    1A02.1A03:    - NEW INTERP'S LOAD ADR (LO,HII)
*    B81DG:        - JSR FROM MONITOR
*
                REP       60
DC              EQU       *
                LDA       B.REG              ; SAVE BANK REGISTER
                PHA
                LDA       SYSBANK            ;    AND SWITCH TO SYSTEM BANK
                STA       B.REG
                CLC                          ; FETCH LOADER'S INTERPRETER POINTER
                LDA       CZPAGE+I.BASE.P
                ADC       #3
                STA       I
                PHA
                LDA       CZPAGE+I.BASE.P+1
                ADC       #0
                STA       I+1
                PHA
                LDA       #0
                STA       SXPAGE+I+1
*
                LDY       I                  ; ALIGN I PTR TO PAGE BOUNDARY
                LDA       #0
                STA       I
                STA       PREV.K
*
                JSR       DCLOOP             ; DECODE
*
                PLA                          ; RETRIEVE LOADER'S INTERPRETER POINTER
                STA       I+1
                PLA
                STA       I
*
                LDY       #1                 ; REPOSITION LOADER'S INTERPRETER POINTER (PUT ENCODE JMP HERE)
                LDA       (I),Y
                STA       CZPAGE+I.BASE.P
                INY
                LDA       (I),Y
                STA       CZPAGE+I.BASE.P+1
*
                LDY       #2                 ; WALK ON INTERPRETER'S FIRST INSTRUCTION (3 BYTES)
                LDA       #0
DCA             STA       (I),Y
                DEY
                BPL       DCA
                PLA                          ; RESTORE BANK REGISTER (ENCODE JMP JUMPS TO HERE)
                STA       B.REG
                RTS
                PAGE
                REP       60
*
* DECODE LOOP SUBROUTINE
*
                REP       60
DCLOOP          EQU       *
                LDX       #7                 ; SHIFT LEFT ONE BIT
                CLC
                LDA       KEY
```

```
                BPL     DC1
                SEC
DC1             ROL     KEY,X
                DEX
                BPL     DC1
*
DC2             TYA
                AND     #7
                EOR     #2
                TAX
                LDA     KEY,X
                PHA
                AND     #7
                TAX
                PLA
                CLC
                ADC     PREV.K
                CLC
                ADC     KEY,X
                STA     PREV.K
                EOR     (I),Y               ; DECODE BYTE
                STA     (I),Y               ; AND PUT IT BACK
                INY
                BNE     DC2
                INC     I+1
                LDA     I+1
                CMP     #<KERNEL.BASE
                BCC     DCLOOP
                RTS
                PAGE
                REP     60
*
* GETKEY SUBROUTINE
*
                REP     60
*
RETRY           DFB     10+1                ;TEN RETRIES
OURTRACK        DS      1                   ;CURRENT TRACK/D3RRA81084
*
GETK            EQU     *
                LDX     #7
                STX     XIDX
                LDX     #SLOT
                LDA     MOTORON,X           ;ENSURE MOTOR STAYS ON
                LDA     E.REG               ; SELECT 1MHZ, ROM
                ORA     #$83
                STA     E.REG
*
* NOTE: THE SEEKDSK3 ROUTINE HAS THESE /D3RRA81084
*  CAVEATS: 1MHZ MODE, MOTOR IS ON, /D3RRA81084
*  DRIVE CURRENTLY SELECTED, ROM+I/O ENABLED! /D3RRA81084
*
GETK010         LDA     #BEGTRK
                STA     OURTRACK            ;WHERE WE SEEK TO /D3RRA81084
                JSR     SEEKDSK3            ;HAVE DISKDH SEEK FOR US /D3RRA81084
GETK020         LDX     #SLOT
                JSR     DOREAD              ;FIND A SECTOR HEADER
                BCS     IOERROR             ;=>RETRY IF BAD
                LDA     SECTOR              ;WHERE ARE WE?
                CMP     #BEGSECT            ;AT THE RIGHT PLACE?
                BNE     GETK020             ;=>NO, GET THERE
*
GETK100         LDX     #1
                JSR     WAIT                ; (X * 1284) + 15 MILISECONDS
                LDX     XIDX
                LDA     VOLUME
                STA     KEY,X
                DEC     XIDX
                BMI     ENUFF
                INC     OURTRACK            ;BUMP FOR NEXT TRACK /D3RRA81084
                LDA     OURTRACK            ;WHERE TO GO /D3RRA81084
                LDX     #SLOT
                JSR     SEEKDSK3            ;DISKDH, PLEASE SEEK ME /D3RRA81084
                LDX     #SLOT
                JSR     DOREAD
                BCC     GETK100
                BCS     IOERROR
*
ENUFF           LDX     #SLOT
                LDA     MOTOROFF,X
                LDA     E.REG               ; SELECT 2MHZ, RAM
```

```
                AND     #$7C
                STA     E.REG
                PAGE
                LDA     SECTOR
                CMP     #ENDSECT            ;TRACKS SYNC'ED?
                BNE     NOTPROT
                LDA     KEY
                EOR     KEY+1
                BEQ     NOTPROT             ;IF FIRST 2 VOLS ARE EQUAL
                SEC
                RTS
*
NOTPROT         LDA     #0
                CLC
                RTS
*
*
DOREAD          JSR     WHICHROM
                BCS     OLDREAD
                JMP     RDADR
OLDREAD         JMP     RDADRX
*
*
WHICHROM        LDA     RDADR
                CMP     #ROMID
                CLC
                BEQ     NEWROM
                SEC
NEWROM          RTS
*
*
IOERROR         DEC     RETRY
                BEQ     ERR1
                JMP     GETK                ; TRY, TRY AGAIN
ERR1            JMP     ENUFF               ; I/O ERROR, CLEANUP AND EXIT
*
*
WAIT            LDY     #0
W1              DEY
                BNE     W1
                DEX
                BNE     W1
                RTS

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

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

End of File -- Lines: 292 Characters: 7017

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

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

                SBTL       "SOS 1.1  BUFFER MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
*ORGBUFMG EQU $F552
*LENBUFMG EQU $31C
                ORG        ORGBUFMG
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* BUFFER MANAGER (VERSION = 1.10   )
*                (DATE    = 8/04/81)
*
* THIS MODULE IS RESPONSIBLE FOR CREATING AND RELEASING BUFFERS
* FOR BOTH THE BLOCK FILE MANAGER AND, LATER, DEVICE HANDLERS
* THE BUFFER MANAGER CREATES BUFFERS BY REQUESTING MEMORY
* SEGMENTS FROM THE MEMORY MANAGER, AND RELEASES THEM VIA SAME.
* THE PRIMARY DATA STRUCTURE IN THIS MODULE IS THE BUFFER TABLE.
*
                REP        60
*
                ENTRY      REQBUF
                ENTRY      REQFXBUF
                ENTRY      GETBUFADR
                ENTRY      CHKBUF
                ENTRY      RELBUF
*
                EXTRN      MMGR
                EXTRN      SXPAGE
                EXTRN      CZPAGE
                EXTRN      CXPAGE
*
                EXTRN      SYSERR
                EXTRN      SERR
                EXTRN      OUTOFMEM
                EXTRN      BUFTBLFULL
                EXTRN      BADSYSBUF
*
                EXTRN      SYSDEATH
                EXTRN      BADBUFNUM
                EXTRN      BADBUFSIZ
*
                ENTRY      BUF.CNT
                ENTRY      PGCT.T
                ENTRY      XBYTE.T
                ENTRY      BUFREF
                PAGE
                REP        60
*
* DATA DECLARATIONS
*
                REP        60
*
Z.REG           EQU        $FFD0
*
* MEMORY MGMT CALL PARM LOCATIONS ON SOS ZPAGE
*
M.TPARMX        EQU        $60                  ; FIRST ADR OF MEM SYS CALL PARMS ON SOS ZPAGE
REQCODE         EQU        M.TPARMX+$0
*
FINDSEG         EQU        $1
SRCHMODE        EQU        M.TPARMX+$1
F.ID            EQU        M.TPARMX+$2
F.PGCT          EQU        M.TPARMX+$3
F.PGCTX         DS         2                    ; TEMP LOC FOR F.PGCT PARM
F.BASE          EQU        M.TPARMX+$5
F.BASEX         DS         2                    ; TEMP LOC FOR F.BASE PARM
F.LIM           EQU        M.TPARMX+$7
```

```
F.LIMX          DS         2                    ; TEMP LOC FOR F.LIM PARM
F.NUM           EQU        M.TPARMX+$9
F.NUMX          DS         1                    ; TEMP LOC FOR F.NUM PARM
*
RELSEG          EQU        $5
RLS.NUM         EQU        M.TPARMX+$1
*
* REQBUF DATA DECLARATIONS
*
RQB.PGCT        DS         1                    ; REQUESTED PAGE COUNT
RQB.BNUM        DS         1                    ; BUFFER NUMBER (FM GETFREE CALL)
*
* REQFXBUF DATA DECLARATIONS
*
RQFB.PGCT       DS         1                    ; REQUESTED PAGE COUNT
RQFB.BNUM       DS         1                    ; BUFFER NUMBER (FM GETFREE CALL)
MAXPGCT         EQU        64                   ; MAX BUFSIZE=16K
F.TPARMX        EQU        $A0                  ; FIRST ADR OF FILE SYS CALL PARMS ON SOS ZPAGE
OPEN.LIST       EQU        F.TPARMX+$5          ; LOC OF OPEN.LIST PARM (OPEN SYS CALL)
*
* BUFCOMPACT DATA DECLARATIONS (SOURCE ALSO USED BY CHKBUF)
*
BUFC.BNUM       DS         1                    ; BUF# OF LOWEST BUFFER IN BUF.TBL
SOURCE          EQU        M.TPARMX+$10         ; & $11
DEST            EQU        M.TPARMX+$12         ; & $13
                PAGE
                REP        60
*
* BUFFER TABLE
*
* THE BUFFER TABLE CONSISTS OF "CNT"-1 ENTRIES (1 TO "CNT"-1).
* EACH ENTRY IS "SIZ" BYTES IN LENGTH.  THE "PGCT" FIELD
* CONTAINS 3 SUBFIELDS.  BIT 7 IS THE "FREE" FLAG (0=ACTIVE,1=FREE)
* BIT 6 IS THE "FIXED" FLAG (0=FLOATING BUFFER,1=FIXED BUFFER)
* BITS 5 THRU 0 CONTAIN THE PAGE COUNT OF AN "ACTIVE" ENTRY
* (0=>1 PAGE,63=>64 PAGES DECIMAL).  THE "XBYTE" FIELD CONTAINS
* THE PROPER XBYTE OF AN "ACTIVE" ENTRY.  THE "ADRH" FIELD
* CONTAINS THE HIGH BYTE OF THE BUFFER ADDRESS.  IF THE
* BUFFER ENTRY IS "FLOATING", THEN THE "SEG" FIELD CONTAINS THE
* SEGMENT NUMBER AND THE LOW BYTE OF THE BUFFER ADDRESS IS
* ASSUMMED TO BE ZERO.
*
* THUS, THE FOLLOWING RESTRICTIONS APPLY TO BUFFERS:
*
* (1) MAXIMUM BUFFER LENGTH IS 64 PAGES (16K)
* (2) "FLOATING" BUFFERS ALWAYS BEGIN ON A PAGE BOUNDARY
*     "FIXED" BUFFERS DO NOT.
* (3) BUFFERS ARE ALWAYS AN INTEGRAL NUMBER OF PAGES IN LENGTH
* (4) BUFFERS ALWAYS RESIDE IN THE 32K BANK MEMORY REGION,
*     A LIMITATION OF FIND.SEG (MEMORY MANAGER)
* (5) MAXIMUM NUMBER OF BUFFERS = 16; ENTRY 0 IS NOT USED.
*
                REP        60
*
* BUFFER TABLE
*
BUF.SIZ         EQU        5
BUF.CNT         EQU        17
BUF.TBL         DS         BUF.SIZ*BUF.CNT
PGCT.T          EQU        BUF.TBL
XBYTE.T         EQU        PGCT.T+BUF.CNT
ADRH.T          EQU        XBYTE.T+BUF.CNT
SEG.T           EQU        ADRH.T+BUF.CNT
ADRL.T          EQU        SEG.T
CHK.T           EQU        ADRL.T+BUF.CNT
ISFIXED         EQU        $40
ISFREE          EQU        $80
*
* BUFFER REFERENCE TABLE
*
* FIRST BYTE IS COUNT, FOLLOWED BY "COUNT" BUFFER #S.
* THIS TABLE IS A LIST OF ALL BUFFERS REFERENCED DURING ONE
* SOS SYSTEM CALL.  BUFFER #S ARE ADDED TO THIS LIST BY
* GETBUFADR AND REMOVED BY CHKSUM.
*
BUFREF.CNT      EQU        17
BUFREF          DS         BUFREF.CNT
ZPAGEX          DS         1
                PAGE
                REP        60
*
```

```
* REQBUF
*
* INPUT:  PAGE.CNT (A)
* OUTPUT: BUFNUM   (A)
* ERROR:  "BUFFER TABLE FULL" - SYSERR
*         "OUT OF MEMORY"     - SYSERR
*         "BAD BUFFER SIZE"   - SYSDEATH
*
* THIS ROUTINE FINDS A FREE ENTRY IN THE BUFFER TABLE
* AND THEN CALLS FIND.SEG (MMGR) TO OBTAIN MEMORY FOR IT.
* IF MEMORY IS FOUND THEN THE BUFFER ENTRY IS MARKED "ACTIVE"
* AND THE BUFFER INFO IS INSERTED INTO THE ENTRY
*
                REP        60
*
REQBUF          EQU        *
*
* IF REQUESTED PGCT OUT OF BOUNDS THEN FATAL ERR
*
                TAY
                BEQ        RQB.ERR2              ; FATAL ERR, INVALID BUFFER SIZE
                CPY        #MAXPGCT+1
                BCS        RQB.ERR2              ; FATAL ERR, INVALID BUFFER SIZE
                STY        RQB.PGCT              ; SAVE PAGE COUNT
*
* FIND FREE ENTRY IN BUF.TBL
*
                JSR        GETFREE
                BCS        RQB.ERR               ; ERR, BUFFER TABLE FULL
                STX        RQB.BNUM
*
* FIND PGCT*256 BYTES OF FREE MEMORY
*
                LDA        RQB.PGCT
                JSR        FSEG
                BCS        RQB.ERR1              ; ERR, OUT OF MEMORY
*
* INSERT PGCT, XBYTE, ADRH, SEG#, CHK BYTE IN BUF.TBL(BUF#)
*
                LDX        RQB.BNUM
                DEC        RQB.PGCT              ; PAGE COUNT FIELD
                LDA        RQB.PGCT
                STA        PGCT.T,X
*
                LDX        F.BASEX               ; XBYTE & ADRH FIELDS
                LDY        F.BASEX+1
                JSR        CNVRT.ADR
                CPX        #$8F
                BNE        RQB010
                LDX        #$7F                  ; IF XBYTE=$8F THEN XBYTE:=$7F
RQB010          TXA
                LDX        RQB.BNUM
                STA        XBYTE.T,X
                TYA
                STA        ADRH.T,X
*
                LDA        F.NUMX                ; SEG# FIELD
                STA        SEG.T,X
*
                LDA        #0                    ; INIT CHECK BYTE TO NULL
                STA        CHK.T,X
*
                TXA                              ; RETURN BUF#
                CLC
                RTS                              ; NORMAL EXIT
*
*
RQB.ERR         LDA        #BUFTBLFULL
                JSR        SYSERR
*
RQB.ERR1        LDA        #OUTOFMEM
                JSR        SYSERR
*
RQB.ERR2        LDA        #BADBUFSIZ
                JSR        SYSDEATH
                PAGE
                REP        60
*
* REQFXBUF
*
* INPUT:  PAGE.CNT (A)
```

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

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

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

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

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

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

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

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

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

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

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

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

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

                SBTL       "SOS 1.1  CHARACTER FILE MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGCFM
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* CHARACTER FILE MANAGER (VERSION = 1.1O   )
*                        (DATE    = 8/04/81)
*
* THIS MODULE TRANSFORMS CHARACTER FILE SYSTEM CALLS INTO
* DEVICE CALLS TO THE APPROPRIATE DEVICE HANDLER.  ONLY
* OPEN, NEWLINE, READ, WRITE AND CLOSE CALLS ARE PERMITTED
* ON CHARACTER FILES.
*
                REP        60
*
                ENTRY      CFMGR
*
                ENTRY      CFCB.MAX
                ENTRY      CFCB.DEV
*
                EXTRN      DMGR
                EXTRN      LEVEL
                EXTRN      MAX.DNUM
                EXTRN      SXPAGE
*
                EXTRN      SYSERR
                EXTRN      SERR
                EXTRN      BADSCNUM
                EXTRN      CFCBFULL
                EXTRN      BADREFNUM
                EXTRN      FNFERR
                PAGE
                REP        60
*
* DATA DECLARATIONS
*
                REP        60
*
* FILE CALL PARM LOCATIONS ON SOS ZPAGE
*
F.TPARMX        EQU        $A0
REQCODE         EQU        F.TPARMX
O.PATH          EQU        F.TPARMX+1           ; OPEN'S PATHNAME LOC
O.REFNUM        EQU        F.TPARMX+3           ; OPEN'S REFNUM LOC
REFNUM          EQU        F.TPARMX+1           ; REFNUM'S LOC IN OTHER CALLS
NL.ISNL         EQU        F.TPARMX+2           ; NEWLINE'S ISNEWLINE LOC
NL.NLCHR        EQU        F.TPARMX+3           ; NEWLINE'S NEWLINECHAR LOC
RW.BUF          EQU        F.TPARMX+2           ; READ/WRITE'S BUF LOC
RW.BYTES        EQU        F.TPARMX+4           ; READ/WRITE'S BYTES LOC
RD.BYTESRD      EQU        F.TPARMX+6           ; READ'S BYTESREAD LOC
*
* FILE REQUEST CODE VALUES
*
OPEN            EQU        8
NEWLINE         EQU        9
READ            EQU        $A
WRITE           EQU        $B
CLOSE           EQU        $C
                PAGE
* DEVICE CALL PARM LOCATIONS ON SOS ZPAGE
*
D.TPARMX        EQU        $C0
D.SCNUM         EQU        D.TPARMX             ; DEVICE SYS CALL # LOC
GDN.DNAME       EQU        D.TPARMX+1           ; GETDEVNUM DNAME LOC
GDN.DNUM        EQU        D.TPARMX+3           ; GETDEVNUM DNUM LOC
```

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

```
DSDIRACC        LDA        #ACCSERR
                JSR        SYSERR
*
DSDIR2          LDA        GBUF+2                ; GET FORWARD LINK.
                CMP        GBUF+3                ; TEST FOR NO LINK.
                BNE        DSDIR3
                CMP        #0
                BEQ        DSTLAST               ; IF NO LINK, THEN FINISHED.
DSDIR3          LDX        GBUF+3
                JSR        DEALLOC               ; FREE THIS BLOCK.
                BCS        DSDIRERR
                LDA        GBUF+2
                STA        BLOKNML
                LDA        GBUF+3
                STA        BLOKNMH               ; READ IN LINKED BLOCK.
                JSR        RDGBUF
                BCC        DSDIR2                ; LOOP UNTIL ALL ARE FREED.
DSDIRERR        RTS
*
*
                PAGE
WORKSPC         EQU        *
V.STATUS        DS         1                     ; VOLUME STATUS, INCLUDES 'ACTIVE' IN BIT 7
H.CREDT         DS         2                     ; DIRECTORY CREATION DATE
                DS         2                     ; DIRECTORY CREATION TIME
                DS         1                     ; VERSION UNDER WHICH THIS DIRECTORY WAS CREATED
                DS         1                     ; EARLIEST VERSION THAT IT'S COMPATABLE WITH
H.ATTR          DS         1                     ; ATTRIBUTES (PROTECT BIT, ETC.)
H.ENTLN         DS         1                     ; LENGTH OF EACH ENTRY IN THIS DIRECTORY.
H.MAXENT        DS         1                     ; MAXIMUM NUMBER OF ENTRIES PER BLOCK
H.FCNT          DS         2                     ; CURRENT NUMBER OF FILES IN THIS DIRECTORY
                DS         2                     ; ADDRESS OF FIRST ALLOCATION BIT MAP
                DS         2                     ; TOTAL NUMBER OF BLOCKS ON THIS UNIT
                DS         5                     ; (FOR FUTURE EXPANSION)
*
D.DEV           DS         1                     ; DEVICE NUMBER OF THIS DIRECTORY ENTRY
D.HEAD          DS         2                     ; ADDRESS OF <SUB> DIRECTORY HEADER
D.ENTBLK        DS         2                     ; ADDRESS OF BLOCK WHICH CONTAINS THIS ENTRY
D.ENTNUM        DS         1                     ; ENTRY NUMBER WITHIN BLOCK.
DFIL            EQU        *
D.STOR          EQU        *-DFIL                ; STORAGE TYPE * 16 + FILE NAME LENGTH
                DS         1
; *-DFIL ; FILE NAME
                DS         15
D.FILID         EQU        *-DFIL                ; USER'S IDENTIFICATION BYTE
                DS         1
D.FRST          EQU        *-DFIL                ; FIRST BLOCK OF FILE
                DS         2
D.USAGE         EQU        *-DFIL                ; NUMBER OF BLOCKS CURRENTLY ALLOCATED TO THIS FILE
                DS         2
D.EOF           EQU        *-DFIL                ; CURRENT END OF FILE MARKER
                DS         3
D.CREDT         EQU        *-DFIL                ; DATE OF FILE'S CREATION
                DS         2
; *-DFIL ; TIME OF FILE'S CREATION
                DS         2
;  EQU *-DFIL ; SOS VERSION THAT CREATED THIS FILE
                DS         1
D.COMP          EQU        *-DFIL                ; BACKWARD VERSION COMPATABILTY
                DS         1
D.ATTR          EQU        *-DFIL                ; 'PROTECT', READ/WRITE 'ENABLE' ETC.
                DS         1
D.AUXID         EQU        *-DFIL                ; USER AUXILLARY IDENTIFACATION
                DS         2
D.MODDT         EQU        *-DFIL                ; FILE'S LAST MODIFICATION DATE
                DS         2
D.MODTM         EQU        *-DFIL                ; FILE'S LAST MODIFICATION TIME
                DS         2
D.DHDR          EQU        *-DFIL                ; HEADER BLOCK ADDRESS OF FILE'S DIRECTORY
                DS         2
*
CMDADR          DS         2
SCRTCH          DS         13                    ; SCRATCH AREA FOR ALLOCATION ADDRESS CONVERSION
OLDEOF          DS         3                     ; TEMP USED IN W/R
OLDMARK         DS         3                     ; USED BY 'RDPOSN' AND 'WRITE'
SCRHIGH         EQU        <SCRTCH               ; AND DEVICE NUMBERS FROM BOB'S CODE.
*
                CHN        SWAPOUT/IN,4,2

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

```
GET.DNUM        EQU        *
*
                LDX        #1                    ; SETUP PTR TO 1ST SDT ENTRY
*
DM070           JSR        SETUP.SDT             ; SET UP ZPAGE PTR TO SDT ENTRY
*
                LDA        (SDTP),Y              ; COMPARE DNAME LENGTHS
                CMP        (G.DNAME),Y
                BNE        NXTSDT
*
                TAY                              ; LENGTHS MATCH, NOW COMPARE CHARS
DM080           LDA        (G.DNAME),Y
                CMP        #$60
                BCC        DM090
                AND        #$DF                  ; UPSHIFT
DM090           CMP        (SDTP),Y
                BNE        NXTSDT
                DEY
                BNE        DM080
*
                TXA                              ; CHARS MATCH
                LDY        #0
                STA        (G.DNUM),Y            ; OUTPUT DEV NUM PARM
                LDY        #DIB.DTYPE            ; SET "N" FLAG IN STATUS REG.
                LDA        (SDTP),Y              ; N=1(BLOCK DEVICE) N=0(CHAR DEVICE)
                CLC
                RTS                              ; NORMAL EXIT
*
NXTSDT          INX                              ; LAST SDT ENTRY?
                CPX        MAX.DNUM
                BCC        DM070
*
                LDA        #>NODNAME             ; ERROR, DNAME NOT FOUND IN SDT
                JSR        SYSERR                ;  RETURN TO CALLER
                PAGE
                REP        100
* SETUP.SDT(IN.X=DNUM, OUT.SDTP, B.REG, Y=0)   X="UNCHANGED"
                REP        100
SETUP.SDT       EQU        *
                LDA        SDT.DIBL,X            ; SET UP ZPAGE PTR TO SDT ENTRY
                STA        SDTP                  ; (POINTS TO DNAME FIELD)
                LDA        SDT.DIBH,X
                STA        SDTP+1
                LDA        SDT.BANK,X
                STA        B.REG
                LDY        #0
                STY        SXPAGE+SDTP+1
                RTS
*
                LST        ON
ZZEND           EQU        *
ZZLEN           EQU        ZZEND-ZZORG
                IFNE       ZZLEN-LENDMGR
                FAIL       2,"SOSORG             FILE IS INCORRECT FOR DEVMGR"
                FIN

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

```

End of File -- Lines: 298 Characters: 7991

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

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

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

```
BADOP           EQU       *
                LDA       #XBADOP             ;ILLEGAL COMMAND
                JMP       EXIT                ;BACK TO YOU
*
BADCMD          EQU       *
                LDA       #XREQCODE           ;INVALID COMMAND
                JMP       EXIT                ;BACK TO YOU
                PAGE
* SETUP WHAT WE HAVE TO BEFORE
*  PERFORMING THE I/O OPERATION:
*
IOSETUP         EQU       *
                LDA       D.BLOCK+1           ;VALIDATE BLOCKNUM
                BEQ       CHKBYTE             ;=> IF <256, IT'S OK
                CMP       #2                  ;IS IT <512?
                BCS       BADBLOCK            ;=>BAD BOY!
                LDA       D.BLOCK             ;YES, CHECK LO HALF
                CMP       #280-256            ; FOR RANGE
                BCC       CHKBYTE             ;=>IT'S OK
BADBLOCK        EQU       *
                LDA       #XBLKNUM            ;BAD BLOCK NUMBER
                JMP       EXIT                ;RETURN BAD NEWS
*
CHKBYTE         EQU       *
                LDA       D.BYTES             ;GET LO COUNT
                BNE       BADCOUNT            ;=>ERR, NOT INTEGRAL BLOCK(S)
                LDA       D.BYTES+1           ;GET HI COUNT
                LSR       A                   ;MAKE BLOCK COUNT
                BCS       BADCOUNT            ;=>BAD IF HALF-BLOCK COUNT
                STA       BLKCOUNT            ;SAVE COUNT OF BLOCKS
*
* DOES REQUESTED BYTECOUNT CAUSE US
*  TO RUN OFF END OF DISK?
*
                LDA       BLKCOUNT            ;NO. ADD STARTBLOCK
                CLC                           ; AND BLKCOUNT AND SEE
                ADC       D.BLOCK             ;  IF WE'RE TOO BIG
                LDX       D.BLOCK+1           ;DID IT START OUT > 255?
                BNE       BLKG255             ;=>YES
                BCC       DRVSETUP            ;=>DEFINITELY < 256
                BCS       CHKLO               ;=>IF CARRY,THEN >256
BLKG255         EQU       *
                BCS       BADCOUNT            ;>255+CARRY IS NOW >511
CHKLO           EQU       *
                CMP       #280-256+1          ;281..511 ?
                BCC       DRVSETUP            ;=>NO, WE ARE OK
BADCOUNT        EQU       *
                LDA       #XBYTECNT           ;ILLEGAL BYTECOUNT
                JMP       EXIT                ;SORRY...
                PAGE
*
* SELECT THE APPROPRIATE DRIVE:
*
DRVSETUP        EQU       *
                LDA       D.COMMAND           ;SAVE THIS COMMAND
                STA       PREVCMD             ; AND DEVICE FOR
                LDA       D.UNITNUM           ;  SUBSEQUENT
                STA       PREVUNIT            ;   'REPEAT' CALL
                LDA       E.REG               ;DOWNSHIFT TO
                ORA       #$80                ; 1MHZ FOR REMAINDER
                STA       E.REG               ;  OF DRIVER EXECUTION
                JSR       UNITSEL             ;SELECT & START IT
*
* SEE IF THE MOTOR STARTED. IF NOT,
*  THEN IT'S EITHER DISKSWITCH OR NODRIVE.
*
                JSR       CHKDRV              ;MOTOR RUNNING?
                BNE       DOIO                ;=>YES, GREAT.
*
* IF WE GET A MOTOR WHEN WE MOVE
*  THE HEAD, THEN IT'S DISKSWITCH.
*
                LDX       D.UNITNUM           ;FORCE HEAD MOTION
                INC       DRVTRACK,X          ; EVEN IF ALREADY ON ZERO
                INC       DRVTRACK,X          ;GIVE HIM A FIRM KNOCKER
                LDA       #0                  ;SEEK TO TRACK ZERO
                JSR       MYSEEK              ; FOR BFM DIR READ
                JSR       CHKDRV              ;RUNNING NOW?
                BNE       DSWITCH             ;=>YES, A SWITCHEROO
                LDA       #0
                LDY       D.UNITNUM           ;FORGET THAT THIS
```

```
                 STA     DRIVESEL,Y          ; DRIVE WAS 'SELECTED'
                 LDA     #XNODRIVE           ;NO, A MISSING DRIVE!
                 JMP     EXIT
*
DSWITCH          EQU     *
                 LDA     #XDISKSW            ;USER PULLED A FAST ONE
                 JMP     EXIT                ; BUT HE CAN'T FOOL US.
                 PAGE
* PREPARE TO DO THE OPERATION:
*
DOIO             EQU     *
                 LDA     D.BUFL              ;COPY USER BUFFER
                 STA     BUFTEMP             ; AND BLOCK NUMBER
                 LDA     D.BUFH              ;  TO OUR WORKSPACE
                 STA     BUFTEMP+1
                 LDA     $1400+D.BUFH
                 STA     $1400+BUFTEMP+1
                 LDA     D.BLOCK
                 STA     BLKTEMP
                 LDA     D.BLOCK+1
                 STA     BLKTEMP+1
*
* IF CALLER GAVE US A COUNT OF ZERO BYTES,
*  THEN WE'RE ALL DONE!
*
                 LDA     D.COMMAND           ;IS IT STATUS?
                 CMP     #2                  ;IF SO, THEN BYTECOUNT
                 BNE     DOIO2               ; IS MEANINGLESS
                 JMP     STATUS
DOIO2            EQU     *
                 LDY     BLKCOUNT            ;BLKS=0?
                 BEQ     READOK              ;=>YES, YOU GET GOOD RETURN
                 CMP     #0                  ;READ COMMAND?
                 BEQ     READREQ             ;=>YES
                 JMP     WRITEREQ
                 PAGE
                 REP     40
*  -- READ --
                 REP     40
READREQ          EQU     *
                 LDA     #0                  ;CLEAR COUNT OF
                 LDY     #0
                 STA     (D.BYTRD),Y         ; BYTES READ
                 INY
                 STA     (D.BYTRD),Y
READREQ2         EQU     *
                 JSR     BLK2SECT            ;COMPUTE TRK/SECTOR THIS BLOCK
*
                 JSR     SECTORIO            ;READ IT PLEASE
                 BCS     READERR             ;=>WE LOSE.
                 INC     SECTOR              ;BUMP TO NEXT
                 INC     SECTOR              ; LOGICAL SECTOR
                 INC     BUF+1               ;BUMP SECTOR BUFFER
                 JSR     SECTORIO            ;READ IT TOO
                 BCS     READERR             ;=>WE LOSE.
                 LDY     #1
                 LDA     (D.BYTRD),Y         ;BUMP COUNT OF
                 CLC
                 ADC     #2
                 STA     (D.BYTRD),Y         ; BYTES READ
*
* MORE BLOCKS TO GO?
*
                 JSR     MOREBLKS            ;SETUP FOR NEXT BLOCK
                 BNE     READREQ2            ;=>MORE TO READ...
READOK           EQU     *
                 LDA     #0                  ;GOOD RETURN
                 JMP     EXIT                ;TELL HAPPY USER
*
READERR          EQU     *
                 JMP     EXIT                ;RETURN ERROR CODE
                 CHN     DISK3.WRT.SRC

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

```

End of File -- Lines: 234 Characters: 7883

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

```
*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SIO.SRC
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
                PAGE
                REP        40
* NAME    : SECTORIO
* FUNCTION: READ OR WRITE A SECTOR
* INPUT   : IBSTRK, IBSECT, MONTIME,
* RETURNS : CARRY CLEAR IF OK (AC=00)
*         : CARRY SET   IF ERROR (AC=ERRCODE)
*         : SEEKWAIT  ALL SETUP
* DESTROYS: ALL REGISTERS
                REP        40
*
SECTORIO        EQU        *
                LDA        #R.RECAL             ;SETUP THE
* R.RECAL MUST BE NON-ZERO!! (SEE BELOW)
                STA        RECALCNT             ; RECAL TRIES
                NOP                             ; PAD ONE BYTE
                STA        E1908                ; A-REG MUST BE NON-ZERO !!!
* E1908 = NON-ZERO LOCKOUT MOUSE
*
                LDY        D.UNITNUM            ;ARE WE ON-TRACK?
                LDA        TRACK
                CMP        DRVTRACK,Y
                BEQ        SOUGHT               ;=>IF SO, FORGET SEEK & DELAY!
*
* WAIT BEFORE STEPPING:
*
                LDA        SEEKWAIT             ;SEEK DELAY NEEDED?
                BEQ        GOSEEK               ;=>NAW...
                LDA        #0
                STA        SEEKWAIT             ;CLEAR THE FLAG
                LDA        #4                   ;ADD SEEKDELAY TO
                JSR        ADDTIME              ; THE TOTAL UPTIME(S)
                TAY                             ;4*25 MS DELAY
SEEKDEL         EQU        *
                LDA        #0
                JSR        MSWAIT
                DEY
                BNE        SEEKDEL
*
* ISSUE THE SEEK:
*
GOSEEK          EQU        *
                LDA        TRACK                ;GET DESTINATION TRACK
                JSR        MYSEEK               ;=>..AND YOU SHALL FIND...
*
SOUGHT          EQU        *
                LDA        IRQMASK              ;SET IRQ MASK FOR
                STA        IMASK                ; CORE ROUTINES
                LDA        #R.IRQ               ;SETUP IRQ RETRIES
                STA        INTRTRY
                LDA        #R.IOERR             ; AND ERROR RETRIES
                STA        RETRYCNT
*
* DELAY FOR ANY REMAINING MOTOR-UP TIME:
*
MDELAY          EQU        *
                LDA        MONTIMEH             ;ANY TIME REMAINING?
                BPL        FINDIT               ;=>NO, WE'RE UP TO SPEED.
                LDA        #1                   ;YES, SO BUMP A SLICE OF
                JSR        ADDTIME              ; UPTIME WHILE WE WAIT
                LDA        #0
                JSR        MSWAIT
                JMP        MDELAY               ;=>GO TILL ENOUGH
*
* FIND THE DESIRED SECTOR:
*
* NOTE: FINDSECT RETURNS WITH
*       IRQ INHIBITED!
*
FINDIT          EQU        *
                PHP                             ;INHIBIT IRQ WHILE
```

```
                 SEI                        ; MESSING WITH VBL FLAGS
                 LDA      E.IER             ;DISABLE VBL IRQ
                 AND      #$18              ; DURING SECTOR I/O
                 STA      E.IER
                 ORA      #$80              ;FOR 'SET' LATER
                 STA      VBLSAVE
                 PLP                        ;RESTORE IRQ STATUS
                 JSR      FINDSECT          ;FIND ME PLEASE
                 BCS      TRYRECAL          ;=>NO? RECAL OR GIVE UP!
                 LDX      #$60              ;SET UP SLOT FOR CORE RTNS
                 LDA      D.COMMAND         ;WHAT'S YOUR PLEASURE?
                 BNE      SIOWRITE          ;=>WRITE
*
                 REP      40
* READ A SECTOR:
*
                 JSR      READ              ;READ THAT SECTOR
                 JSR      FIXIRQ            ;ENABLE IRQ IF OK
                 LDA      VBLSAVE           ;ALLOW VBL DURING
                 STA      E.IER             ; POSTNIB
                 BCS      BADIO             ;=>I/O ERR OR IRQ
                 LDA      E.REG             ;SET 2MHZ FOR POSTNIB
                 AND      #$7F
                 STA      E.REG
                 JSR      POSTNIB           ;POSTNIB/CHECKSUM
                 BCS      IORETRY           ;=>I/O ERR:BAD CHKSUM
                 JMP      SIOGOOD           ;=>GOOD READ
*
                 REP      40
* WRITE A SECTOR:
*
SIOWRITE         EQU      *
                 JSR      WRITE             ;WRITE THE DATA
                 JSR      FIXIRQ            ;RE-ENABLE IRQ IF OK
                 LDA      VBLSAVE           ;RESTORE
                 STA      E.IER             ; VBL IRQ
                 BCC      SIOGOOD           ;=>GOOD WRITE
                 BVC      SIOWPROT          ;=>WRITE PROTECTED
*
                 REP      40
* IT DIDN'T GO WELL FOR US:
*
BADIO            EQU      *
                 DO       1-REV0ROM         ;FOR REV1
                 BVS      FINDIT            ;=>IRQ. JUST RETRY IT.
                 ELSE                       ;FOR REV0
*
* THE REV1 ROM TAKES CARE OF THE
*  IRQ RETRY COUNT, BUT REV0 DOESN'T:
*
                 BVC      IORETRY           ;=>I/O ERROR. RETRY IT
                 LDA      ROMREV            ;WHICH ROM?
                 BNE      FINDIT            ;=>REV1. HE DOES IT.
                 LDA      INTRTRY           ;REV0. OUT OF RETRIES?
                 BPL      BADIO2            ;=>NO.
                 STA      IMASK             ;SET HI BIT FOR IRQ MASK
BADIO2           EQU      *
                 DEC      INTRTRY           ;ONE LESS RETRY
                 JMP      FINDIT            ;=>RETRY AFTER IRQ
                 FIN
*
* RETRY AFTER AN I/O ERROR:
*
IORETRY          EQU      *
                 DEC      RETRYCNT          ;ANY RETRIES LEFT?
                 BNE      FINDIT            ;=>YEAH, RETRY AFTER ERROR
*
* RETRIES EXHAUSTED. RECALIBRATE:
*
TRYRECAL         EQU      *
                 LDA      VBLSAVE           ;ALLOW VBL IF RECAL
                 STA      E.IER             ; OR UNRECOVERABLE ERROR
                 DEC      RECALCNT          ;HAVE WE RECALIBRATED YET?
                 BMI      SIOERR            ;=>YUP. WE'RE DEAD.
                 JSR      RECAL             ;NO, TRY OUR LUCK
                 LDY      D.UNITNUM         ;ARE WE ON-TRACK?
                 LDA      TRACK
                 CMP      DRVTRACK,Y
                 BNE      NOTSAME
                 JMP      SOUGHT            ;=>IF SO, FORGET RESEEK
NOTSAME          EQU      *
```

```
                 JMP      GOSEEK           ;TRY AGAIN ON TARGET TRACK
*
                 REP      40
SIOERR           EQU      *
                 LDA      #XIOERROR        ;RETURN CODE
                 SEC                       ;INDICATE HARD ERROR
                 BCS      SIORET
SIOWPROT         EQU      *
                 LDA      #XNOWRITE        ;RETURN CODE
                 SEC                       ;INDICATE HARD ERROR
                 BCS      SIORET
SIOGOOD          EQU      *
                 LDA      #0
                 CLC                       ;INDICATE GOOD COMPLETION
SIORET           LDX      #0               ; SAY OK TO MOUSE
                 STX      E1908            ; WITH THIS GLOBAL $1908
                 RTS
                 PAGE
                 REP      40
* NAME    : FINDSECT
* FUNCTION: LOCATE A DESIRED SECTOR
* INPUT   : IBTRK, IBSECT SETUP
* RETURNS : CARRY CLEAR IF OK,
*         : CARRY SET   IF ERROR.
* DESTROYS: ALL REGISTERS & 'TEMP'
* NOTE    : RETURNS WITH IRQ DISABLED IF NO ERROR!
                 REP      40
*
FINDSECT         EQU      *
                 LDA      #R.FIND*16       ;SETUP NUMBER OF REVS
                 STA      RETRYADR         ; ALLOWED TO FIND SECTOR
                 LSR      TEMP             ;COMPUTE LATENCY FIRST TIME THRU
FINDSEC2         EQU      *
                 LDX      #$60             ;FAKE SLOT FOR CORE ROUTINES
                 JSR      RDADR            ;GET NEXT ADDRESS FIELD
                 BCS      RDADERR          ;=>UGH! AN ERROR!
*
* MAKE SURE WE'RE ON THE CORRECT TRACK:
*
                 LDA      TRACK            ;IS IT
                 CMP      CSSTV+2          ; CORRECT TRACK?
                 BNE      FINDERR          ;=>NO?!? IT'S USELESS!
                 LDA      SECTOR           ;IS IT
                 CMP      CSSTV+1          ; DESIRED SECTOR?
                 BEQ      FINDGOOD         ;=>YEAH. GOT IT!
*
* COMPUTE LATENCY. EACH TWO-SECTOR
*  DISTANCE IS 25 MS OF UPTIME.
*
                 LDA      TEMP             ;LATENCY ALREADY COMPUTED?
                 BMI      RDADERR          ;=>YES.
                 LDA      SECTOR           ;HOW FAR AWAY IS OUR
                 SEC                       ; DESIRED SECTOR?
                 ROR      TEMP             ;PREVENT RECOMPUTATION
                 SBC      CSSTV+1
                 AND      #$0F
                 LSR      A                ;EACH 2-SECTORS IS 25 MS
                 JSR      ADDTIME
*
* KEEP LOOKING TILL WE FIND IT:
*
RDADERR          EQU      *
                 JSR      FIXIRQ           ;ENABLE IRQ IF APPROPRIATE
                 DEC      RETRYADR         ;ANY RETRIES LEFT?
                 BEQ      FINDERR          ;=>NO, WE CAN'T FIND IT.
*
* COMPENSATE FOR A BUG IN RDADR: IF WE TRY
*  TO CALL RDADR AGAIN BEFORE THE DATA MARK
*  GOES BY, THEN RDADR WILL ACCIDENTALLY CALL
*  THAT AN ERROR. WE CAN AVOID THIS 'FAKE'
*  ERROR BY DELAYING PAST THE DATA MARK.
                 LDY      #200             ;1 MS IS PLENTY
ADRDELAY         EQU      *
                 DEY
                 BNE      ADRDELAY
                 JMP      FINDSEC2         ;=>NOW TRY LOOKING AGAIN
*
                 REP      40
FINDGOOD         EQU      *
                 LDA      #0               ;CLEAR VOLNUM OUT OF
                 STA      MONTIMEH         ; MOTORTIME!
```

```
                CLC                             ;INDICATE NO ERROR
                RTS
*
FINDERR         EQU     *
                JSR     FIXIRQ                  ;ENABLE IRQ IF APPROPRIATE
                LDA     #0                      ;CLEAR VOLNUM OUT OF
                STA     MONTIMEH                ; MOTORTIME!
                SEC                             ;INDICATE THE ERROR
                RTS

                CHN     DISK3.USEL.SRC

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


```

End of File -- Lines: 255 Characters: 8245

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

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

                SBTL       'SOS 1.1  DISK /// DRIVER'
TEST            EQU        0                    ;FOR FUNNY-MODE TESTING
                INCLUDE    SOSORG,6,1,254
                DO         TEST
                ORG        $2000
                ELSE
                REL
                ORG        ORGDISK3
                FIN
ZZORG           EQU        *
                CHR        '-'
                MSB        OFF
*
                REP        40
*    COPYRIGHT (C) APPLE COMPUTER INC.
*          ALL RIGHTS RESERVED
                REP        40
*
REV0ROM         EQU        0                    ;1=SUPPORT REV0 ROM
*
                DO         1-TEST
                ENTRY      DIB1                 ;DIB1
                ENTRY      DIB2                 ;DIB2
                ENTRY      DIB3                 ;DIB3
                ENTRY      DIB4                 ;DIB4
                ENTRY      SEEKDSK3             ;SEEK CURRENT DRIVE
*
                EXTRN      SYSERR
*
                EXTRN      XREQCODE
                EXTRN      XBADOP
                EXTRN      XNODRIVE
                EXTRN      XIOERROR
                EXTRN      XNOWRITE
                EXTRN      XBYTECNT
                EXTRN      XBLKNUM
                EXTRN      XDISKSW
                EXTRN      XCTLCODE
*
                EXTRN      E1908                ; GLOBAL FLAG FOR MOUSE DRIVER
* TO SAY WE CANNOT BE INTERRUPTED
*
                ELSE
XREQCODE        EQU        $20
XBADOP          EQU        $26
XNODRIVE        EQU        $28
XIOERROR        EQU        $27
XNOWRITE        EQU        $2B
XBYTECNT        EQU        $2C
XBLKNUM         EQU        $2D
XDISKSW         EQU        $2E
XCTLCODE        EQU        $21
                FIN
                PAGE
* DISK /// CONTROLLER EQUATES:
*
*       MOTOR SELECT BITS:
*
*    DRIVE    INT   EXT1   EXT2
*    -----    ---   ----   ----
*     .D1      1     X      X
*     .D2      X     0      1
*     .D3      X     1      0
*     .D4      X     1      1
*
MS.INT          EQU        $C0D4                ;MOTOR   SELECT:INTERNAL DRIVE
MD.INT          EQU        $C0D5                ;MOTOR DESELECT:INTERNAL DRIVE
*
MS.EXT1         EQU        $C0D3                ;MOTOR   SELECT:EXTERNAL DRIVE
MS.EXT2         EQU        $C0D1                ;MOTOR   SELECT:EXTERNAL DRIVE
MD.EXT1         EQU        $C0D2                ;MOTOR DESELECT:EXTERNAL DRIVE
```

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

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

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

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

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

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

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

                SBTL       "SOS 1.1  FILE MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGFMGR
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* FILE MANAGER (VERSION = 1.1O   )
*              (DATE    = 8/04/81)
*
* THIS MODULE IS ENTERED FROM THE SYSTEM CALL MANAGER, AND
* IS RESPONSIBLE FOR SWITCHING TO EITHER THE BLOCK FILE
* MANAGER, OR THE CHARACTER FILE MANAGER.
*
                REP        60
*
                ENTRY      FMGR
                ENTRY      LEVEL
*
                EXTRN      BFMGR
                EXTRN      CFMGR
                EXTRN      SYSERR
                EXTRN      SERR
                EXTRN      BADPATH
                EXTRN      FNFERR
                EXTRN      LVLERR
*
F.TPARMX        EQU        $A0                   ; LOC OF FILE SYSTEM CALL PARMS
OPEN            EQU        $8
CLOSE           EQU        $C
SETLEVEL        EQU        $12
GETLEVEL        EQU        $13
F.REQCODE       EQU        F.TPARMX
F.LEVEL         EQU        F.TPARMX+$1
PATHNAME        EQU        F.TPARMX+$1
REFNUM          EQU        F.TPARMX+$1
PERIOD          EQU        $2E
LEVEL           DFB        $1
                PAGE
                REP        60
*
* FILE MANAGER
*
                REP        60
FMGR            EQU        *
*
                LDA        F.REQCODE
                CMP        #OPEN
                BCC        FMGR010
                BEQ        FMGR020
                CMP        #CLOSE
                BCC        FMGR030
                BEQ        FMGR040
                CMP        #SETLEVEL
                BEQ        SLEVEL
                CMP        #GETLEVEL
                BEQ        GLEVEL
*
FMGR010         JMP        BFMGR                 ; EXIT
*
FMGR020         LDY        #1
                LDA        (PATHNAME),Y
                CMP        #PERIOD
                BNE        FMGR010
                JSR        CFMGR
                BCC        FMGR024
                LDA        SERR
```

```
                CMP     #FNFERR
                BEQ     FMGR026
FMGR024         RTS                         ; EXIT
*
FMGR026         LDA     #0
                STA     SERR
                JMP     BFMGR               ; EXIT
*
FMGR030         LDA     REFNUM
FMGR031         BPL     FMGR010
                JMP     CFMGR               ; EXIT
*
FMGR040         LDA     REFNUM
                BNE     FMGR031
                JSR     BFMGR               ; CLOSE (0)
                JMP     CFMGR               ; EXIT
*
SLEVEL          LDA     F.LEVEL
                BEQ     LVL.ERR
                CMP     #4
                BCS     LVL.ERR
                STA     LEVEL
                RTS
LVL.ERR         LDA     #LVLERR
                JSR     SYSERR
*
GLEVEL          LDY     #0
                LDA     LEVEL
                STA     (F.LEVEL),Y
                RTS
*
                LST     ON
ZZEND           EQU     *
ZZLEN           EQU     ZZEND-ZZORG
                IFNE    ZZLEN-LENFMGR
                FAIL    2,"SOSORG           FILE IS INCORRECT FOR FMGR"
                FIN

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


```

End of File -- Lines: 119 Characters: 2884

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

*************************************************************************
* APPLE /// SOS 1.3 SOURCE CODE FILE: FNDFIL
*************************************************************************
* ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
                PAGE
*
*
FINDFILE        JSR        LOOKFILE             ; SEE IF FILE EXISTS
                BCS        NOFIND               ; BRANCH IF AN ERROR WAS ENCOUNTERED
MOVENTRY        LDY        H.ENTLN              ; MOVE ENTIRE ENTRY INFO TO A SAFE AREA
MOVENT1         LDA        (DRBUFPL),Y
                STA        DFIL+D.STOR,Y
                DEY
                BPL        MOVENT1
                LDA        #0                   ; TO INDICATE ALL IS WELL
NOFIND          RTS                             ; RETURN CONDITION CODES.
                PAGE
*
*
LOOKFILE        JSR        PREPROOT             ; FIND VOLUME AND SET UP OTHER BORING STUFF
                BCS        FNDERR               ; PASS BACK ANY ERROR ENCOUNTERED
                LDY        #0                   ; TEST TO SEE IF ONLY ROOT WAS SPECIFIED.
                LDA        (PATHNML),Y
                BNE        LOOKFIL0             ; BRANCH IF MORE THAN ROOT.
                LDA        #GBUF/256            ;  OTHERWISE, REPORT A BADPATH ERROR
                STA        DRBUFPH              ; (BUT FIRST CREATE A PHANTOM ENTRY FOR OPEN)
                LDA        #4
                STA        DRBUFPL
                LDY        #D.AUXID             ; FIRST MOVE IN ID, AND DATE STUFF.
PHANTM1         LDA        (DRBUFPL),Y
                STA        DFIL,Y
                DEY
                CPY        #D.CREDT-1
                BNE        PHANTM1
PHANTM2         LDA        ROOTSTUF-D.FILID,Y
                STA        DFIL,Y
                DEY
                CPY        #D.FILID-1
                BNE        PHANTM2
                LDA        #DIRTYP*$10          ; FAKE DIRECTORY FILE
                STA        DFIL+D.STOR
                LDA        #BADPATH             ; (CARRY IS SET)
                RTS
*
ROOTSTUF        DFB        0,2,0,4
                DFB        0,0,8,0
*
LOOKFIL0        LDA        #0                   ; RESET FREE ENTRY INDICATOR
                STA        NOFREE
                SEC                             ; INDICATE THAT THE DIRECTORY TO BE SEARCHED HAS HEADER IN THIS BLOCK
LOOKFIL1        LDA        #0                   ; RESET ENTRY COUNTER
                STA        TOTENT
                JSR        LOOKNAM              ; LOOK FOR NAME POINTED TO BY 'PATHNML'
                BCC        NAMFOJMP             ; BRANCH IF NAME WAS FOUND.
                LDA        ENTCNTL              ; HAVE WE LOOKED AT ALL OF THE
                SBC        TOTENT               ;  ENTRIES IN THIS DIRECTORY?
                BCC        DCRENTH              ; MAYBE, CHECK HI COUNT.
                BNE        LOOKFIL2             ; NO, READ NEXT DIRECTORY BLOCK
                CMP        ENTCNTH              ; HAS THE LAST ENTRY BEEN LOOKED AT (ACC=0)
                BEQ        ERRFNF               ; YES, GIVE 'FILE NOT FOUND' ERROR.
                BNE        LOOKFIL2             ; BRANCH ALWAYS.
DCRENTH         DEC        ENTCNTH              ; SHOULD BE AT LEAST 1
                BPL        LOOKFIL2             ; (THIS SHOULD BE BRANCH ALWAYS...)
ERRDIR          LDA        #DIRERR              ; REPORT DIRECTORY MESSED UP.
FNDERR          SEC                             ; INDICATE ERROR HAS BEEN ENCOUNTERED.
                RTS
NAMFOJMP        JMP        NAMFOUND             ; AVOID BRANCH OUT OF RANGE
*
                PAGE
LOOKFIL2        STA        ENTCNTL              ; KEEP RUNNING COUNT
                LDA        #GBUF/256            ; RESET INDIRECT POINTER
                STA        DRBUFPH
                LDA        GBUF+2               ; GET LINK TO NEXT DIRECTORY BLOCK
                BNE        NXTDIR0              ; (IF THERE IS ONE)
                CMP        GBUF+3               ; ARE BOTH ZERO, I.E. NO LINK?
```

```
                BEQ     ERRDIR          ; IF SO, THEN NOT ALL ENTRIES WERE ACCOUNTED FOR.
NXTDIR0         STA     BLOKNML
                LDA     GBUF+3
                STA     BLOKNMH
                JSR     RDGBUF          ; GO READ THE NEXT LINKED DIRECTORY IN.
                BCC     LOOKFIL1        ; BRANCH IF NO ERROR.
                RTS                     ; RETURN ERROR (IN ACCUMULATOR).
TELFREEX        JMP     TELFREE
*
FNF0X           JMP     FNF0            ; AVOID BRANCH OUT OF RANGE
*
CFLAG           DS      1               ; AM I CREATING?
TTSAVE          DS      2               ; CURRENT BLOCK ADDR
BLOKSAVE        DS      2               ; PARENT DIR ADDR
*
ERRFNF          LDA     NOFREE          ; WAS ANY FREE ENTRY FOUND?
                BNE     FNF0X
                LDA     GBUF+2          ; TEST LINK
                BNE     TELFREEX
                CMP     GBUF+3          ; IF BOTH ARE ZERO, THEN GIVE UP
                BNE     TELFREEX        ; BRANCH IF NOT LAST DIR BLOCK
                LDA     CFLAG           ; DOING A CREATE?
                BEQ     FNF0X           ; NO, SIMPLY REPORT NOT FOUND
*
* EXTEND THE DIRECTORY BY A BLOCK
*
                LDA     BLOKSAVE        ; BUT NOT
                ORA     BLOKSAVE+1      ;    IF A ROOT DIRECTORY!
                BEQ     FNF0X           ;        FORU BLOCKS HARD CODED
                LDA     TTLINK          ; FETCH CURRENT DIRECTORY
                STA     TLINK           ; ADDR (GBUF)
                LDA     TTLINK+1        ; AND ALLLOCATE A NEW
                STA     TLINK+1         ; BY LINKING TO CURRENT
                JSR     DIRWRT
                BCS     FNF0            ; RATS! NO SPACE SAY "DIRFULL"
*
* SAVE CURRENT BLOCK ADDR
*
                LDA     TTLINK
                STA     TTSAVE
                LDA     TTLINK+1
                STA     TTSAVE+1
*
* FETCH DESCENDENT
*
                LDA     GBUF+2
                STA     BLOKNML
                LDA     GBUF+3
                STA     BLOKNMH
                JSR     ZERGBUF         ; INIT THE NEW DIR BLOCK
*
* AND INSERT BACK POINTER
* TO "CURRENT BLOCK"
*
                LDA     TTSAVE
                STA     GBUF
                LDA     TTSAVE+1
                STA     GBUF+1
                JSR     WRTGBUF
                BCS     ERTS
*
* UPDATE DIR'S HEADER IN PARENT
*
                LDA     BLOKSAVE
                STA     BLOKNML         ; PREPARE TO READ PARENT
                LDX     BLOKSAVE+1
                STX     BLOKNMH
                JSR     RDGBUF          ; FETCH PARENT
                LDY     #D.USAGE        ; BUMP BLOCKS USED BY HEADER
                LDA     (DEBUPTR),Y
                SEC
                ADC     #0              ; BY JUST ONE BLOCK
                STA     (DEBUPTR),Y
                INY
                LDA     (DEBUPTR),Y     ; TWO BYTE BLOCKS USED
                ADC     #0
                STA     (DEBUPTR),Y
                LDY     #D.EOF+1        ; INCREASE EOF BY $200
                LDA     (DEBUPTR),Y
                CLC
                ADC     #2
```

```
                STA     (DEBUPTR),Y
                INY
                LDA     (DEBUPTR),Y
                ADC     #0
                STA     (DEBUPTR),Y
                JSR     WRTGBUF             ; REWRITE PARENT DIR BLOCK
                LDA     TTSAVE+1            ; REFETCH CURRENT DIR BLOCK
                STA     BLOKNMH
                LDA     TTSAVE
                STA     BLOKNML
                JSR     RDGBUF              ; BACK FROM THE SHADOWS AGAIN
                JMP     ERRFNF              ; VOILA! WE HAVE EXTENDED THE DIRECTORY!
*
TELFREE         STA     D.ENTBLK
                LDA     GBUF+3
                STA     D.ENTBLK+1          ; ASSUME FIRST ENTRY OF NEXT BLOCK
                LDA     #1                  ;  IS FREE FOR USE.
                STA     D.ENTNUM
                STA     NOFREE              ;  MARK D.ENTNUM AS VALID (FOR CREATE)
FNF0            LDY     #0                  ; TEST FOR 'FILE NOT FOUND' VERSUS 'PATH NOT FOUND'
                LDA     (PATHNML),Y
                TAY
                INY
                LDA     (PATHNML),Y         ; IF NON-ZERO THEN 'PATH NOT FOUND'
ERRPATH1        SEC                         ; IN EITHER CASE, INDICATE ERROR.
                BEQ     FNF1
                LDA     #PATHNOTFND         ; REPORT NO SUCH PATH.
ERTS            RTS
FNF1            LDA     #FNFERR             ; REPORT FILE NOT FOUND.
                RTS
                PAGE
*
NAMFOUND        LDA     (PATHNML),Y         ; (Y=0)
                SEC
                ADC     PATHNML             ; TEST FOR LAST NAME IN PATH
                TAY                         ; IF ZERO, THEN THAT WAS LAST NAME
                CLC                         ; TO INDICATE SUCCESS
                LDA     PATHBUF,Y
                BEQ     FILFOUND
*NOW CHANGE THE PATHNAME POINTER TO POINT AT THE NEXT NAME IN THE PATH
                STY     PATHNML
                LDA     DRBUFPL             ; SAVE PARENTS
                STA     DEBUPTR             ; ENTRY POINTER
                LDA     DRBUFPH
                STA     DEBUPTR+1           ; IN CASE ENTRY ON PAGE 2
                LDA     BLOKNML             ; ADDRESS (DIR EXTEND)
                STA     BLOKSAVE
                LDA     BLOKNMH
                STA     BLOKSAVE+1
                LDY     #D.STOR             ; BE SURE THIS IS A DIRECTORY ENTRY
                LDA     (DRBUFPL),Y         ; HIGH NIBBLE WILL TELL
                AND     #$F0
                CMP     #DIRTYP*16          ; IS IT A SUB-DIRECTORY?
                BNE     ERRPATH1            ; REPORT THE USER'S MISTAKE
                LDY     #D.FRST             ; GET ADDRESS OF FIRST SUB-DIRECTORY BLOCK
                LDA     (DRBUFPL),Y
                STA     BLOKNML             ; (NO CHECKING IS DONE HERE FOR A VALID
                INY                         ;  BLOCK NUMBER... )
                STA     D.HEAD              ; SAVE AS FILE'S HEADER BLOCK TOO.
                LDA     (DRBUFPL),Y
                STA     BLOKNMH
                STA     D.HEAD+1
                JSR     RDGBUF              ; READ SUB-DIRECTORY INTO GBUF
                BCS     FNDERR1             ; RETURN IMMEDIATELY ANY ERROR ENCOUNTERED.
                LDA     GBUF+HCENT+4        ; GET THE NUMBER OF FILES
                STA     ENTCNTL             ;  CONTAINED IN THIS DIRECTORY
                LDA     GBUF+HCENT+5
                STA     ENTCNTH
                LDA     GBUF+HCMP+4         ; TEST BACKWARD COMPATIBILITY
                BEQ     MOVHEAD
ERRCOMP         LDA     #CPTERR             ; TELL THEM THIS DIRECTORY IS NOT COMPATABLE
NONAME          EQU     *
FNDERR1         SEC
                RTS
MOVHEAD         JSR     MOVHED0             ; MOVE INFO ABOUT THIS DIRECTORY
                JMP     LOOKFIL0            ; DO NEXT LOCAL PATHNAME
*
MOVHED0         LDX     #$A                 ; MOVE INFO ABOUT THIS DIRECTORY
MOVHED1         LDA     GBUF+HCRDT+4,X
                STA     H.CREDT,X
                DEX
```

```
                BPL     MOVHED1
                RTS
*
                PAGE
*
*
FILFOUND        EQU     *
ENTADR          LDA     H.MAXENT            ; FIGURE OUT WHICH IS ENTRY NUMBER THIS IS.
                SEC
                SBC     CNTENT              ; MAX ENTRIES - COUNT ENTRIES + 1 = ENTRY NUMBER
                ADC     #0                  ; (CARRY IS/WAS SET)
                STA     D.ENTNUM
                LDA     BLOKNML
                STA     D.ENTBLK
                LDA     BLOKNMH             ; AND INDICATE BLOCK NUMBER OF THIS DIRECTORY.
                STA     D.ENTBLK+1
                CLC
                RTS
*
LOOKNAM         LDA     H.MAXENT            ; RESET COUNT OF FILES PER BLOCK
                STA     CNTENT
                LDA     #GBUF/256
                STA     DRBUFPH
                LDA     #4
LOKNAM1         STA     DRBUFPL             ; RESET INDIRECT POINTER TO GBUF
                BCS     LOKNAM2             ; BRANCH IF THIS BLOCK CONTAINS A HEADER
                LDY     #D.STOR
                LDA     (DRBUFPL),Y         ; GET LENGTH OF NAME IN DIRECTORY
                BNE     ISNAME              ; BRANCH IF THERE IS A NAME.
                LDA     NOFREE              ; TEST TO SEE IF A FREE ENTRY HAS BEEN DECLARED.
                BNE     LOKNAM2             ; YES BUMP TO NEXT ENTRY
                JSR     ENTADR              ; SET ADDRESS FOR CURRENT ENTRY
                INC     NOFREE              ; INDICATE A FREE SPOT HAS BEEN FOUND
                BNE     LOKNAM2             ; BRANCH ALWAYS.
*
ISNAME          AND     #$F                 ; STRIP TYPE (THIS IS CHECKED BY 'FILFOUND')
                INC     TOTENT              ; (BUMP COUNT OF VALID FILES FOUND)
                CMP     (PATHNML),Y         ; ARE BOTH NAMES OF THE SAME LENGTH?
                BNE     LOKNAM2             ; NO, BUMP TO NEXT ENTRY
                TAY
CMPNAME         LDA     (DRBUFPL),Y         ; COMPARE NAMES LETTER BY LETTER
                CMP     (PATHNML),Y
                BNE     LOKNAM2
                DEY                         ; HAVE ALL LETTERS BEEN COMPARED?
                BNE     CMPNAME             ; NO, CONTINUE..
                CLC                         ; BY GOLLY, WE GOT US A MATCH!
                RTS
*
LOKNAM2         DEC     CNTENT              ; HAVE WE CHECKED ALL POSSIBLE ENTRIES IN THIS BLOCK?
                BEQ     NONAME              ; YES, GIVE UP.
                LDA     H.ENTLN             ; ADD ENTRY LENGTH TO CURRENT POINTER
                CLC
                ADC     DRBUFPL
                BCC     LOKNAM1             ; BRANCH IF WE'RE STILL IN THE FIRST PAGE.
                INC     DRBUFPH             ; LOOK ON SECOND PAGE
                CLC                         ; CARRY SHOULD ALWAYS BE CLEAR BEFORE LOOKING AT NEXT.
                BCC     LOKNAM1             ; BRANCH ALWAYS...
                PAGE
*
*
PREPROOT        JSR     FINDVOL             ; FIND CORRECT VOLUME AND DEVICE NUMBER
                BCC     ROOT1               ; BRANCH IF IT WAS FOUND.
ROOT0           JSR     LOOKVOL             ; OTHERWISE LOOK ON ALL DEVICES.
                BCS     SRITZ               ; CAN'T FIND IT.
ROOT1           LDA     #0                  ; ZERO OUT DIRECTORY TEMPS
                LDY     #42                 ; (DECIMAL)
CLRDSP          STA     D.DEV,Y
                DEY
                BPL     CLRDSP
                LDY     #VCBDEV             ; SET UP DEVICE NUMBER
                LDA     (VCBPTR),Y
                STA     DEVNUM
                STA     D.DEV               ; FOR FUTURE REFERENCE
                INY
                LDA     (VCBPTR),Y          ; GET CURRENT STATUS OF THIS VOLUME
                STA     V.STATUS
                LDY     #VCBROOT            ; GET BLOCK ADDRESS OF ROOT DIRECTORY TOO.
                LDA     (VCBPTR),Y
                STA     BLOKNML
                STA     D.HEAD              ; PRESERVE AS HEADER
                INY
```

```
                 LDA      (VCBPTR),Y
                 STA      BLOKNMH
                 STA      D.HEAD+1
                 JSR      RDGBUF              ; GO READ IN ROOT
                 BCC      ROOT2               ; BRANCH IF NO ERROR
                 PHA                          ; SAVE ERROR CODE
                 LDY      #VCBSTAT            ; CHECK THIS BUGGER FOR AN OPEN FILE.
                 LDA      (VCBPTR),Y
                 ASL      A                   ; (SHIFT OPEN STATUS INTO CARRY)
                 PLA                          ; GET ERROR CODE AGAIN
                 BCS      ROOTERR             ;  BRANCH IF ERROR NEEDS TO BE REPORTED
                 BNE      ROOT0               ; OTHERWISE, LOOK ELSEWHERE (BRANCH ALWAYS).
*
ROOT2            JSR      CHKROOT             ; VERIFY ROOT NAME
                 BEQ      ROOT3               ; BRANCH IF MATCHED.
                 LDY      #VCBSTAT            ; TEST FOR OPEN FILES ON THIS VOLUME BEFORE
                 LDA      (VCBPTR),Y          ;  LOOKING FOR IT ELSEWHERE.
                 BPL      ROOT0
                 JSR      USRREQ              ;  REQUEST USER MOUNT VOLUME
                 BCC      ROOT1               ;  USER SAID S/HE DID-- CHECK IT
                 LDA      #VNFERR             ; REPORT VOLUME NOT FOUND ERR IF REFUSE TO INSERT
SRITZ            RTS
*
                 PAGE
ROOT3            LDY      #$F                 ; (NOTE: X CONTAINS THE LENGTH OF THE ROOT NAME)
ROOTINFO         LDA      GBUF+HCRDT+3,Y      ; SAVE HEADER INFO.
                 STA      V.STATUS,Y
                 DEY
                 BNE      ROOTINFO            ; LOOP TIL ALL 15 BYTES MOVED
                 LDA      H.FCNT
                 STA      ENTCNTL
                 LDA      H.FCNT+1
                 STA      ENTCNTH
                 TXA                          ; NOW THAT ROOT IS IDENTIFIED, ADJUST
                 SEC                          ;  PATH NAME POINTER TO NEXT NAME IN THE PATH
                 ADC      PATHNML
                 STA      PATHNML
                 CLC                          ; INDICATE NO ERROR
ROOTERR          RTS
*
*
CHKROOT          LDY      #0                  ; GET LENGTH OF NAME
                 LDA      (PATHNML),Y
                 TAY
                 TAX                          ; SAVE IN X FOR LATTER ADJUSTMENT TO PATH POINTER
                 EOR      GBUF+4
                 AND      #$F                 ; DOES PATHNAME HAVE SAME LENGTH AS DIRECTORY NAME?
                 BNE      NOTROOT             ; BRANCH IF NOT
CKROOT1          LDA      (PATHNML),Y         ; COMPARE CHARACTER BY CHARACTER
                 CMP      GBUF+4,Y
                 BNE      NOTROOT
                 DEY
                 BNE      CKROOT1             ; LOOP UNTIL ALL CHARACTERS MATCH
NOTROOT          RTS
*
                 PAGE
FINDVOL          LDA      #VCB/256            ; SEARCH VCB FOR VOLUME NAME
                 STA      VCBPTR+1
                 LDA      #0
                 STA      D.DEV
                 STA      VCBPTR
FNDVOL1          PHA                          ; SAVE LAST SEARCH POSITION
                 TAX
                 LDY      #0                  ; (INDEX TO PATHNAME POINTER)
                 LDA      VCB,X               ; GET LENGTH OF VOLUME NAME TO COMPARE
                 BEQ      NXTVCB              ; BRANCH IF VCB ENTRY IS EMPTY
                 CMP      (PATHNML),Y         ; ARE NAMES OF SAME LENGTH?
                 BNE      NXTVCB              ; NO, INDEX NEXT VCB
                 CLC                          ; SCAN NAME BACKWARDS
                 TAY
                 TXA
                 ADC      VCB,X
                 TAX                          ; NOW BOTH INDEXES POINT TO LAST CHARACTER OF THE NAMES TO COMPARE
VOLNAM           LDA      (PATHNML),Y
                 CMP      VCB,X
                 BNE      NXTVCB
                 DEX
                 DEY
                 BNE      VOLNAM              ; CHECK ALL CHARACTERS
                 PLA                          ; SINCE A MATCH IS FOUND
                 STA      VCBPTR              ;  SET UP INDEX TO VCB ENTRY
```

```
                TAX
                LDA     VCB+VCBSWAP,X       ;  BRANCH IF
                BEQ     FOUNDVOL            ;  VOLUME NOT SWAPPED
                JSR     SWAPIN              ;  IF USER REALLY WANTS IT, THEN BRING IN IF SWAPPED
                BCC     FOUNDVOL            ;  BRANCH IF SUCCESS
                LDA     #XIOERROR           ;  USER REFUSES TO MOUNT
                RTS
FOUNDVOL        CLC                         ; INDICATE VOLUME FOUND
                RTS
*
NXTVCB          PLA                         ; GET CURRENT INDEX AGAIN.
                CLC
                ADC     #VCBSIZE            ; VCB ENTRY LENGTH.
                BCC     FNDVOL1             ; BRANCH IF THER IS ANOTHER TO CHECK
                RTS                         ; RETURN WITH CARRY SET TO SHOW FAILURE.
                PAGE
*
*
LOOKVOL         LDX     #12                 ; (1) COUNT+(12)DEVICE LIST
LOOKVOL1        LDA     BLKDLST,X           ;  EXTRN
                STA     SCRTCH,X            ;  MY CHANGEABLE COPY
                DEX
                BPL     LOOKVOL1            ;  WORK BACKWARDS SO
                STA     TOTDEVS             ;  ENTRY ZERO IS TOTAL DEVICES LISTED
                INX                         ;  MAKE XREG = ZERO
LOKDEV1         INX
                STX     SCRTCH
                LDA     SCRTCH,X
                CMP     D.DEV
                BEQ     NXTDEV              ; DON'T LOOK AGAIN ON A DRIVE THAT HAS BEEN CHECKED
                STA     DEVNUM              ; CHECK FOR DEVICE ALREADY LOGGED IN A VCB
                JSR     DEVVCB              ; (CARRY CLEAR IF IT'S THERE)
                BCC     LOKVOL1
                LDA     #0                  ; FIND A FREE VCB TO LOG THIS GUY IN
ENTVCB          TAX                         ; INDEX TO NEXT VCB ENTRY
                LDA     VCB,X
                BEQ     FREEVCB             ; FOUND A FREE SPOT.
                TXA                         ; NOW INDEX TO NEXT, AND KEEP LOOKIN
                CLC
                ADC     #VCBSIZE            ; (EACH VCB ENTRY IS 32 BYTES)
                BCC     ENTVCB              ; BRANCH IF MORE TO FIND
                LDA     #0
ENTVCB2         EQU     *                   ; SEE IF WE CAN REPLACE A DEVICE
                TAX
                LDA     VCB+VCBSTAT,X       ; VCB HAS FILES OPEN?
                BEQ     FREEVCB             ; NO, USE IT!
                TXA
                CLC
                ADC     #VCBSIZE            ; SEARCH NEXT VCB ENTRY
                BCC     ENTVCB2
                RTS                         ; FAILED TO FIND A FREE VCB ENTRY
*
CHKVLOG         LDY     #0                  ; MAKE SURE VOLUME WAS ACTUALLY LOGGED IN
                LDA     (VCBPTR),Y
                BNE     FOUNDVOL            ; AH, MADE IT...
                LDA     #DUPVOL             ; WELL, NOT QUITE, THIS VOLUME CAN'T BE LOGGED
                SEC
                RTS
                PAGE
*
FREEVCB         STX     VCBPTR              ; NOW THIS IS THE POINTER TO A FREE VCB
                LDA     #2                  ; ROOT DIRECTORIES ALWAYS AT BLOCK 2
                LDX     #0
                BEQ     GETROOT             ; BRANCH ALWAYS
LOKVOL1         LDY     #VCBSTAT            ; MAKE SURE NO FILES ARE ACTIVE ON
                LDA     (VCBPTR),Y          ;  THE VOLUME BEFORE LOGGING IT IN.
                BMI     SNSWIT              ;  BRANCH IF FILES ACTIVE
                LDY     #VCBROOT+1          ; GET ADDRESS OF ROOT DIRECTORY
                LDA     (VCBPTR),Y          ; HIGH FIRST.
                TAX
                DEY                         ; THEN LOW.
                LDA     (VCBPTR),Y
GETROOT         JSR     GETROT0
                BCC     LOKVOL2             ; BRANCH IF SUCCESSFULLY READ.
                LDA     #0                  ; OTHERWISE, TAKE THIS DEVICE OUT OF VCB
                TAY
                STA     (VCBPTR),Y          ; (VOLUME 'OFF LINE')
                BEQ     NXTDEV              ; BRANCH ALWAYS
*
LOKVOL2         JSR     LOGVCB              ; GO UPDATE VCB TO INCLUDE CURRENT VOLUME INFO
                BCS     NXTDEV              ;  IF NOT A SOS DISKETTE, SKIP TO NEXT DEVICE
```

```
                JSR     CHKROOT             ; GO COMPARE TO SEE IF WE FOUND WHAT WE'RE
                BEQ     CHKVLOG             ;  LOOKING FOR...
*
NXTDEV          LDX     SCRTCH              ; LOOK AT OTHER DEVICES?
                CPX     TOTDEVS
                BCC     LOKDEV1             ; YES.
                LDA     #VNFERR             ; REPORT VOLUME NOT FOUND.
                RTS
*
SNSWIT          EQU     *                   ;  SENSE DSWITCH
                LDY     #VCBDEV
                LDA     (VCBPTR),Y
                STA     DEVNUM              ;  MAKE SURE DEVICE NUMBER IS CURRENT
                JSR     TWRPROT1            ;  USES DEVNUM
                LDA     DSWGLOB             ;  DISK SWITCH GLOBAL
                BEQ     NXTDEV              ;  BRANCH IF NO DISK SWITCH
                JSR     VERFYVOL            ;  COMPARES VCBPTR VS. DEVNUM CONTENTS
                BCC     NXTDEV              ;  BRANCH IF DISK HAS NOT BEEN SWITCHED
                JSR     CHKROOT             ;  COMPARES PATHNML VS. GBUF
                BNE     NXTDEV              ;  IGNORE IF NOT WHAT WE ARE LOOKING FOR
                LDX     #0                  ;  LOOK FOR FREE
                JSR     SNSWIT1
                BCS     NXTDEV              ;  ANY ERRORS LOGGING IN THE NEW VOLUME
                JMP     CHKVLOG             ;  MAKE SURE THE NEW VOLUME IS LOGGED
SNSWIT1         LDA     VCB,X               ;  VCB ENTRY
                BEQ     SNSWIT2             ;  BRANCH IF FOUND
                TXA
                CLC
                ADC     #VCBSIZE            ;  LOOK AT NEXT VCB AREA
                TAX
                BCC     SNSWIT1
                RTS                         ;  CAN'T BE LOGGED IN!
SNSWIT2         LDA     #0
                STA     DUPLFLAG            ;  TURN OFF DUPLICATE VOLUME FLAG
                STX     VCBPTR
                JSR     LOGVCB1             ;  PARTIALLY LOG IN THE NEW VOLUME
                BCS     NONSOS              ;  CS MEANS NONSOS ERROR
                LDA     DUPLFLAG            ;  WAS IT A DUPLICATE VOLUME?
                BNE     SNSWIT6             ;  BRANCH IF YES
                LDY     #VCBSWAP            ;  BY MAKING SWAP BYTE NON ZERO
                LDA     #1
                STA     (VCBPTR),Y          ;  SO SWAPOUT WON'T AFFECT
                LDA     DEVNUM              ;  A REG PASSES DEVNUM TO SWAPOUT
                JSR     SWAPOUT             ;  OLD ACTIVE MOUNT MUST BE SWAPPED
                BCC     SNSWIT3
                LDA     #XIOERROR           ;  USER REFUSED TO REPLACE OLD VOLUME
                RTS
SNSWIT3         LDY     #VCBSWAP            ;  NOW LOG IN THE NEW ALL THE WAY
                LDA     #0
                STA     (VCBPTR),Y
SNSWIT4         JSR     VERFYVOL            ;  DON'T BOTHER TO ASK IF NEW VOLUME IS ALREADY MOUNTED
                BCC     SNSWIT5             ;  BRANCH IF NEW VOLUME ON LINE
                JSR     USRREQ              ;  ASK USER TO REMOUNT NEW VOLUME
                BCC     SNSWIT4             ;  USER SAYS THEY DID: CHECK IT OUT
                LDA     #VNFERR
SNSWIT5         RTS
SNSWIT6         LDA     #DUPVOL
                SEC
                RTS
                PAGE
*
NONSOS          LDA     #NOTSOS             ; TELL EM IT'S NOT A SOS DISK (COULD BE PASCAL)
                RTS                         ;  CARRY SHOULD ALREADY BE SET
*
*
DEVVCB          LDA     #0                  ; SCAN VCB FOR DEVICE SPECIFIED IN 'DEVNUM'
DVCB1           TAX                         ; FIRST TEST FOR VALID VCB.
                LDA     VCB,X
                BEQ     DVCB2
                LDA     VCB+VCBSWAP,X       ;  SWAPPED VOLUMES DON'T COUNT
                BNE     DVCB2               ;  AS LOGGED IN
                LDA     VCB+VCBDEV,X        ; GET DEVICE NUMBER
                CMP     DEVNUM              ; TEST AGAINST REQUESTED DEVICE
                BEQ     FOUNDEV             ; YES, SET UP A POINTER TO IT
DVCB2           TXA                         ; BUMP TO NEXT VCB
                CLC
                ADC     #VCBSIZE
                BCC     DVCB1               ; BRANCH IF MORE TO LOOK AT.
                RTS                         ; RETURN CARRY SET TO INDICATE NOT FOUND
*
TSTDUPVOL       LDX     VCBPTR              ; PRESERVE CURRENT ADDR OF FREE VCB
```

```
                JMP        DISPATCH
                SBTL       "NON-MASKABLE INTERRUPT RECEIVER"
                REP        60
*
*  THIS IS THE NON-MASKABLE INTERRUPT RECEIVER.  WHEN AN
*  NMI OCCURS, THE CPU PASSES CONTROL TO THE NMI RECEIVER
*  THROUGH THE NMI VECTOR.  THE OPERATION OF THE NMI
*  RECEIVER IS ESSENTIALLY THE SAME AS THE GIR EXCEPT
*  THAT IT IS NOT CONCERNED WITH BRK, AND THE ONLY VALID
*  SOURCE OF AN NMI IS THE KEYBOARD OR THE I/O DEVICE THAT
*  HAS ALLOCATED THE NMI RESOURCE.
*
                REP        60
*
*
NMI.RCVR        EQU        *
*
*  SAVE CPU REGISTERS A, X, & Y ON CURRENT STACK
*
                PHA
                TXA
                PHA
                TYA
                PHA
*
*  CHECK FOR STACK OVERFLOW
*
                TSX
                CPX        #$FA
                BCC        NMI005
                LDA        #>STKOVFL
                JSR        SYSDEATH
*
*  SET UP INTERRUPT ENVIRONMENT:
*    BINARY ARITHMETIC, 2 MHZ, I/O ENABLED,
*    RAM WRITE ENABLED, PRIMARY STACK,
*    AND $F000 RAM SELECTED.  PRESERVE
*    USER STATE OF SCREEN AND RESET LOCK.
*
NMI005          CLD
                LDA        E.REG
                TAX
                AND        #BITON5+BITON4
                ORA        #BITON6+BITON2
                STA        E.REG
*
*  IF NOT ALREADY ON PRIMARY STACK, SAVE USER'S
*  STACK POINTER AND SET UP SOS STACK POINTER.
*
                TXA
                AND        #BITON2
                BNE        NMI010
                TXA
                TSX
                STX        SP.SAVE
                LDX        #>E.SAVE
                TXS
                TAX
*
*  SAVE SYSTEM CONTROL REGISTERS E, Z, & B ON SOS STACK
*
NMI010          TXA
                PHA
                LDA        Z.REG
                PHA
                LDA        B.REG
                PHA
                LDA        EXPNSLOT
                PHA
                BIT        $CFFF
                BIT        $C020                ;RESET I/O SPACE
                LDA        #$00
                STA        EXPNSLOT
*
*  SET INTERRUPT ZERO PAGE
*
                LDA        #0
                STA        Z.REG
*
*  SEE IF NMI IS FROM KEYBOARD OR I/O DEVICE
*
```

```
                LDA       E.IORB
                BMI       NMI030
*
*  NMI IS FROM I/O DEVICE
*
                LDA       SIRTABLE            ;NMI ALLOCATED?
                BPL       NMI020
                JSR       CALLNMI
                SEI
                JMP       DISPATCH
CALLNMI         LDA       SIRADR.L
                STA       NMIADR.L
                LDA       SIRADR.B
                STA       B.REG
                JMP       (NMIADR.L)
*
*  BAD INTERRUPT -- SYSTEM DEATH
*
NMI020          LDA       #>BADINT1           ;NMI NOT ALLOCATED
                JSR       SYSDEATH
*
*  NMI IS FROM THE KEYBOARD
*
NMI030          LDA       SYSBANK
                STA       B.REG
                JSR       KYBDNMI
                SEI
                JMP       DISPATCH
                SBTL      "DISPATCHER"
                REP       60
*
*  THIS IS THE DISPATCHER.  UPON COMPLETION, ALL SOS CALLS
*  AND INTERRUPT HANDLERS RETURN CONTROL TO THE DISPATCHER.
*  ITS PURPOSE IS TO SET UP THE APPROPRIATE ENVIRONMENT AND
*  PASS CONTROL TO WHATEVER CODE SHOULD RUN NEXT.
*
*  WHEN SOS IS INTERRUPTED, CONTROL ALWAYS RETURNS TO THE
*  INTERRUPTED CODE.  HOWEVER, WHEN THE USER IS INTERRUPTED,
*  BY EITHER A SOS CALL OR AN INTERRUPT, THE DISPATCHER
*  MUST CHECK THE EVENT QUEUE.  IF THERE IS AN ACTIVE EVENT
*  WITH A PRIORITY HIGHER THAN THE CURRENT EVENT FENCE,
*  CONTROL IS PASSED TO THE EVENT CODE.  OTHERWISE, CONTROL
*  RETURNS TO THE INTERRUPTED CODE.
*
                REP       60
*
DISPATCH        EQU       *
*
*  DISABLE INTERRUPTS AND RESTORE
*  SYSTEM CONTROL REGISTERS B & Z
*
                SEI
                LDA       E.REG
                ORA       #BITON6             ;ENABLE I/O
                STA       E.REG
                PLA
                JSR       SELC800             ;RESTORE I/O SPACE
                PLA
                STA       B.REG
                PLA
                STA       Z.REG
*
*  CHECK SAVED ENVIRONMENT REGISTER
*  IF RETURNING TO PRIMARY STACK
*    THEN RESTORE E REG AND RELAUNCH SOS
*    ELSE RESET STACK POINTER & RESTORE E REG
*
                PLA
                ORA       #BITON5             ;SET SCREEN STATE TO
                BIT       SCRNMODE            ;  CURRENT SCREEN MODE
                BMI       DSP005
                AND       #BITOFF5
DSP005          TAY
                AND       #BITON2
                BEQ       DSP010
                STY       E.REG
                BNE       DSP030
DSP010          PLA
                TAX
                TXS
                STY       E.REG
```

```
*
*  CHECK FOR ACTIVE EVENT WITH PRIORITY > FENCE
*
DSP020          LDA        CEVPRI
                LDX        EVQ.LINK
                CMP        EVQ.PRI,X
                BCS        DSP030
*
*  PROCESS ACTIVE EVENT TRAP
*  SAVE E, Z, B, & CALLER'S PRIORITY ON STACK THEN CALL
*  EVENT.  UPON RETURN, RESTORE PRIORITY, B, Z, & E THEN
*  CHECK FOR MORE EVENTS.
*
                LDA        E.REG
                PHA
                LDA        Z.REG
                PHA
                LDA        B.REG
                PHA
                LDA        CEVPRI
                PHA
                JSR        DO.EVENT
                SEI
                PLA
                STA        CEVPRI
                PLA
                STA        B.REG
                PLA
                STA        Z.REG
                PLA
                ORA        #BITON5              ;SET SCREEN STATE TO
                BIT        SCRNMODE             ;  CURRENT SCREEN MODE
                BMI        DSP025
                AND        #BITOFF5
DSP025          STA        E.REG
                JMP        DSP020
*
*  RESTORE CPU REGISTERS Y, X, & A AND LAUNCH
*
DSP030          PLA
                TAY
                PLA
                TAX
                PLA
                RTI
                PAGE
                REP        60
*
*  THIS SUBROUTINE CALLS THE HIGHEST PRIORITY ACTIVE EVENT.
*  FIRST, IT DELINKS THE FIRST ENTRY ON THE ACTIVE LIST AND
*  LINKS IT TO THE FREE LIST.  THEN, IT SETS UP THE BANK,
*  ADDRESS, ID, & STATUS AND CALLS THE EVENT VIA AN RTI.
*
                REP        60
*
DO.EVENT        EQU        *
*
*  WRITE ENABLE RAM
*
                LDY        E.REG
                TYA
                AND        #BITOFF3
                STA        E.REG
*
*  DELINK ENTRY FROM ACTIVE LIST AND RELINK IT TO FREE LIST
*
                LDX        EVQ.LINK
                LDA        EVQ.LINK,X
                STA        EVQ.LINK
                LDA        EVQ.FREE
                STA        EVQ.LINK,X
                STX        EVQ.FREE
*
*  SET FENCE TO EVENT PRIORITY THEN RESTORE E REG
*
                LDA        EVQ.PRI,X
                STA        CEVPRI
                STY        E.REG
*
*  SET UP B, Z, E, ADDRESS, ID, & STATUS
*
```

```
                LDA        EVQ.BANK,X
                STA        B.REG
                LDA        EVQ.ADRH,X
                PHA
                LDA        EVQ.ADRL,X
                PHA
                LDY        EVQ.ID,X
                PHP
                PLA
                AND        #$82
                PHA
                TYA
                RTI

                CHN        IPL.SRC2

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


```

End of File -- Lines: 664 Characters: 17425

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

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

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

```
*      A, Y:  UNDEFINED
*
*
*  DEALCSIR -- DEALLOCATE SYSTEM INTERNAL RESOURCES
*
*    PARAMETERS:
*      A:  NUMBER OF BYTES IN TABLE
*      X:  TABLE ADDRESS (LOW BYTE)
*      Y:  TABLE ADDRESS (HIGH BYTE)
*
*    NORMAL EXIT -- SIRS DEALLOCATED
*      CARRY:  CLEAR
*      A, X, Y:  UNDEFINED
*
*    ERROR EXIT -- SIRS NOT DEALLOCATED
*      CARRY:  SET
*      X:  SIR NUMBER
*      A, Y:  UNDEFINED
*
                REP        60
                PAGE
*
IDBYTE          DFB        $81
*
ALLOCSIR        EQU        *
                CLC
                PHP
                SEI
                STA        SIRARGSIZ           ;SAVE TABLE SIZE
                LDA        E.REG
                STA        SIRTEMP
                ORA        #BITON2             ;FORCE PRIMARY STACK
                AND        #BITOFF3            ;  AND WRITE ENABLE
                STA        E.REG
                LDA        SIRTEMP
                PHA
                LDA        Z.REG
                PHA
                LDA        #$00
                STA        Z.REG               ;SET ZERO PAGE := $00
                STX        SIRARGS
                STY        SIRARGS+1           ;SET POINTER TO TABLE
*
                LDY        #$00
ASIR010         LDA        (SIRARGS),Y         ;GET SIR NUMBER
                CMP        #SIRTBLSIZ
                TAX
                BCS        ASIR020
                LDA        SIRTABLE,X          ;CHECK ALLOCATION
                BMI        ASIR020
                LDA        IDBYTE
                STA        SIRTABLE,X          ;ALLOCATE SIR
                INY
                STA        (SIRARGS),Y         ;RETURN ID BYTE
                INY
                LDA        (SIRARGS),Y
                STA        SIRADR.L,X          ;SAVE INTERRUPT ADDRESS
                INY
                LDA        (SIRARGS),Y
                STA        SIRADR.H,X
                INY
                LDA        (SIRARGS),Y
                STA        SIRADR.B,X
                INY
                CPY        SIRARGSIZ
                BCC        ASIR010
*
                CLC
                INC        IDBYTE              ;BUMP ID BYTE
                BMI        SIREXIT
                LDA        #$81
                STA        IDBYTE
                BMI        SIREXIT
*
ASIR020         STX        SIRTEMP             ;SAVE BAD SIR NUMBER
ASIR030         SEC
                TYA
                SBC        #5
                TAY
                BCC        ASIR040
                LDA        (SIRARGS),Y         ;GET SIR NUMBER
```

```
                TAX
                LDA     #FALSE
                STA     SIRTABLE,X          ;RELEASE ALLOCATED SIRS
                BEQ     ASIR030
*
ASIR040         LDX     SIRTEMP             ;RETURN BAD SIR
                SEC
*
*
*
SIREXIT         PLA
                STA     Z.REG               ;RESTORE Z REGISTER
                PLA
                STA     E.REG               ;RESTORE E REGISTER
                BCC     SIREXIT1
                PLA
                ORA     #BITON0
                PHA
SIREXIT1        PLP
                RTS
*
*
*
DEALCSIR        EQU     *
                CLC
                PHP
                SEI
                STA     SIRARGSIZ           ;SAVE TABLE SIZE
                LDA     E.REG
                STA     SIRTEMP
                ORA     #BITON2             ;FORCE PRIMARY STACK
                AND     #BITOFF3            ;  AND WRITE ENABLE
                STA     E.REG
                LDA     SIRTEMP
                PHA
                LDA     Z.REG
                PHA
                LDA     #$00
                STA     Z.REG               ;SET ZERO PAGE := $00
                STX     SIRARGS
                STY     SIRARGS+1           ;SET POINTER TO TABLE
*
                LDY     #$00
DSIR010         LDA     (SIRARGS),Y         ;GET SIR NUMBER
                TAX
                CPX     #SIRTBLSIZ
                BCS     DSIR030
                INY
                LDA     SIRTABLE,X
                BPL     DSIR030             ;VERIFY ALLOCATION
                CMP     (SIRARGS),Y
                BNE     DSIR030
                INY
                INY
                INY
                INY
                CPY     SIRARGSIZ
                BCC     DSIR010
*
                LDY     SIRARGSIZ
DSIR020         SEC
                TYA
                SBC     #5
                TAY
                BCC     SIREXIT
                LDA     (SIRARGS),Y         ;GET SIR NUMBER
                TAX
                LDA     #FALSE
                STA     SIRTABLE,X
                BEQ     DSIR020
*
DSIR030         SEC
                BCS     SIREXIT
                SBTL    "SELECT I/O EXPANSION ROM"
                REP     60
*
*  SUBROUTINE 'SELC800' IS CALLED TO SELECT THE C800 I/O EX-
*  PANSION ADDRESS SPACE FOR A PERIPHERAL SLOT.  ON ENTRY,
*  THE SLOT NUMBER IS PASSED IN THE ACCUMULATOR.  IF NO
*  ERROR OCCURS, CARRY IS CLEARED; OTHERWISE, CARRY IS SET
*  AND THE PREVIOUS SLOT REMAINS SELECTED.
```

```
*
*  PARAMETERS:
*     A:  SLOT NUMBER
*
*  NORMAL EXIT -- NEW SLOT SELECTED
*     CARRY:  CLEAR
*     A:  UNDEFINED
*     X, Y:  UNCHANGED
*
*  ERROR EXIT -- SLOT NOT CHANGED
*     CARRY:  SET
*     A, X, Y:  UNCHANGED
*
*  WARNING !!!
*     'SELC800' USES SELF-MODIFYING CODE!
*
                REP        60
*
SELC800         EQU        *
                CMP        #$05                 ;CHECK SLOT NUMBER
                BCS        SC8EXIT              ;  INVALID
                PHP
                SEI
                STA        EXPNSLOT
                ORA        #$C0                 ;MAKE SLOT INTO $CN00
                STA        CNADDR+2             ;  AND MODIFY BIT ADDRESS
                BIT        $C020
                BIT        $CFFF                ;DESELECT PREVIOUS SLOT
CNADDR          BIT        $C0FF                ;  AND SELECT CURRENT SLOT
                PLP
SC8EXIT         RTS
                SBTL       "NMI DISABLE / ENABLE"
                REP        60
*
*  THE SUBROUTINES NMIDSBL AND NMIENBL ARE CALLED TO
*  DISABLE AND ENABLE NMI, RESPECTIVELY.  THERE ARE NO
*  INPUT PARAMETERS.  ON EXIT, THE REGISTERS ARE UN-
*  DEFINED.  NMIDSBL CLEARS THE CARRY FLAG IF NMI WAS
*  SUCCESSFULLY DISABLED; OTHERWISE, CARRY IS SET.
*
                REP        60
*
NMIDSBL         EQU        *
                LDX        E.REG
                BIT        NMIFLAG
                BPL        NDS020
                TXA
                ORA        #BITON7
                STA        E.REG                ;SET 1MHZ
                LDA        #$00
                STA        NMICNTR
                STA        NMICNTR+1
NDS010          BIT        NMIFLAG              ;NMI PENDING?
                BPL        NDS020               ;  NO
                INC        NMICNTR              ;BUMP NMI COUNTER
                BNE        NDS010               ;  AND RECHECK NMI FLAG
                INC        NMICNTR+1
                BNE        NDS010
                LDA        #>NMIHANG            ;CAN'T LOCK NMI
                JSR        SYSDEATH
NDS020          TXA                             ;GET E.REG
                AND        #BITOFF4             ;DISABLE NMI
                STA        E.REG
                RTS
*
*
*
NMIENBL         EQU        *
                LDA        E.REG
                ORA        #BITON4              ;ENABLE NMI
                STA        E.REG
                RTS
                SBTL       "KEYBOARD NMI HANDLER"
                REP        60
*
*  BY DEFAULT, KEYBOARD NMI IS IGNORED.  THE USER MAY
*  PROCESS NMI BY CHANGING THE ADDRESS IN SYSTEM GLOBAL.
*
                REP        60
*
NMIDBUG         EQU        *
```

```
                TSX                             ;SAVE THE STACK POINTER
                STX        NMISPSV
                LDA        #$03                 ;SELECT MONITOR'S ZERO PAGE
                STA        Z.REG
                LDA        E.REG
                ORA        #$03                 ;SELECT MONITOR ROM
                STA        E.REG
                JSR        $F901                ;CALL THE MONITOR
*
NMICONT         EQU        *
                LDA        E.REG
                ORA        #BITON2              ;FORCE PRIMARY STACK
                STA        E.REG
                LDX        NMISPSV
                TXS                             ;RESTORE STACK POINTER
                RTS
                SBTL       "EVENT QUEUE MANAGER"
                REP        60
*
*  THE EVENT QUEUE IS USED TO HOLD THE PARAMETERS OF EVENTS
*  THAT HAVE BEEN DETECTED BUT NOT YET RECOGNIZED.  EVENT
*  QUEUE ENTRIES ARE ORGANIZED INTO TWO LINKED LISTS; A FREE
*  LIST AND AN ACTIVE LIST.  EACH ENTRY IS SIX BYTES LONG,
*  WITH THE FIRST BYTE (BYTE 0) USED AS A LINK.  THE LINK
*  BYTE CONTAINS THE TABLE INDEX OF THE NEXT ENTRY IN THE
*  LIST.  BECAUSE OF THE INDEXING METHOD, THE EVENT QUEUE
*  MUST NOT EXCEED 256 BYTES.
*
*  ENTRY ZERO IS A SPECIAL ENTRY.  BYTE 0 IS THE INDEX OF
*  THE FIRST ACTIVE ENTRY; BYTE 1 CONTAINS A ZERO, ALLOWING
*  ENTRY 0 TO BE USED AS THE ACTIVE EVENT LIST TERMINATER;
*  BYTE 2 CONTAINS THE INDEX OF THE FIRST FREE ENTRY; AND
*  BYTES 4 THROUGH 6 ARE UNUSED.
*
*  THE FREE LIST IS LINKED LIFO.  THE ONLY VALID BYTE IN A
*  FREE ENTRY IS THE LINK BYTE; THE REMAINING BYTES ARE
*  UNDEFINED.  THE FREE LIST IS TERMINATED BY A LINK BYTE
*  CONTAINING A ZERO.
*
*  THE ACTIVE LIST IS LINKED IN DECREASING PRIORITY ORDER
*  WITH ENTRIES OF EQUAL PRIORITY LINKED FIFO.  BYTES 1
*  THROUGH 5 CONTAIN THE EVENT PRIORITY, EVENT ID, LOW BYTE
*  OF THE EVENT ADDRESS, HIGH BYTE OF THE EVENT ADDRESS, AND
*  THE ADDRESS BANK.  THE ACTIVE LIST IS TERMINATED BY AN
*  ENTRY WITH AN EVENT PRIORITY OF ZERO.
*
                REP        60
                PAGE
                REP        60
*
*  SUBROUTINE 'QUEEVENT' IS USED TO ENTER AN EVENT INTO THE
*  EVENT QUEUE.  ACTIVE EVENTS ARE LINKED IN DECREASING
*  PRIORITY ORDER WITH EVENTS OF EQUAL PRIORITY LINKED FIFO.
*  EVENTS ARE REMOVED FROM THE QUEUE AS THEY ARE RECOGNIZED
*  BY THE DISPATCHER.
*
*  PARAMETERS:
*    X:  EVENT PARAMETER ADDRESS (LOW BYTE)
*    Y:  EVENT PARAMETER ADDRESS (HIGH BYTE)
*
*    EVENT       0       1       2       3       4
*    PARMS:  +-------+-------+-------+-------+-------+
*            |  PRI  |   ID  | ADR.L | ADR.H | ADR.B |
*            +-------+-------+-------+-------+-------+
*            PRI:  EVENT PRIORITY
*            ID:   EVENT ID BYTE
*            ADR:  EVENT ADDRESS (LOW, HIGH, BANK)
*
*  EXIT CONDITIONS:
*    CARRY:  CLEAR
*    A, X, Y:  UNDEFINED
*
                REP        60
*
QUEEVENT        EQU        *
                CLC
                PHP
                SEI
                LDA        E.REG
                STA        QEVTEMP
                ORA        #BITON2              ;FORCE PRIMARY STACK
```

```
                AND     #BITOFF3            ;  AND WRITE ENABLE
                STA     E.REG
                LDA     QEVTEMP
                PHA
                LDA     Z.REG
                PHA
                LDA     #0
                STA     Z.REG               ;SET ZERO PAGE := 0
*
                STX     QEVARGS
                STY     QEVARGS+1           ;SET ARGUMENT POINTER
                LDY     #0
                LDA     (QEVARGS),Y         ;GET PRIORITY
                BEQ     Q.EXIT              ;  IGNORE IF ZERO
*
                LDX     EVQ.FREE
                BEQ     Q.FULL
                STX     QEV.THIS            ;GET FIRST FREE ENTRY
                LDA     EVQ.LINK,X          ;  AND DELINK IT
                STA     EVQ.FREE
*
                LDY     #EVQ.SIZ-2
QEV010          LDA     (QEVARGS),Y         ;COPY ARGUMENTS
                STA     EVQ.BANK,X          ;  INTO NEW ENTRY
                DEX
                DEY
                BPL     QEV010
*
                LDX     QEV.THIS
                LDY     #0
QEV020          STY     QEV.LAST
                LDA     EVQ.LINK,Y
                TAY
                LDA     EVQ.PRI,Y           ;SCAN EVENT QUEUE
                CMP     EVQ.PRI,X           ;  FOR PROPER POSITION
                BCS     QEV020
*
                TYA
                STA     EVQ.LINK,X          ;RELINK EVENT INTO QUEUE
                TXA
                LDY     QEV.LAST
                STA     EVQ.LINK,Y
*
Q.EXIT          PLA
                STA     Z.REG               ;RESTORE Z REGISTER
                PLA
                STA     E.REG               ;RESTORE E REGISTER
                PLP
                RTS
*
Q.FULL          LDA     #>EVQOVFL           ;EVENT QUEUE OVERFLOW
                JSR     SYSDEATH
                LST     ON

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

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

End of File -- Lines: 465 Characters: 12364

```
                 STA        CHG.NEW+1
                 BCC        CHGS050
                 BCS        CHGS052               ; OVERFLOW, PEG IT
                 REP        35
* CHANGE MODE = 3(LIMIT DOWN)
                 REP        35
* CHG.NEW:=ST.LIM(SEG#)-PGCT
*
CHGS040          SEC
                 LDA        ST.LIML,X
                 SBC        CHG.PGCTX
                 STA        CHG.NEW
                 LDA        ST.LIMH,X
                 SBC        CHG.PGCTX+1
                 STA        CHG.NEW+1
                 BCC        CHGS044               ; OVERFLOW, PEG IT
*
* IF CHG.NEW >= ST.BASE(SEG#) THEN EXIT
*
                 LDA        CHG.NEW
                 CMP        ST.BASEL,X
                 LDA        CHG.NEW+1
                 SBC        ST.BASEH,X
                 BCS        CHGS046
*
* OTHERWISE CHG.NEW:=ST.BASE(SEG#)
*
CHGS044          LDA        ST.BASEL,X
                 STA        CHG.NEW
                 LDA        ST.BASEH,X
                 STA        CHG.NEW+1
*
CHGS046          JMP        CHGS.EXIT
*
* DETERMINE NEW BANK/PAGE'S REGION,
* IF NEW BANK/PAGE IS INVALID THEN
* SET TO BASE OR LIMIT (CASE CHANGE MODE)
*
CHGS050          LDX        CHG.NEW
                 LDY        CHG.NEW+1
                 JSR        REGION
                 BCS        CHGS052
                 BNE        CHGS052
                 BEQ        CHGS100
CHGS052          LDA        CHG.MODE
                 CMP        #1
                 BNE        CHGS054
                 LDX        #>VRT.BASE
                 LDY        #<VRT.BASE
                 JMP        CHGS056
CHGS054          LDX        >VRT.LIM
                 LDY        >VRT.LIM+1
CHGS056          STX        CHG.NEW
                 STY        CHG.NEW+1
                 PAGE
*
* COMPUTE BANK/PAGE OF ADJACENT SEGMENT, IF ANY
*   CASE CHANGE MODE
*
CHGS100          LDX        CHG.NUM
                 LDA        CHG.MODE
                 CMP        #1
                 BNE        CHGS200
*   "1" IF ST.FLINK(SEG#)=0 THEN EXIT
                 LDA        ST.FLINK,X
                 BEQ        CHGS.EXIT
*       X,Y:=ST.LIM(ST.FLINK(SEG#))+1
                 TAY
                 LDA        ST.LIML,Y
                 TAX
                 LDA        ST.LIMH,Y
                 TAY
                 INX
                 BNE        CHGS110
                 INY
*       IF CHG.NEW < X,Y THEN CHG.NEW:=X,Y
CHGS110          CPY        CHG.NEW+1
                 BCC        CHGS.EXIT
                 BEQ        CHGS120
                 BCS        CHGS300
CHGS120          CPX        CHG.NEW
```

```
                BCC        CHGS.EXIT
                BCS        CHGS300
*   "2" IF ST.BLINK(SEG#)=0 THEN EXIT
CHGS200         LDA        ST.BLINK,X
                BEQ        CHGS.EXIT
*       X,Y:= ST.BASE(ST.BLINK(SEG#))-1
                TAY
                LDA        ST.BASEL,Y
                TAX
                LDA        ST.BASEH,Y
                TAY
                TXA
                BNE        CHGS210
                DEY
CHGS210         DEX
*       IF CHG.NEW > X,Y THEN CHG.NEW:=X,Y
                CPY        CHG.NEW+1
                BCC        CHGS300
                BEQ        CHGS220
                BCS        CHGS.EXIT
CHGS220         CPX        CHG.NEW
                BCS        CHGS.EXIT
*
CHGS300         STX        CHG.NEW
                STY        CHG.NEW+1
                PAGE
                REP        35
*
* COMPUTE DELTA PAGE COUNT AND RETURN IT TO CALLER
* (CASE OF CHG.MODE)
*
                REP        35
CHGS.EXIT       LDX        CHG.NUM
                LDY        #0
                LDA        CHG.MODE
                CMP        #1
                BCC        CHGS500
                BEQ        CHGS510
                CMP        #3
                BCC        CHGS520
                BEQ        CHGS530
*
* "0" -- PAGECOUNT:=NEW-BASE
*
CHGS500         SEC
                LDA        CHG.NEW
                SBC        ST.BASEL,X
                STA        (CHG.PGCT),Y
                LDA        CHG.NEW+1
                SBC        ST.BASEH,X
                JMP        CHGS600
*
* "1" -- PAGECOUNT:=BASE-NEW
*
CHGS510         SEC
                LDA        ST.BASEL,X
                SBC        CHG.NEW
                STA        (CHG.PGCT),Y
                LDA        ST.BASEH,X
                SBC        CHG.NEW+1
                JMP        CHGS600
*
* "2" -- PAGECOUNT:=NEW-LIM
*
CHGS520         SEC
                LDA        CHG.NEW
                SBC        ST.LIML,X
                STA        (CHG.PGCT),Y
                LDA        CHG.NEW+1
                SBC        ST.LIMH,X
                JMP        CHGS600
*
* "3" -- PAGECOUNT:=LIM-NEW
*
CHGS530         SEC
                LDA        ST.LIML,X
                SBC        CHG.NEW
                STA        (CHG.PGCT),Y
                LDA        ST.LIMH,X
                SBC        CHG.NEW+1
*
```

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

                CHN       MEMMGR.C.SRC

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

End of File -- Lines: 606 Characters: 12658

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

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

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

```
                STA     (GSI.PGCT),Y
*
                CLC
                RTS                             ; NORMAL EXIT
*
GSI.ERR         LDA     #BADSEGNUM
                JSR     SYSERR                  ; ERR EXIT
                PAGE
                REP     60
*
* GET.SEG.NUM(IN.BANKPAGE; OUT.SEGNUM) SYSTEM CALL
*
*
                REP     60
*
GET.SEG.NUM     EQU     *
*
* CONVERT BANKPAGE TO INTERNAL FORMAT
*
                LDX     GSN.BKPG
                LDY     GSN.BKPG+1
                JSR     CNVRT.IBP
                BCS     GSN.ERR                 ; ERR - INVALID BANK PAGE
                STX     GSN.BKPG
                STY     GSN.BKPG+1
*
* QUIT IF NO ENTRIES IN SEG TABLE
*
                LDA     ST.ENTRY
                BEQ     GSN.ERR1                ; ERR - SEG NOT FOUND
*
* L1: IF BANKPAGE>ST.LIM(SEG#) THEN ERR
*
GSN010          TAX
                LDA     ST.LIML,X
                CMP     GSN.BKPG
                LDA     ST.LIMH,X
                SBC     GSN.BKPG+1
                BCC     GSN.ERR1                ; ERR - SEG NOT FOUND
*
* IF BANKPAGE>=ST.BASE(SEG#) THEN FOUND!
*
                LDA     GSN.BKPG
                CMP     ST.BASEL,X
                LDA     GSN.BKPG+1
                SBC     ST.BASEH,X
                BCS     GSN020
*
* SEG#:=ST.FLINK(SEG#); GOTO L1
*
                LDA     ST.FLINK,X
                BEQ     GSN.ERR1                ; ERR - SEG NOT FOUND
                JMP     GSN010
*
* RETURN SEG# TO CALLER
*
GSN020          LDY     #0
                TXA
                STA     (GSN.NUM),Y
                CLC
                RTS                             ; NORMAL EXIT
*
GSN.ERR         RTS                             ; ERROR EXIT
*
GSN.ERR1        LDA     #SEGNOTFND
                JSR     SYSERR                  ; ERROR EXIT
                PAGE
                REP     60
*
* RELEASE.SEG(IN.SEGNUM) SYSTEM CALL
*
                REP     60
*
RELEASE.SEG     EQU     *
*
* IF ST.FLINK(SEG#)=ST.FREE THEN ERR
*
                LDX     RLS.NUM
                BEQ     RLS.ALL                 ; RELEASE.SEG(SEG#=0)
                CPX     #ST.CNT
                BCS     RLS.ERR                 ; ERR - SEG# TOO LARGE
```

```
                LDA     ST.FLINK,X
                BMI     RLS.ERR             ; ERR - INVALID SEGNUM
                BPL     REL.SEG             ; RELEASE.SEG(SEG#>0)
                REP     35
*
* RELEASE ALL
*
                REP     35
RLS.ALL         LDX     ST.ENTRY
                BEQ     RLS0.EXIT
                STX     RLS.NUM
*
RLS0.LOOP       LDA     ST.ID,X
                CMP     #$10                ; CARRY SET/CLEARED HERE
*
                LDA     ST.FLINK,X
                PHA
                BCC     RLS006              ; IF ID=SYS SEG THEN SKIP
                JSR     REL.SEG             ; RELEASE ONE SEGMENT
RLS006          PLA
                BEQ     RLS0.EXIT
                STA     RLS.NUM
                TAX
                BNE     RLS0.LOOP           ; ALWAYS TAKEN
*
RLS0.EXIT       CLC
                RTS                         ; NORMAL EXIT ; ALL NON SYSTEM SEGMENTS RELEASED.
                REP     35
*
* REL SEG
*
                REP     35
* Y:=ST.FLINK(SEG#)
* X:=ST.BLINK(SEG#)
*
REL.SEG         TAY
                LDA     ST.BLINK,X
                TAX
*
* IF X<>0 THEN ST.FLINK(X):=Y
*         ELSE ST.ENTRY:=Y
*
                BEQ     RLS010
                TYA
                STA     ST.FLINK,X
                JMP     RLS020
RLS010          STY     ST.ENTRY
*
* IF Y<>0 THEN ST.BLINK(Y):=X
*
                TYA
RLS020          BEQ     RLS030
                TXA
                STA     ST.BLINK,Y
*
* ST.FLINK(SEG#):=ST.FREE
* ST.FREE:=SEG# AND #$80
*
RLS030          LDA     ST.FREE
                LDX     RLS.NUM
                STA     ST.FLINK,X
                TXA
                ORA     #$80
                STA     ST.FREE
*
                CLC
                RTS                         ; NORMAL EXIT
*
RLS.ERR         LDA     #BADSEGNUM
                JSR     SYSERR              ; ERR EXIT
                PAGE
                REP     60
*
* CONVERT INTERNAL BANK PAGE
*
* INPUT:  EXTERNAL BANK (X)
*            "     PAGE (Y)
* OUTPUT: INTERNAL BKPG LOW  (X)
*            "     BKPG HIGH (Y)
*         REGION (A) 0=>VIRT BANK
*                    1=>PHY BANK (0-$2000)
```

```
*                    2=>   "     ($A000-$FFFF)
* ERROR:  CARRY SET ("INVALID BANK PAGE")
*
                REP        60
*
CNVRT.IBP       EQU        *
*
* CONVERT FROM EXTERNAL TO INTERNAL FORMAT
*
* CASE OF BANK:  ADD PAGE BIAS
*
                TYA
                CPX        #$F
                BEQ        CNVI010
                BCS        CNVI020
*
                CMP        #$20                 ; BANK < "F"
                BCC        CNVI.ERR1
                CMP        #$A0
                BCS        CNVI.ERR1
                SEC
                SBC        #$20
                JMP        CNVI030
*
CNVI010         CMP        #$20                 ; BANK = "F"
                BCS        CNVI.ERR1
                CLC
                ADC        #$80
                JMP        CNVI030
*
CNVI020         CPX        #$10                 ; BANK = "10"
                BNE        CNVI.ERR1
                CMP        #$A0
                BCC        CNVI.ERR1
                SEC
                SBC        #$80
*
CNVI030         TAY                             ; SHIFT BANK RIGHT ONE BIT
                TXA                             ; INTO HIGH BIT OF PAGE BYTE.
                LSR        A
                TAX
                TYA
                BCC        CNVI040
                ORA        #$80
*
* EXCHANGE X & Y
*
CNVI040         PHA
                TXA
                TAY
                PLA
                TAX
*
* COMPUTE REGION (VIRT=0,PHY1=1,PHY2=2)
*
                JSR        REGION               ; REGION RETURNED IN A REG.
                BCS        CNVI.ERR1            ; ERR - INVALID BANK PAGE
*
                RTS                             ; NORMAL EXIT
*
CNVI.ERR1       LDA        #BADBKPG
                JSR        SYSERR
                PAGE
                REP        60
*
* CONVERT EXTERNAL BANK PAGE
*
* INPUT:  INTERNAL BKPG LOW  (X)
*              "         HIGH (Y)
* OUTPUT: EXTERNAL BANK (X)
*              "    PAGE (Y)
* ERROR:  NO ERROR CHECKING DONE.  ASSUMES THAT INTERNAL #S
* ARE VALID.
*
                REP        60
*
CNVRT.XBP       EQU        *
*
* CONVERT FROM INTERNAL TO EXTERNAL FORMAT
*
                TXA
```

```
                ASL        A
                TXA
                AND        #$7F
                TAX
                TYA
                ROL        A
                TAY
*
* CASE OF BANK: ADD PAGE BIAS
*
                TXA
                CPY        #$F
                BEQ        CNVX020              ; BANK = "F"
                BCS        CNVX010
*
                CLC                             ; BANK < "F"
                ADC        #$20
                JMP        CNVX020
*
CNVX010         CLC                             ; BANK = "10"
                ADC        #$80
*
* EXCHANGE X & Y
*
CNVX020         PHA
                TYA
                TAX
                PLA
                TAY
                RTS                             ; NORMAL EXIT
                PAGE
                REP        60
*
* REGION
*
* INPUT:  INTERNAL BKPG LOW  (X)
*             "         HIGH (Y)
* OUTPUT: REGION (A)
*         INTERNAL BKPG LOW  (X) UNCHANGED
*             "         HIGH (Y)    "
* ERROR:  CARRY SET ("INVALID BANK/PAGE")
*
                REP        60
*
REGION          EQU        *
                STX        RGN.BKPG
                STY        RGN.BKPG+1
*
* IF BANKPAGE>PHY2LIM THEN ERR
*
                LDA        #>PHY2LIM
                CMP        RGN.BKPG
                LDA        #<PHY2LIM
                SBC        RGN.BKPG+1
                BCC        RGN.ERR              ; ERR - INVALID BANK PAGE
*
* IF BANKPAGE>=PHY2BASE THEN REGION:=2
*
                LDA        RGN.BKPG
                CMP        #>PHY2BASE
                LDA        RGN.BKPG+1
                SBC        #<PHY2BASE
                BCC        RGN010
                LDA        #2
                BNE        RGN040
*
* IF BANKPAGE>PHY1LIMIT THEN ERR
*
RGN010          LDA        #>PHY1LIM
                CMP        RGN.BKPG
                LDA        #<PHY1LIM
                SBC        RGN.BKPG+1
                BCC        RGN.ERR              ; ERR - INVALID BANK PAGE
*
* IF BANKPAGE>=PHY1BASE THEN REGION:=1
*
                LDA        RGN.BKPG
                CMP        #>PHY1BASE
                LDA        RGN.BKPG+1
                SBC        #<PHY1BASE
                BCC        RGN020
```

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

```
GTFR030         TYA
                CLC
                RTS                               ; NORMAL EXIT
*
GTFR.ERR        LDA       #SEGTBLFULL
                JSR       SYSERR
*
                LST       ON
ZZEND           EQU       *
ZZLEN           EQU       ZZEND-ZZORG
                IFNE      ZZLEN-LENMEMMG
                FAIL      2,"SOSORG               FILE IS INCORRECT FOR MEMMGR"
                FIN

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


```

End of File -- Lines: 500 Characters: 10157

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

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

                SBTL       "SOS 1.1  OPERATOR MESSAGE/REPLY"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGOMSG
ZZORG           EQU        *
                MSB        OFF
                REP        60
*
*          COPYRIGHT (C) APPLE COMPUTER INC. 1981
*                   ALL RIGHTS RESERVED
*
                REP        60
*
*  THIS MODULE CONTAINS THE BLOCK FILE MANAGERS'S OPERATOR
*  INTERFACE.  IT DISPLAYS A MESSAGE IN A FOUR LINE BY
*  FOURTY COLUMN WINDOW, THEN WAITS FOR THE USER TO TOGGLE
*  THE ALPHA-LOCK KEY BEFORE RETURNING.
*
*  THE VERTICAL BLANKING FLAGS AND COMPOSITE BLANKING
*  TIMER ARE USED TO MAINTAIN THE DISPLAY.  MEMORY PAGE
*  $02 IS USED FOR TEMPORARY STORAGE.  ON EXIT, ALL
*  RESOURCES ARE RESTORED TO THEIR PREVIOUS STATES.
*
*  ENTRY POINT:  OPMSGRPLY
*
*  PARAMETERS:  X -- MESSAGE ADDRESS (LOW BYTE)
*               Y -- MESSAGE ADDRESS (HIGH BYTE)
*     (THE MESSAGE MUST RESIDE IN THE CURRENT BANK)
*
*  RESULT:  A -- RESPONSE KEYSTROKE
*           X, Y -- UNDEFINED
*
                REP        60
*
*
                ENTRY      OPMSGRPLY
*
                EXTRN      SCRNMODE
                PAGE
*
*  HARDWARE EQUATES
*
Z.REG           EQU        $FFD0
E.REG           EQU        $FFDF
*
KBPORT          EQU        $C008
*
BELL            EQU        $C040
*
VM0             EQU        $C050
VM1             EQU        $C052
VM2             EQU        $C054
VM3             EQU        $C056
*
E.T2            EQU        $FFE8
E.ACR           EQU        $FFEB
E.PCR           EQU        $FFEC
E.IFR           EQU        $FFED
E.IER           EQU        $FFEE
*
*  ZERO PAGE DECLARATIONS
*
                DSECT
ZPBASE          EQU        $200
                ORG        $0000                 ;ZERO PAGE DECLARATIONS
MSGPTR          DS         2                     ;MESSAGE POINTER
MSGIDX          DS         1
*
SCRNIDX         DS         1
SCRNPTR         DS         2
DATAPTR         DS         2
```

```
                STA     DEVNUM          ; SAVE DEVICE NUMBER.
                LDA     #1              ; FAKE OUT 'LOKVOL'
                STA     SCRTCH          ; TO THINK TO LOOK ONLY ONCE.
                STA     TOTDEVS
                LDA     #<VCB
                STA     VCBPTR+1
                STA     PATHNMH         ; (TO MAKE HARMLESS)
                LDA     #0
                STA     SISTER+PATHNMH
                LDX     VCBPTR
                STX     PATHNML
                STA     VCB,X           ; FORCE CURRENT VOLUME OFF LINE, THEN LOG WHATS THERE.
                JSR     FREEVCB         ; GO READ ROOT DIRECTORY.
                BCS     RTVOLNAM        ; RETURN ANY ERRORS
                LDX     VCBPTR          ; MAKE SURE VOLUME WAS LOGGED IN
                LDA     VCB,X
                BEQ     NOVOLM          ; RETURN ERROR
                RTS                     ; ELSE RETURN NORMALLY
NOVOLM          LDA     #VNFERR         ; TELL USER 'NO VOLUME'
                SEC
RTVOLNAM        TAX                     ; SAVE REAL ERROR WHILE DUPLICATE IS CHECKED
                LDA     DUPLFLAG
                BEQ     RTV1            ; BRANCH IF NOT DUPLICATE
                LDX     #DUPVOL
RTV1            TXA                     ; RECALL ERROR
                RTS

                CHN     VOLUME,4,1

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


End of File -- Lines: 596 Characters: 24986
```

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

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

                PAGE
GETMARK         LDY       #FCBMARK            ; MOVE CURRENT POSITION MARKER TO
GMARK1          LDA       (FCBPTR),Y          ; USER'S 4 BYTE BUFFER POINTED TO BY
                PHA                           ; C.MRKPTR IN SOS ZPAGE
                INY
                CPY       #FCBMARK+3          ; USE STACK AS TEMPORARY STORAGE FOR THREE BYTE
                BNE       GMARK1              ; POSITION VALUE.
                LDA       #0                  ; THE FOURTH (HIGHEST ORDER) BYTE IS ALWAYS ZERO.
                LDY       #3
                PHA
MOVMRK          PLA
                STA       (C.MRKPTR),Y        ; MOVE TO USER'S SPACE
                DEY                           ; IS THERE ANOTHER TO PULL FROM STACK?
                BPL       MOVMRK              ; YES, GET NEXT LOWER BYTE FROM STACK.
                CLC                           ; INDICATE NO ERROR.
                RTS
*
SETMARK         JSR       ADJMARK             ; MAKE ADJUSTMENTS TO REQUESTED MARK ACCORDING TO BASE.
                BCC       SMARK1              ; BRANCH IF ADJUSTMENT WAS VALID.
                RTS
SMARK1          LDX       #2                  ; NOW COMPARE END OF FILE WITH NEW
                LDY       #FCBEOF+2           ; POSITION TO BE SURE IT'S WITHIN
CMPEOF          LDA       TPOSLL,X            ; THE BOUNDS OF CURRENTLY DEFINED
                CMP       (FCBPTR),Y          ; LIMITS.
                BCC       CKSAMBLK            ; BRANCH IF MARK<EOF
                BNE       ERRMEOF             ; RETURN ERROR IF MARK>= EOF
                DEY
                DEX
                BPL       CMPEOF
                BMI       CKSAMBLK            ; BRANCH ALWAYS
ERRMEOF         LDA       #POSNERR            ; TELL USER MARK IS OUT OF RANGE.
                RTS                           ; (CARRY IS SET TO INDICATE ERROR)
*
ADJMARK         LDA       C.MARK+3            ; MAKE SURE FOURTH BYTE OF DISPLACE IS ZIP
                BNE       ERRPOSN             ; BRANCH TO ERR IF NOT
                LDX       #$FD                ; ANTICIPATE OTHER THAN BASE OF ZERO
                LDY       #FCBMARK            ; FURTHER ASSUME IT'S A BASE OFFSET FROM CURRENT POSITION
                LDA       C.BASE              ; NOW FIND OUT WHAT IT REALLY IS.
                LSR       A                   ; (CARRY SET=SUBTRACT, NON ZERO REMAINDER= OFFSET FROM EOF)
                BCS       SUBMARK
                BEQ       ADJMRK              ; BRANCH IF MARK IS FROM BEGINNING OF FILE
ADDPOSN         LDA       (FCBPTR),Y          ; ADD USER QUANTITY TO CURRENT
                ADC       C.MARK+3,X          ; POSITION TO FORM NEW POSITION.
                STA       >TPOSLL-$FD,X       ; (NOTE: ZERO PAGE REFERENCE WRAPS AROUND IN Z-PAGE)
                INY
                INX
                BNE       ADDPOSN             ; ADD ALL THREE BYTES
                BCS       ERRPOSN             ; BRANCH IF OVERFLOW
                BEQ       ADJMRK1             ; BRANCH ALWAYS
*
                PAGE
SUBMARK         BNE       SUBPOSN             ; BRANCH IF IT'S AN OFFSET FROM CURRENT POSITION
                LDY       #FCBEOF             ; OTHERWISE ASSUME OFFSET FROM END OF FILE.
SUBPOSN         LDA       (FCBPTR),Y          ; SUBTRACT USER QUANTITY TO FORM
                SBC       C.MARK+3,X          ; NEW POSITION. IF FINAL
                STA       >TPOSLL-$FD,X       ; RESULT IS L.T. ZERO, THEN REPORT
                INY                           ; POSITION ERROR...
                INX
                BNE       SUBPOSN
                BCS       ADJMRK1             ; BRANCH IF LEGAL POSITION CALCULATED.
ERRPOSN         LDA       #POSNERR
                SEC                           ; INDICATE ERROR
                RTS
*
ADJMRK          LDX       #2                  ; FIRST SET UP POSITION TEMPS USED
ADJMRK0         LDA       C.MARK,X            ; BY BOTH POSITION ROUTINES
                STA       TPOSLL,X
                DEX
                BPL       ADJMRK0
ADJMRK1         CLC                           ; NO ERRORS
                RTS
```

```
*
*
RDPOSN          EQU      *
CKSAMBLK        EQU      *
                LDY      #FCBMARK+1         ; FIRST TEST TO SEE IF NEW POSITION IS
                LDA      (FCBPTR),Y         ; WITHIN THE SAME (CURRENT) DATA BLOCK.
                AND      #$FE
                STA      SCRTCH
                INY                         ; BUMP TO ACCESS HIGHEST ORDER ADDRESS BYTE
                LDA      TPOSLH             ; GET MIDDLE BYTE OF NEW POSITION
                SEC
                SBC      SCRTCH
                STA      SCRTCH
                BCC      TYPMARK            ; BRANCH IF POSSIBLY L.T. CURRENT POSITION
                CMP      #2                 ; MUST BE WITHIN 512 BYTES OF BEGINNING OF CURRENT
                BCS      TYPMARK
                LDA      TPOSHI             ; NOW MAKE SURE WERE TALKIN ABOUT
                CMP      (FCBPTR),Y         ; THE SAME 64K CHUNK!
                BNE      TYPMARK            ; BRANCH IF WE AREN'T.
                JMP      SVMARK             ; IF WE IS, ADJUST FCB AND POSPTR AND RETURN.
*
TYPMARK         LDY      #FCBSTYP           ; NOW FIND OUT WHICH TYPE
                LDA      (FCBPTR),Y         ; OF FILE WE'RE POSITIONING ON.
                BEQ      FERRTYP            ; THERE IS NO SUCH TYPE AS ZERO, BRANCH NEVER!
                CMP      #4                 ; IS IT A TREE CLASS FILE?
                BCC      CHKDSKSW           ; YES, GO POSITION
                JMP      DIRMARK            ; NO, TEST FOR DIRECTORY TYPE.
*
CHKDSKSW        EQU      *                  ; MAKE SURE S/HE HASN'T MOVED THE VOLUME
                LDY      #FCBDEVN
                LDA      (FCBPTR),Y
                STA      DEVNUM             ; MAKE SURE DEVICE NUMBER PARM IS CURRENT
                JSR      TWRPROT1           ; PASSES DEVNUM (CHECK DISK SWITCH)
                LDA      DSWGLOB            ; DISK SWITCH GLOBAL
                BEQ      TREPOS             ; BRANCH IF NONE DETECTED
CHKDSKS1        JSR      VERFYVOL           ; MATCHES VCBPTR VS. DEVNUM
                BCC      TREPOS             ; BRANCH IF DISK HASN'T SWITCHED
                JSR      USRREQ             ; POLITELY ASK USER TO MOUNT
                BCC      CHKDSKS1           ; SAID HE DID, CHECK AGAIN
                LDA      #VNFERR            ; REFUSES TO MOUNT
                RTS
*
FERRTYP         LDY      #FCBREFN           ; CLEAR ILLEGALLY TYPED FCB ENTRY
                STA      (FCBPTR),Y
                LDA      #BADREFNUM         ; TELL EM THERE IS NO SUCH FILE
                SEC
                RTS
*
                PAGE
TREPOS          LDY      #FCBSTYP           ; USE STORAGE TYPE AS NUMBER
                LDA      (FCBPTR),Y         ; OF LEVELS (SINCE 1=SEED, 2=SAPLING, AND 3=TREE)
                STA      LEVELS
                LDY      #FCBSTAT           ; SINCE IT'S A DIFFERENT DATA
                LDA      (FCBPTR),Y         ; BLOCK, MUST NOT FORGET PREVIOUS DATA.
                AND      #DATMOD            ; THEREFORE, SEE IF PREVIOUS DATA WAS MODIFIED
                BEQ      POSNEW1            ; THEN DISK MUST BE UPDATED.
                JSR      WFCBDAT            ; GO WRITE CURRENT DATA BLOCK.
                BCS      POSERR             ; RETURN ANY ERROR ENCOUNTERED.
*
POSNEW1         LDY      #FCBMARK+2         ; TEST TO SEE IF CURRENT
                LDA      (FCBPTR),Y         ; INDEX BLOCK IS GOING TO BE USABLE...
                AND      #$FE               ; OR IN OTHER WORDS-
                STA      SCRTCH             ; IS NEW POSITION WITHIN 128K OF THE BEGINNING
                LDA      TPOSHI             ; OF CURRENT SAPLING LEVEL CHUNK.
                SEC
                SBC      SCRTCH
                BCC      POSNEW2            ; BRANCH IF A NEW INDEX BLOCK IS ALSO NEEDED
                CMP      #2                 ; NEW POSITION IS > THAN BEGINING OF OLD. IS IT WITHIN 128K?
                BCS      POSNEW2            ; BRANCH IF NOT.
                LDX      LEVELS             ; IS THE FILE WE'RE DEALING WITH A SEED?
                DEX
                BNE      DATLEVEL           ; NO, USE CURRENT INDEXES.
TSTINY          LDA      TPOSLH             ; IS NEW POSITION UNDER 512?
                LSR      A
                ORA      TPOSHI
                BNE      NOIDXDAT           ; NO, MARK BOTH DATA AND INDEX BLOCK AS UN-ALLOCATED.
                LDY      #FCBFRST
                LDA      (FCBPTR),Y         ; FIRST BLOCK IS ONLY BLOCK AND IT'S DATA!
                STA      BLOKNML
                INY
                LDA      (FCBPTR),Y         ; (HIGH BLOCK ADDRESS)
```

```
                JMP     RNEWPOS             ; GO READ IN BLOCK AND SET APPROPRIATE STATUSES.
*
                PAGE
POSNEW2         LDY     #FCBSTAT            ; GOTA CHECK TO SEE IF PREVIOUS
                LDA     (FCBPTR),Y          ; INDEX BLOCK WAS MODIFIED.
                AND     #IDXMOD
                BEQ     POSNIDX             ; READ IN OVER IT IF CURRENT IS UP TO DATE.
                JSR     WFCBIDX             ; GO UPDATE INDEX ON DISK (BLOCK ADDR IN FCB)
                BCS     POSERR
POSNIDX         LDX     LEVELS              ; BEFORE READING IN TOP INDEX, CHECK TO BE SURE
                CPX     #3                  ; THAT THERE IS A TOP INDEX...
                BEQ     POSINDEX            ; BRANCH IF FILE IS FULL BLOWN TREE.
                LDA     TPOSHI              ; IS NEW POSITION WITHIN RANGE OF A
                LSR     A                   ; SAPLING FILE (L.T. 128K)?
                PHP                         ; ANTICIPATE NO GOOD.
                LDA     #TOPALC+IDXALC+DATALC ; (TO INDICATE NO LEVEL IS ALLOCATED FOR NEW POSITION.)
                PLP                         ; Z FLAG TELLS ALL...
                BNE     NODATA              ; GO MARK 'EM ALL DUMMY.
                JSR     CLRSTATS            ; GO CLEAR STATUS BITS 0,1,2 (INDEX/DATA ALLOC STATUS).
                DEX                         ; (UNAFFECTED SINCE LOADED ABOVE) CHECK FOR SEED
                BEQ     TSTINY              ; IF SEED, CHECK FOR POSITION L.T. 512...
                JSR     RFCBFST             ; GO GET ONLY INDEX BLOCK
                BCS     POSERR              ; BRANCH IF ERROR
                LDY     #FCBIDXB            ; SAVE NEWLY LOADED INDEX BLOCK'S ADDRESS
                LDA     BLOKNML
                STA     (FCBPTR),Y
                INY
                LDA     BLOKNMH
                STA     (FCBPTR),Y
                BCC     DATLEVEL            ; BRANCH ALWAYS...
POSERR          SEC
                RTS
*
POSINDEX        JSR     CLRSTATS            ; CLEAR ALL ALLOCATION REQUIREMENTS FOR PREVIOUS POSITION
                JSR     RFCBFST             ; GET HIGHEST LEVEL INDEX BLOCK.
                BCS     POSERR
                LDA     TPOSHI              ; THEN TEST FOR A SAP LEVEL INDEX BLOCK
                LSR     A
                TAY
                LDA     (TINDX),Y
                INC     TINDX+1
                CMP     (TINDX),Y           ; (BOTH HI AND LO WILL BE ZERO IF NO INDEX EXISTS)
                BNE     SAPLEVEL
                CMP     #0                  ; ARE BOTH BYTES ZERO?
                BNE     SAPLEVEL
                DEC     TINDX+1             ; DON'T LEAVE WRONG POINTERS LAYING AROUND!
NOIDXDAT        LDA     #IDXALC+DATALC      ; SHOW NEITHER INDEX OR DATA BLOCK ALLOCATED.
                JMP     NODATA
*
                PAGE
SAPLEVEL        STA     BLOKNML             ; READ IN NEXT LOWER INDEX BLOCK
                LDA     (TINDX),Y           ; (HI ADDRESS)
                STA     BLOKNMH
                DEC     TINDX+1
                JSR     RFCBIDX             ; READ IN SAPLING LEVEL
                BCS     POSERR
DATLEVEL        LDA     TPOSHI              ; NOW GET BLOCK ADDRESS OF DATA BLOCK
                LSR     A
                LDA     TPOSLH              ; ( IF THERE IS ONE )
                ROR     A
                TAY
                LDA     (TINDX),Y           ; DATA BLOCK ADDRESS LOW
                INC     TINDX+1
                CMP     (TINDX),Y
                BNE     POSNEW3
                CMP     #0
                BNE     POSNEW3
                LDA     #DATALC             ; SHOW DATA BLOCK AS NEVER BEEN ALLOCATED
                DEC     TINDX+1
*
NODATA          LDY     #FCBSTAT
                ORA     (FCBPTR),Y          ; SET STATUS TO SHOW WHATS MISSIN'
                STA     (FCBPTR),Y
                LSR     A                   ; THROW AWAY BIT THAT SAYS DATA BLOCK UN-ALLOCATED
                LSR     A                   ; CUZ WE KNOW THAT. CARRY NOW INDICATES IF INDEX BLOCK
                JSR     ZIPDATA             ; ALSO IS INVALID AND NEEDS TO BE ZEROED (CARRY UNDISTURBED)
                BCC     SVMARK              ; BRANCH IF INDEX BLOCK DOESN'T NEED ZIPPIN.
ZIPIDX          STA     (TINDX),Y
                INY
                BNE     ZIPIDX
                INC     TINDX+1
```

```
ZPIDX1          STA       (TINDX),Y
                INY
                BNE       ZPIDX1
                DEC       TINDX+1               ; RESTORE PROPER ADDRESS
                JMP       SVMARK
*
ZIPDATA         LDA       #0                    ; ALSO IS INVALID AND NEEDS TO BE ZEROED.
                TAY
ZIPDAT0         STA       (DATPTR),Y            ; ZERO OUT DATA AREA
                INY
                BNE       ZIPDAT0
                INC       DATPTR+1
ZPDAT1          STA       (DATPTR),Y
                INY
                BNE       ZPDAT1
                DEC       DATPTR+1
                RTS
*
                PAGE
*
POSNEW3         STA       BLOKNML               ; GET DATA BLOCK OF NEW POSITION
                LDA       (TINDX),Y             ; (HI ADDRESS)
                DEC       TINDX+1
RNEWPOS         STA       BLOKNMH
                JSR       RFCBDAT
                BCS       PRITZ                 ; RETURN ANY ERROR
                JSR       CLRSTATS              ; SHOW WHOLE CHAIN IS ALLOCATED
SVMARK          LDY       #FCBMARK+2            ; UPDATE POSITION IN FILE CONTROL BLOCK
                LDX       #2
SVMRK1          LDA       (FCBPTR),Y            ; REMEMBER OLDMARK IN CASE
                STA       OLDMARK-FCBMARK,Y     ; CALLING ROUTINE FAILS LATER
                LDA       TPOSLL,X
                STA       (FCBPTR),Y
                DEY
                DEX                             ; MOVE 3 BYTE POSITION MARKER
                BPL       SVMRK1
*
                CLC                             ; LAST, BUT NOT LEAST, SET UP
                LDA       DATPTR                ; INDIRECT ADDRESS TO BUFFER PAGE POINTED
                STA       POSPTR                ; TO BY THE CURRENT POSITION MARKER.
                LDA       TPOSLH
                AND       #1
                ADC       DATPTR+1
                STA       POSPTR+1
                LDA       SISDATP
                STA       SISPOSP               ; SISTER PAGE BYTE ALSO.
                RTS                             ; CARRY SHOULD ALWAYS BE CLEAR
PRITZ           SEC                             ; RANDOM ERROR
                RTS                             ; RETURN
*
*
CLRSTATS        LDY       #FCBSTAT              ; CLEAR ALLOCATION STATES FOR DATA BLOCK
                LDA       (FCBPTR),Y            ; AND BOTH LEVELS OF INDEXES.
                AND       #$FF-TOPALC-IDXALC-DATALC
                STA       (FCBPTR),Y            ; THIS SAYS THAT EITHER THEY EXIST CURRENTLY
                RTS                             ; OR THAT THEY'RE UNNECESSARY FOR CURRENT POSITION.
*
                PAGE
*
DIRMARK         CMP       #DIRTYP               ; IS IT A DIRECTORY?
                BEQ       DIRPOS                ; YES...
                LDA       #CPTERR               ; NO, THERE IS A COMPATABLITY PROBLEM-
                JSR       SYSERR                ; THE DAMN THING SHOULD OF NEVER BEEN OPENED!
*
DIRPOS          LDA       SCRTCH                ; RECOVER RESULTS OF PREVIOUS SUBTRACTION.
                LSR       A                     ; USE DIFFERENCE AS COUNTER AS TO HOW MANY
                STA       CNTENT                ; BLOCKS MUST BE READ TO GET TO NEW POSITION.
                LDY       #FCBMARK+1            ; TEST FOR POSITION DIRECTION.
                LDA       (FCBPTR),Y
                CMP       TPOSLH                ; CARRY INDICATES DIRECTION...
                BCC       DIRFWRD               ; IF SET, POSITION FORWARD.
DIRVRSE         LDY       #0                    ; OTHERWISE, READ DIRECTORY FILE IN REVERSE ORDER.
                JSR       DIRPOS1               ; READ PREVIOUS BLOCK.
                BCS       DRPOSERR              ; BRANCH IF ANYTHING GOES WRONG.
                INC       CNTENT                ; COUNT UP TO 128
                BPL       DIRVRSE               ; LOOP IF THERE IS MORE BLOCKS TO PASS OVER.
                BMI       SVMARK                ; BRANCH ALWAYS.
*
DIRFWRD         LDY       #2                    ; POSITION IS FORWARD FROM CURRENT POSITION.
                JSR       DIRPOS1               ; READ NEXT DIRECTORY BLOCK.
                BCS       DRPOSERR
```

```
                DEC      CNTENT
                BNE      DIRFWRD           ; LOOP IF POSITION NOT FOUND IN THIS BLOCK.
                BEQ      SVMARK            ; BRANCH ALWAYS.
*
DIRPOS1         LDA      (DATPTR),Y        ; GET LINK ADDRESS OF PREVIOUS OR
                STA      BLOKNML           ; NEXT DIRECTORY BLOCK.
                INY                        ; BUT FIRST BE SURE THERE IS A LINK.
                CMP      (DATPTR),Y
                BNE      DIRPOS2           ; BRANCH IF CERTAIN LINK EXISTS
                CMP      #0                ; ARE BOTHE LINK BYTES 0?
                BNE      DIRPOS2           ; NOPE, JUST HAPPEN TO BE THE SAME VALUE.
                LDA      #EOFERR           ; SOMETHING IS WRONG WITH THIS DIRECTORY FILE!
DRPOSERR        SEC                        ; INDICATE ERROR
                RTS
*
DIRPOS2         LDA      (DATPTR),Y        ; (HIGH ORDER BLOCK ADDRESS)
                STA      BLOKNMH
* DROP INTO 'RFCBDAT' (READ FILE'S DATA BLOCK)
*
* NOTE: FOR DIRECTORY POSITIONING NO OPTIMIZATION HAS BEEN
* DONE SINCE DIRECTORY FILES WILL ALMOST ALWAYS BE LESS
* THAN 6 BLOCKS. IF MORE SPEED IS REQUIRED OR DIRECTORY
* TYPE FILES ARE TO BE USED FOR OTHER PURPOSES REQUIRING
* MORE BLOCKS, THEN THE RECOMMENDED METHOD IS TO CALL
* 'RFCBDAT' FOR THE FIRST BLOCK AND GO DIRECTLY TO
* DEVICE (VIA JMP (IOUNITL)) HANDLER FOR SUBSEQUENT
* ACCESSES.
* ALSO NOTE THAT NO CHECKING IS DONE FOR READ/WRITE
* ENABLE SINCE A DIRECTORY FILE CAN ONLY BE OPENED
* FOR READ ACCESS.
*
                PAGE
*
RFCBDAT         LDA      #RDCMD            ; SET READ COMMAND.
                STA      DHPCMD
                LDX      #DATPTR           ; USE X TO POINT AT ADDRESS OF DATA BUFFER
                JSR      FILEIO1           ; GO DO FILE INPUT.
                LDY      #FCBDATB          ; SAVE BLOCK NUMBER JUST READ IN FCB.
                BCC      FCBLOKNM          ; BRANCH IF NO ERRORS HAPPENED.
                RTS                        ; RETURN ERROR
*
RFCBIDX         LDA      #RDCMD            ; PREPARE TO READ IN INDEX BLOCK.
                STA      DHPCMD
                LDX      #TINDX            ; POINT AT ADDRESS OF CURRENT INDEX BUFFER
                JSR      FILEIO1           ; GO READ INDEX BLOCK.
                BCS      RDFCBERR          ; REPORT ERROR
                LDY      #FCBIDXB          ; SAVE BLOCK ADDRESS OF THIS INDEX IN FCB.
FCBLOKNM        LDA      BLOKNML
                STA      (FCBPTR),Y
                INY
                LDA      BLOKNMH
                STA      (FCBPTR),Y
                CLC
RDFCBERR        RTS
*
RFCBFST         LDX      #TINDX            ; POINT AT ADDRESS OF INDEX BUFFER
                LDY      #FCBFRST          ; AND BLOCK ADDRESS OF FIRST FILE BLOCK IN FCB
                LDA      #RDCMD            ; AND LASTLY, MAKE IT A READ!
* DROP INTO DOFILEIO
*
DOFILEIO        STA      DHPCMD            ; SAVE COMMAND.
                LDA      (FCBPTR),Y        ; GET DISK BLOCK ADDRESS FROM FCB.
                STA      BLOKNML
                INY                        ; BLOCK ZERO NOT LEGAL.
                CMP      (FCBPTR),Y
                BNE      FILEIO
                CMP      #0                ; ARE BOTH BYTES ZERO?
                BNE      FILEIO            ; NO, CONTINUE WITH REQUEST.
                LDA      #ALCERR           ; OTHERWISE REPORT ALLOCATION ERROR.
                JSR      SYSDEATH          ; NEVER RETURNS...
*
                PAGE
FILEIO          LDA      (FCBPTR),Y        ; GET HIGH ADDRESS OF DISK BLOCK
                STA      BLOKNMH
FILEIO1         LDA      0,X               ; GET MEMORY ADDRESS OF BUFFER FROM
                STA      DBUFPL            ; S.O.S. ZERO PAGE POINTED TO BY
                JSR      WRAPADJ           ;GO ADJUST FOR BANK CROSSING <SRS 82.162>
                LDA      1,X
                STA      DBUFPH            ; SET HI BYTE
                LDA      SISTER+1,X        ; AND BANK PAIR BYTE. <SRS 82.162>
                STA      SISBPH
```

```
                LDY     #FCBDEVN
                LDA     (FCBPTR),Y      ; OF COURSE HAVING THE DEVICE NUMBER
                STA     DEVNUM          ; WOULD MAKE THE WHOLE OPERATION MORE MEANINGFUL...
FILEIO2         LDA     #2              ; ALSO, SET UP BYTE COUNT TO 512 AND
                STA     RQCNTH          ; SET 'BYTES READ' POINTER TO
                STA     IOACCESS        ; (INTERUPT! SET TO INDICATE REG CALL MADE TO DEV HANDLER. RETURN
INTERUPT!)
                LDA     #>TRASH         ; A PLACE TO THROW BYTES READ AWAY
                STA     BRDPTR
                LDA     #<TRASH         ; LOCALLY DEFINED
                STA     BRDPTR+1
                LDA     #0              ; SO THAT IT DOESN'T MESS UP ANY OTHER DATA.
                STA     RQCNTL
                STA     SSBRDPH         ; ('BYTES READ' IS THROWN AWAY)
RPEATIO1        LDA     DEVNUM          ; TRANSFER THE DEVICE NUMBER FOR DISPATCHER TO CONVERT TO UNIT NUMBER.
                STA     UNITNUM
RPEATIO0        LDY     #$9             ; PREPARE TO SAVE DEVICE PARMS
SAVPRMS         LDA     DEVICE,Y        ; MOVE FROM Z PAGE
                STA     RPTBLOK,Y       ; TO MY OWN SPACE
                DEY                     ; FROM $C9 THROUGH $C0
                BPL     SAVPRMS
DMGRGO          EQU     *               ; CALL EXTERNAL DEVICE MANAGER
                LDA     #0
                STA     SERR            ; CLEAR GLOBAL ERROR VALUE
                JSR     DMGR            ; CALL THE DRIVER
                BCC     RRITZ           ; RTS IF NO ERRORS
                CMP     #XDISKSW        ; DISKSWITCH ITERATES
                BEQ     RPEATIO2        ; BRANCH IF DISK SWITCH AND REPEAT I/O REQUEST
                SEC                     ; REPORT ERROR
RRITZ           RTS
RPEATIO2        LDY     #$9             ; LENGTH OF PARM BLOCK
GETPRMS         LDA     RPTBLOK,Y
                STA     DEVICE,Y        ; RESTORE POSSIBLY DISTURBED PARM BLOCK
                DEY
                BPL     GETPRMS
                JMP     DMGRGO          ; AND TRY THE I/O AGAIN
*
*
TRASH           DS      2               ; ONLY USED TO PUT BYTES READ TO SLEEP
RPTBLOK         DS      10              ; DMGR PARM SAVE BLOCK
*
*
WFCBFST         LDY     #FCBDEVN        ; FETCH THE
                LDA     (FCBPTR),Y      ;  DEVICE NUMBER
                TAX                     ;    AND UPDATE
                JSR     UPBMAP          ;      ITS BITMAP
                LDX     #TINDX          ; POINT AT ADDRESS OF INDEX BLOCK
                LDY     #FCBFRST        ; AND THE DISK ADDRESS OF FILE'S FIRST BLOCK IN FCB
                LDA     #WRTCMD         ; LASTLY, MAKE IT A WRITE REQUEST.
                JMP     DOFILEIO        ; AND GO DO IT!
*
WFCBDAT         LDX     #DATPTR
                LDY     #FCBDATB        ; POINT AT MEMORY ADDRESS WITH X AND DISK ADDRESS WITH Y.
                LDA     #WRTCMD         ; WRITE DATA BLOCK.
                JSR     DOFILEIO
                BCS     FILIOERR        ; REPORT ANY ERRORS
                LDA     #$FF-DATMOD     ; MARK DATA STATUS AS CURRENT.
                JMP     FCBUPDAT
*
WFCBIDX         LDY     #FCBDEVN        ; MAKE SURE
                LDA     (FCBPTR),Y      ;  THE BITMAP
                TAX                     ;    FOR THIS DEVICE ("X")
                JSR     UPBMAP          ;      IS UPDATED
                LDX     #TINDX          ; POINT AT ADDRESS OF INDEX BUFFER
                LDY     #FCBIDXB        ; AND BLOCK ADDRESS OF THAT INDEX BLOCK.
                LDA     #WRTCMD
                JSR     DOFILEIO        ; GO WRITE OUT INDEX BLOCK.
                BCS     FILIOERR        ; REPORT ANY ERRORS
                LDA     #$FF-IDXMOD     ; MARK INDEX STATUS AS CURRENT.
FCBUPDAT        LDY     #FCBSTAT        ; CHANGE STATUS BYTE TO
                AND     (FCBPTR),Y      ; REFLECT SUCCESSFUL DISK FILE UPDATE.
                STA     (FCBPTR),Y      ; (CARRY IS UNAFFECTED)
FILIOERR        RTS
*
*
                PAGE
OPEN            JSR     FINDFILE        ; FIRST OF ALL LOOK UP THE FILE...
                BCC     OPEN0
                CMP     #BADPATH        ; IS AN ATTEMPT TO OPEN A ROOT DIRECTORY?
                BNE     ERROPN          ; NO, PASS BACK ERROR
*
```

```
OPEN0           JSR     TSTOPEN             ; FIND OUT IF ANY OTHER FILES ARE WRITING
                BCC     OPEN1               ; TO THIS SAME FILE. (BRANCH IF NOT)
ERRBUSY         LDA     #FILBUSY            ; REPORT SHARED ACCESS NOT ALLOWED.
ERROPN          SEC
                RTS                         ; RETURN ERROR.
*
OPEN1           LDA     DATPTR              ; GET ADDRESS OF FIRST FREE FCB FOUND
                STA     FCBPTR              ; DURING TEST OPEN SEQUENCE AND USE
                LDA     DATPTR+1            ; IT AS FILE CONTROL AREA. IF HIGH BYTE OF
                STA     FCBPTR+1            ; POINTER IS ZERO, THEN NO FCB
                BNE     ASGNFCB             ; IS AVAILABLE FOR USE.
                LDA     #FCBFULL            ; REPORT FCB FULL ERROR.
                SEC
                RTS
*
ASGNFCB         LDY     #$1F                ; ASSIGN FCB, BUT FIRST
                LDA     #0                  ; CLEAN OUT ANY OLD RUBBISH LEFT AROUND...
CLRFCB          STA     (FCBPTR),Y
                DEY
                BPL     CLRFCB
                LDY     #FCBENTN            ; NOW BEGIN CLAIM BY MOVING IN FILE
FCBOWNR         LDA     D.DEV-1,Y           ; OWNERSHIP INFORMATION.
                STA     (FCBPTR),Y          ; NOTE: THIS CODE DEPENDS UPON THE DEFINED
                DEY                         ; ORDER OF BOTH THE FCB AND DIRECTORY ENTRY
                BNE     FCBOWNR             ; BUFFER (D.). BEWARE OF CHANGES!!! *************
                LDA     DFIL+D.STOR         ; GET STORAGE TYPE.
                LSR     A                   ; STRIP OFF FILE NAME LENGTH.
                LSR     A
                LSR     A                   ; (BY DIVIDING BY 16)
                LSR     A
                TAX                         ; SAVE IN X FOR LATER TYPE COMPARISON
                LDY     #FCBSTYP
                STA     (FCBPTR),Y          ; SAVE STORAGE TYPE.
                LDA     C.OPLSTLN           ; IS THERE AN OPEN LIST?
                BEQ     DEFOPEN             ; NO, USE DEFAULT REQUST ACCESS...
                LDY     #0                  ; YES, FIND OUT WHAT ACCESS IS REQUESTED.
                LDA     (C.OPLIST),Y        ; IF REQ-ACCESS IS ZERO, THEN
                BEQ     DEFOPEN             ; USE DEFAULTS...
                AND     DFIL+D.ATTR         ; CHECK REQUEST AGAINST ATTRIBUTES.
                CMP     (C.OPLIST),Y        ; WERE ALL ACCESS REQUESTS SATISFIED?
                BEQ     SVATTRB             ; YES, SAVE ATTRIBUTES.
                LDA     #ACCSERR            ; REPORT ACCESS REQUEST CAN'T BE MET.
                SEC
                RTS
                PAGE
DEFOPEN         LDA     DFIL+D.ATTR         ; GET FILES ATTRIBUTES AND
                AND     #READEN+WRITEN      ; USE IT AS A DEFAULT ACCESS REQUEST.
SVATTRB         LDY     #FCBATTR
                CPX     #DIRTYP             ; IF DIRECTORY, DON'T ALLOW WRITE ENABLE
                BNE     SVATTR1
                AND     #READEN
SVATTR1         STA     (FCBPTR),Y
                AND     #WRITEN             ; CHECK FOR WRITE ENABLED REQUESTED.
                BEQ     OPEN2               ; BRANCH IF READ ONLY OPEN.
                LDA     TOTENT              ; OTHERWISE, BE SURE NO ONE ELSE IS READING SAME
                BNE     ERRBUSY             ; FILE (SET UP BY TSTOPEN).
OPEN2           LDA     DFIL+D.COMP         ; OH, BY THE WAY... IS THIS FILE
                BEQ     OPEN3               ; COMPATABLE WITH VERSION 0000? ***************
ERRCMPAT        LDA     #CPTERR             ; REPORT FILE IS INCOMPATABLE!
                SEC
                RTS
*
OPEN3           CPX     #TRETYP+1           ; IS IT A TREE TYPE FILE?
                BCC     OPEN4               ; TEST FOR FURTHER COMPATABLITY. IT MUST
                CPX     #DIRTYP             ; BE EITHER A TREE OR A DIRECTORY.
                BNE     ERRCMPAT            ; REPORT INCOMPATABLE.
OPEN4           LDY     #FCBFRST            ; MOVE ADDRESS OF FIRST BLOCK OF FILE
                LDA     DFIL+D.FRST         ; INTO FCB. NO CHECKING IS DONE FOR VALIDITY.
                STA     (FCBPTR),Y
                STA     BLOKNML
                INY
                LDA     DFIL+D.FRST+1
                STA     (FCBPTR),Y          ; NOTE: THE FCB HAS NOT BEEN OFFICIALLY
                STA     BLOKNMH             ; CLAIMED YET. TO DO THIS, THE FIRST BYTE
                LDY     #FCBEOF             ; MUST CONTAIN A VALID REFERENCE NUMBER.
EOFCBMV         LDA     DFIL+D.EOF-FCBEOF,Y ; MOVE CURRENT END OF FILE
                STA     (FCBPTR),Y          ; TO FCB.
                INY
                CPY     #FCBEOF+3
                BNE     EOFCBMV
                LDA     DFIL+D.USAGE
```

```
                STA     (FCBPTR),Y          ; AND CURRENT BLOCK COUNT OF FILE.
                INY
                LDA     DFIL+D.USAGE+1
                STA     (FCBPTR),Y
                LDA     C.OPLSTLN           ; NOW THAT WE'VE COME THIS FAR, FIND
                BEQ     DEFBUFR             ; OUT WHICH TYPE OF BUFFER AND ALLOCATE IT!
                CMP     #1                  ; WAS IT ONLY TO SET ATTRIBUTES?
                BEQ     DEFBUFR
                CMP     #4                  ; IS A FULL ADDRESS INCLUDED?
                BEQ     UBUFSPEC
                LDA     #BADLSTCNT
                SEC
                RTS
*
                PAGE
UBUFSPEC        LDY     #1                  ; (INDEX TO 'PAGECNT' OF OPEN LIST)
                LDA     (C.OPLIST),Y        ; IS USER SPECIFING THE BUFFER?
                BEQ     DEFBUFR             ; NO, USE DEFAULT BUFFER (DYNAMIC)
                CPX     #TRETYP+1           ; IF TREE TYPE FILE, THEN AT LEAS 4 PAGES ARE NEEDED.
                BCC     ONEKTST             ; BRANCH IF TREE TYPE.
                CMP     #2                  ; DID USER GIVE AT LEAST 2 PAGES FOR DIRECTORY TYPE?
                BCS     FIXDBUF             ; YES, LOG IT WITH BUFFER MANAGER
ERRBTS          LDA     #BTSERR             ; REPORT NOT ENOUGH BUFFER SPACE.
                SEC
                RTS
*
ONEKTST         CMP     #4                  ; IS THERE AT LEAST ONE KILOBYTE BUFFER FOR TREES?
                BCC     ERRBTS              ; NO, THEN TO HELL WITH IT!.
FIXDBUF         JSR     REQFXBUF            ; CALL BOB AND ASK FOR HIM TO FIX IT...
                BCC     FCBUFFER            ; GO SAVE BUFFER NUMBER.
ERROPN1         RTS                         ; RETURN ANY ERROR ENCOUNTERED.
*
DEFBUFR         LDA     #4                  ; ASSUME TREE FILE (4 PAGES REQUIRED)
                CPX     #TRETYP+1
                BCC     BUFREQST            ; BRANCH IF IT IS A TREE.
                LDA     #2                  ; OTHERWIZE, WE JUST NEED TWO PAGES.
BUFREQST        JSR     REQBUF              ; CALL BOB TO ALLOCATE A DYNAMIC BUFFER.
                BCS     ERROPN1             ; REPORT ANY ERRORS.
FCBUFFER        LDY     #FCBBUFN            ; SAVE BUFFER NUMBER AND THEN
                STA     (FCBPTR),Y          ; FIND OUT WHERE IT IS.
                JSR     GTBUFFRS            ; HAVE BOB RETURN ADDRESS IN DATA & INDEX POINTERS.
                BCS     ERROPEN2            ; IF ERROR, FREE BUFFER BEFOR RETURNING.
                LDY     #FCBREFN            ; NOW CLAIM FCB FOR THIS FILE.
                LDA     CNTENT              ; THIS WAS SET UP BY 'TSTOPEN'.............
                STA     (FCBPTR),Y
                LDY     #FCBLEVL            ; MARK LEVEL
                LDA     LEVEL               ; AT WHICH
                STA     (FCBPTR),Y          ; FILE WAS OPENED
                LDY     #FCBSTYP            ; GET STORAGE TYPE AGAIN.
                LDA     (FCBPTR),Y          ; FILE MUST BE POSITIONED TO BEGINNING.
                CMP     #TRETYP+1           ; IS IT A TREE FILE?
                BCS     OPNDIR              ; NO, ASSUME IT'S A DIRECTORY.
                LDA     #$FF                ; FOOL THE POSITION ROUTINE INTO GIVING
                LDY     #FCBMARK            ; A VALID POSITION WITH PRELOADED DATA, ETC.
OPNPOS          STA     (FCBPTR),Y
                INY
                CPY     #FCBMARK+3
                BNE     OPNPOS
                LDY     #2                  ; SET DESIRED POSITION TO ZERO.
                LDA     #0
OPNPOS1         STA     TPOSLL,Y
                DEY
                BPL     OPNPOS1
                JSR     RDPOSN              ; LET TREE POSITION ROUTINE DO THE REST.
                BCC     OPENDONE            ; BRANCH IF SUCCESSFUL.
*
                PAGE
ERROPEN2        PHA                         ; SAVE ERROR CODE.
                LDY     #FCBBUFN            ; SINCE ERROR WAS ENCOUNTERED BEFORE FILE
                LDA     (FCBPTR),Y          ; WAS SUCCESSFULLY OPENED, THEN
                JSR     RELBUF              ; IT'S NECESSARY TO FREE THE BUFFER AND
                LDY     #FCBREFN            ; FILE CONTROL BLOCK.
                LDA     #0
                STA     (FCBPTR),Y
                PLA
                SEC
                RTS
*
OPNDIR          JSR     RFCBDAT             ; READ IN FIRST BLOCK OF DIRECTORY FILE.
                BCS     ERROPEN2            ; RETURN ANY ERROR AFTER FREEING BUFFER & FCB
OPENDONE        LDY     #VCBOPNC            ; INCREMENT OPEN COUNT FOR THIS
```

```
* ONLY ADDRESSES IN THE RANGE $8200-$8E00 WILL BE ADJUSTED.
*
* UPON EXIT, A CONTAINS HIGH BYTE OF ADDRESS & Y CONTAINS UPDATED X-BYTE.
* THIS ROUTINE LEAVES X UNCHANGED.
*
WRAPADJ         LDA       1,X                 ; GET HIGH ADDRESS BYTE <SRS 82.162>
                LDY       SISTER+1,X          ; CHECK X-BYTE <SRS 82.162>
                BPL       WRAPDNE             ; NOT AN EXTENDED ADDRESS. <SRS 82.162>
                CMP       #$82                ; DOES IT NEED UPDATING? <SRS 82.162>
                BCC       WRAPDNE             ; NO <SRS 82.162>
                CPY       #$8F                ; SPECIAL BANK? <SRS 82.162>
                BCS       WRAPDNE             ; NO <SRS 82.162>
                AND       #$7F                ; ADJUST THE ADDRESS <SRS 82.162>
                STA       1,X                 ; UPDATE <SRS 82.162>
                INC       SISTER+1,X          ; INCREMENT X-BYTE <SRS 82.162>
                INY                           ; UPDATE Y ALSO <SRS 82.162>
*
WRAPDNE         RTS                           ; RETURN VALID HIGH ADDRESS AND BANK BYTE.

                CHN       CLOSE/EOF,4,2

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

End of File -- Lines: 669 Characters: 27951

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

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

                SBTL       "SOS 1.1 SYSTEM CALL MANAGER"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGSCMGR
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* SYSTEM CALL MANAGER (VERSION = 1.1O  )
*                     (DATE    = 8/04/81)
*
* THE SYSTEM CALL MANAGER:
* (1) RETRIEVE THE SYSCALL #,
* (2) DETERMINE THE LOCATION OF THE SYSTEM CALL PARMS AND
*     MOVE THEM TO THE SOS ZPAGE,
* (3) TRANSFER CONTROL TO THE APPROPRIATE INTERFACE MANAGER,
*     (FILE,DEVICE,UTILITY,MEMORY)
*
                REP        60
*
                ENTRY      SCMGR
*
                EXTRN      FMGR
                EXTRN      DMGR
                EXTRN      UMGR
                EXTRN      MMGR
                EXTRN      DBUGBRK
*
                EXTRN      SYSERR
                EXTRN      SERR
                EXTRN      BADSCNUM
                EXTRN      BADCZPAGE
                EXTRN      BADXBYTE
                EXTRN      BADSCPCNT
                EXTRN      BADSCBNDS
*
                EXTRN      SZPAGE
                EXTRN      SXPAGE
                EXTRN      CZPAGE
                EXTRN      CXPAGE
                EXTRN      CSPAGE
                PAGE
                REP        60
*
* SYSTEM CALL PARAMETER DEFINITION TABLES
*
* EACH ENTRY IS FOUR BYTES LONG.  THE FIRST BYTE CONTAINS THE
* NUMBER OF PARMS IN THE CALL.  THE REMAINING SIX NIBBLES, EACH
* DEFINE A PARAMETER IN THE CALL.  THE FIRST BIT OF THE
* NIBBLE DEFINES WHETHER THE PARM IS INPUT (0) OR OUTPUT (1).
* THE NEXT BIT DEFINES WHETHER THE PARM IS BY VALUE (0)
* OR BY REFERENCE (1).  THE FINAL TWO BITS SPECIFY THE
* PARM LENGTH IN BYTES (E.G. 0=LENGTH OF 1, 3=LENGTH OF 4 BYTES)
*
                REP        60
*
*   FILE SYSTEM CALL DEFINITIONS
*
FSC.CNT         EQU        $13
FSC.TBL         EQU        *
                DFB        $3,$5D,$00,$00      ; SCNUM=$C0 - CREATE
                DFB        $1,$50,$00,$00      ;   "  =$C1 - DESTROY
                DFB        $2,$55,$00,$00      ;   "  =$C2 - RENAME
                DFB        $3,$5D,$00,$00      ;   "  =$C3 - SET.FILE.INFO
                DFB        $3,$5D,$00,$00      ;   "  =$C4 - GET.FILE.INFO
                DFB        $4,$55,$99,$00      ;   "  =$C5 - VOLUME
                DFB        $1,$50,$00,$00      ;   "  =$C6 - SET.PREFIX
                DFB        $2,$50,$00,$00      ;   "  =$C7 - GET.PREFIX
```

```
                DFB     $4,$58,$D0,$00      ;   "  =$C8 - OPEN
                DFB     $3,$00,$00,$00      ;   "  =$C9 - NEW.LINE
                DFB     $4,$05,$19,$00      ;   "  =$CA - READ
                DFB     $3,$05,$10,$00      ;   "  =$CB - WRITE
                DFB     $1,$00,$00,$00      ;   "  =$CC - CLOSE
                DFB     $1,$00,$00,$00      ;   "  =$CD - FLUSH
                DFB     $3,$00,$30,$00      ;   "  =$CE - SET.MARK
                DFB     $2,$0B,$00,$00      ;   "  =$CF - GET.MARK
                DFB     $3,$00,$30,$00      ;   "  =$D0 - SET.EOF
                DFB     $2,$0B,$00,$00      ;   "  =$D1 - GET.EOF
                DFB     $1,$00,$00,$00      ;   "  =$D2 - SET.LEVEL
                DFB     $1,$80,$00,$00      ;   "  =$D3 - GET.LEVEL
                PAGE
*
*   DEVICE SYSTEM CALL DEFINITIONS
*
DSC.CNT         EQU     5
DSC.TBL         EQU     *
                DFB     $5,$05,$11,$90      ; SCNUM=$80 - D.READ
                DFB     $4,$05,$11,$00      ;   "  =$81 - D.WRITE
                DFB     $3,$00,$50,$00      ;   "  =$82 - D.STATUS
                DFB     $3,$00,$50,$00      ;   "  =$83 - D.CONTROL
                DFB     $2,$58,$00,$00      ;   "  =$84 - GET.DEV.NUM
                DFB     $4,$05,$D0,$00      ;   "  =$85 - D.INFO
*
*   UTILITY SYSTEM CALL DEFINITIONS
*
USC.CNT         EQU     5
USC.TBL         EQU     *
                DFB     $1,$00,$00,$00      ; SCNUM=$60 - SET.FENCE
                DFB     $1,$80,$00,$00      ;   "  =$61 - GET.FENCE
                DFB     $1,$50,$00,$00      ;   "  =$62 - SET.TIME
                DFB     $1,$50,$00,$00      ;   "  =$63 - GET.TIME
                DFB     $2,$0B,$00,$00      ;   "  =$64 - JOYSTICK
                DFB     $0,$00,$00,$00      ;   "  =$65 - COLD.START
*
*   MEMORY SYSTEM CALL DEFINITIONS
*
MSC.CNT         EQU     5
MSC.TBL         EQU     *
                DFB     $4,$11,$08,$00      ; SCNUM=$40 - REQUEST.SEG
                DFB     $6,$00,$99,$98      ;   "  =$41 - FIND.SEG
                DFB     $3,$00,$90,$00      ;   "  =$42 - CHANGE.SEG
                DFB     $5,$09,$99,$80      ;   "  =$43 - GET.SEG.INFO
                DFB     $2,$18,$00,$00      ;   "  =$44 - GET.SEG.NUM
                DFB     $1,$00,$00,$00      ;   "  =$45 - RELEASE.SEG
*
*   DEBUG SYSTEM CALL DEFINITION
*
DBUG            EQU     $FE
                PAGE
                REP     60
*
* DATA DECLARATIONS
*
                REP     60
Z.REG           EQU     $FFD0
SP.SAVE         EQU     $01FF
Z.SAVE          EQU     $01FD
B.SAVE          EQU     $01FC
*
ADR.LOW         EQU     $2000               ; LOW    ADDRESS   (BOUNDS CHECKING)
ADR.HIGH        EQU     $B800               ; HIGH   ADDRESS
ADR.MID         EQU     $A000               ; MIDDLE ADDRESS
*
* SCMGR'S VARIABLES
*
SCM.VARS        EQU     $E0
SCNUM           EQU     SCM.VARS+0          ; SYSTEM CALL NUMBER
SCRNUM          EQU     SCM.VARS+0          ; SYSTEM CALL REQUEST NUMBER
SCPTR           EQU     SCM.VARS+1          ;&2  SYSTEM CALL POINTER
MOVE.VARS       EQU     SCPTR+2             ; !! (LOOKOUT) !!
*
*
F.TPARMX        EQU     $A0                 ; FILE SYS CALL PARM START LOC
D.TPARMX        EQU     $C0                 ; DEVICE SYS CALL PARM START LOC
U.TPARMX        EQU     $C0                 ; UTILITY SYS CALL PARM START LOC
M.TPARMX        EQU     $60                 ; MEMORY SYS CALL PARM START LOC
*
* MOVE.PARM'S VARIABLES
*
```

```
TPARMX          EQU        MOVE.VARS+0         ; TARGET ADR OF SYS CALL PARMS
DFN.PTR         EQU        MOVE.VARS+1         ;&2
DFN.PTRX        EQU        MOVE.VARS+3
SCPTRX          EQU        MOVE.VARS+4
RGHT.NIB        EQU        MOVE.VARS+5
SCT.DFN         EQU        MOVE.VARS+6
SCT.DCNT        EQU        MOVE.VARS+7
PARM.CNT        EQU        MOVE.VARS+8
                PAGE
                REP        60
*
* SYSTEM CALL MANAGER
*
                REP        60
*
SCMGR           EQU        *
                LDA        #<SZPAGE            ; SET Z REG TO SOS ZPAGE
                STA        Z.REG
*
* SET SYSTEM X BYTES TO ABSOLUTE ADDRESS MODE.
*
                LDA        #0
                STA        SXPAGE+SCPTR+1
                STA        SERR                ; AND INIT SYSTEM ERR CODE
*
* CALLER'S Z REG MUST BE $1A !!
* (B REG NOT CHECKED)
*
                LDA        Z.SAVE
                CMP        #<CZPAGE
                BEQ        SCM005
                LDA        #>BADCZPAGE
                JSR        SYSERR              ; EXIT TO DISPATCHER
*
* RETRIEVE CALLER'S PC ON HIS STACK
*
SCM005          LDX        SP.SAVE
                LDA        CSPAGE+6,X
                STA        SCPTR+1
                LDA        CSPAGE+5,X
                STA        SCPTR
                BNE        SCM010              ; AND POINT IT TO SYS CALL NUM
                DEC        SCPTR+1
SCM010          DEC        SCPTR
*
* ADVANCE CALLER'S PC ON HIS STACK.
*
                CLC
                LDA        CSPAGE+5,X
                ADC        #2
                STA        CSPAGE+5,X
                BCC        SCM020
                INC        CSPAGE+6,X
*
* RETRIEVE SYSTEM CALL NUMBER
*
SCM020          LDY        #0
                LDA        (SCPTR),Y
                CMP        #DBUG
                BNE        SCM025
                JSR        DBUGBRK             ; DEBUG SYSTEM CALL
SCM025          STA        SCNUM
*
* RETRIEVE SYSTEM CALL PARAMETER ADDRESS
*
                INY
                LDX        #>SCPTR
                JSR        POINTER
                BCC        SCM030
                RTS                            ; ERROR EXIT
*
* CASE INTERFACE CODE OF SYSTEM CALL NUMBER
*  (INTERFACE CODE STRIPPED, LEAVING REQUEST CODE)
*
SCM030          LDA        #$20
                BIT        SCNUM
                BPL        SCM050
                LDA        SCNUM
                AND        #$3F
                STA        SCRNUM
                BVC        SCM040
```

```
*
                LDA       #F.TPARMX           ; "11XXXXXX" - JMP TO FILE MANAGER.
                STA       TPARMX
                LDX       #>FSC.TBL
                LDY       #<FSC.TBL
                LDA       #FSC.CNT
                JSR       MOVE.PARMS
                BCS       SCM.ERR1            ; ERR EXIT
                JMP       FMGR
*
SCM040          LDA       #D.TPARMX           ; "10XXXXXX" - JMP TO DEVICE MANAGER.
                STA       TPARMX
                LDX       #>DSC.TBL
                LDY       #<DSC.TBL
                LDA       #DSC.CNT
                JSR       MOVE.PARMS
                BCS       SCM.ERR1            ; ERR EXIT
                JMP       DMGR
*
SCM050          BVC       SCM.ERR
                PHP
                LDA       SCNUM
                AND       #$1F
                STA       SCRNUM
                PLP
                BEQ       SCM060
*
                LDA       #U.TPARMX           ; "011XXXXX" - JMP TO UTILITY MANAGER.
                STA       TPARMX
                LDX       #>USC.TBL
                LDY       #<USC.TBL
                LDA       #USC.CNT
                JSR       MOVE.PARMS
                BCS       SCM.ERR1            ; ERR EXIT
                JMP       UMGR
*
SCM060          LDA       #M.TPARMX           ; "010XXXXX" - JMP TO MEMORY MANAGER.
                STA       TPARMX
                LDX       #>MSC.TBL
                LDY       #<MSC.TBL
                LDA       #MSC.CNT
                JSR       MOVE.PARMS
                BCS       SCM.ERR1            ; ERR EXIT
                JMP       MMGR
*
SCM.ERR         LDA       #>BADSCNUM          ; ERROR, INVALID SYSTEM CALL NUMBER.
SCM.ERR1        JSR       SYSERR              ;   EXIT TO DISPATCHER ON ERROR
                PAGE
                REP       60
*
* MOVE.PARMS
*
* MOVES THE CALLER'S PARAMETERS TO THE OPERATING SYSTEM'S
* ZERO PAGE, ACCORDING TO THE SPECIFICATIONS CONTAINED
* IN THE SPECIFIED SYS CALL DFN TABLE.
*
* INPUT: (A) = MAX # ENTRIES IN PARM DFN TABLE
*        (X) = PARM DFN TBL ADR (LO)
*        (Y) =         "        (HI)
*      SCPTR = ADR OF CALLER'S SYS CALL PARMS
* ERROR: CARRY SET (SYSERR)
*
                REP       60
*
MOVE.PARMS      EQU       *
                STX       DFN.PTR             ; SAVE ADR OF DEFINITION TABLE
                STY       DFN.PTR+1
*
* IF REQ NUM > MAX REQ NUM (A REG)
*
                CMP       SCRNUM
                BCS       MOVE010
*
*   THEN ERR(BAD SYS CALL NUM)
*
                LDA       #>BADSCNUM
                BCC       SYSERR1             ;BRANCH ALWAYS TAKEN
*
* CALCULATE DEFINITION TABLE INDEX
*  AND INIT SYS CALL PARM INDEX
*
```

```
MOVE010         LDA        SCRNUM
                ASL        A
                ASL        A
                STA        DFN.PTRX
                LDA        #0
                STA        SXPAGE+DFN.PTR+1     ; AND X BYTE
                STA        SCPTRX
*
* IF SCPTR(SCPTRX)<>DFN.PTR(DFN.PTRX) THEN ERR
*
                TAY
                LDA        (SCPTR),Y
                LDY        DFN.PTRX
                CMP        (DFN.PTR),Y
                BEQ        INITLOOPCT
*
                LDA        #>BADSCPCNT          ; ERR, CALLER'S PARM COUNT INVALID
SYSERR1         JSR        SYSERR               ; EXIT
*
* INIT LOOP CTR(PARM.CNT) TO # OF PARMS IN SYS CALL
*
INITLOOPCT      STA        PARM.CNT
*
* ADVANCE PTRS
*
*
                INC        SCPTRX
                INC        DFN.PTRX
*
* MOVE REQ CODE TO SYS ZPAGE PARM LIST
*  AND ADVANCE SYS ZPAGE PTR (X=TPARMX)
*
                LDA        SCRNUM
                LDX        TPARMX
                STA        0,X
                INX
*
* INIT NIBBLE FLAG TO "RIGHT" NIBBLE
*  ZERO STATE="LEFT" NIBBLE
*
                LDA        #$FF
                STA        RGHT.NIB
                REP        60
*
*  BEGIN PARAMETER PROCESSING LOOP
*
PARMLOOP        LDA        RGHT.NIB
                EOR        #$FF                 ; COMPLEMENT NIBBLE FLAG
                STA        RGHT.NIB
*
* IF "LEFT" NIBBLE
*
                BNE        ELSE.RNIB
*
* THEN FETCH SYS CALL PARM DFN
*  AND # OF BYTES IN PARM WITHIN IT
*
                LDY        DFN.PTRX
                LDA        (DFN.PTR),Y
                STA        SCT.DFN
                AND        #$30
                LSR        A
                LSR        A
                LSR        A
                LSR        A
                STA        SCT.DCNT
                BPL        VALUE                ;BRANCH ALWAYS
*
* ELSE FETCH SYS CALL PARM DFN
*  AND # OF BYTES IN PARM WITHIN IT
*  FROM "RIGHT" NIBBLE OF DFN BYTE
*
ELSE.RNIB       LDA        SCT.DFN
                TAY
                AND        #$03
                STA        SCT.DCNT
                TYA
                ASL        A
                ASL        A
                ASL        A
                ASL        A
```

```
                STA     SCT.DFN
                INC     DFN.PTRX            ; ADVANCE SYS CALL DFN PTR
                REP     60
*
*  PARAMETER PASSED BY VALUE
*
                REP     60
VALUE           BIT     SCT.DFN
                BVS     REFERENCE
                BMI     VAL.OUT
*
*  INPUT BY VALUE
*
                LDY     SCPTRX              ; MOVE BYTES TO ZPAGE
VAL.IN          LDA     (SCPTR),Y
                STA     0,X
                INY
                INX
                DEC     SCT.DCNT
                BPL     VAL.IN
                STY     SCPTRX
                JMP     ENDLOOP1
*
*  OUTPUT BY VALUE
*
VAL.OUT         CLC                         ; BUILD PTR TO PARM ON ZPAGE
                LDA     SCPTR
                ADC     SCPTRX
                STA     0,X
                INX
                LDA     SCPTR+1
                ADC     #0
                STA     0,X
*
                CLC                         ; ADVANCE INDEX TO NEXT PARM
                LDA     SCPTRX
                ADC     SCT.DCNT
                STA     SCPTRX
*
                LDA     SXPAGE+SCPTR+1      ; INCLUDE X BYTE
                STA     SXPAGE,X
                JMP     ENDLOOP2
                REP     60
*
*  PARAMETER PASSED BY REFERENCE
*
                REP     60
REFERENCE       BPL     REF1
*
* "LIST" PTR FOUND, CHK IF "LENGTH" PARM = 0
*
                LDY     SCPTRX
                INY
                INY
                LDA     (SCPTR),Y
                BEQ     ENDLOOP0            ; "LENGTH" PARM=0, SKIP "LIST" PARM
*
REF1            LDY     SCPTRX              ; MOVE PTR TO ZPAGE
                JSR     POINTER
                BCS     PARM.ERR            ; ERROR EXIT
*
* ADVANCE SYSTEM ZPAGE POINTER (X), CALLER'S PARM PTR.
* DECREMENT PARM CTR AND CHECK IF LAST PARM PROCESSED.
*
ENDLOOP0        INX
                INC     SCPTRX
ENDLOOP2        INX
                INC     SCPTRX
ENDLOOP1        DEC     PARM.CNT
                BEQ     PARM.EXIT
                BMI     PARM.EXIT           ;SPECIAL FOR 'COLD START'
                JMP     PARMLOOP
*
*  END OF PARAMETER PROCESSING LOOP
*
                REP     60
*
PARM.EXIT       CLC                         ; NO ERRORS
PARM.ERR        RTS                         ; RETURN TO SYS CALL MANAGER
                PAGE
                REP     60
```

```
*
* POINTER
*
* INPUT:   SRC ADR   (SCPTR),Y & (SCPTR),Y+1
*          DEST ADR  (X)
*
* OUTPUT:  SCPTR     UNCHANGED
*          X REG         "
*          A,Y REGS  FLATTENED
*
* ERROR:   CARRY SET (SYSERR)
*
* POINTER.  RETRIEVES THE CALLER'S POINTER PARAMETER IN
* (SCPTR),Y, PERFORMS ADDRESS COMPENSATION, IF NECESSARY
* AND PLACES THE RESULTING POINTER AT X, X+1 AND SXPAGE+1,X.
*
                REP        60
*
POINTER         EQU        *
                LDA        (SCPTR),Y
                PHA
                INY
                LDA        (SCPTR),Y
                BEQ        INDIRECT
*
                STA        1,X                  ; DIRECT POINTER
                PLA
                STA        0,X
                LDY        #0
                BEQ        PTR010
*
INDIRECT        PLA                             ; INDIRECT POINTER
                TAY
                LDA        CZPAGE,Y
                STA        0,X
                LDA        CZPAGE+1,Y
                STA        1,X
                LDA        CXPAGE+1,Y
                TAY
*
PTR010          LDA        1,X
*
* CHECK BOUNDS OF CALLER'S POINTER PARAMETER
*
                CPY        #$8F
                BCC        PTR.X808E
                BEQ        PTR.X8F
                BCS        PTR.ERR1             ; ERROR, INVALID X BYTE
PTR.X8F         CMP        #<ADR.LOW
                BCC        PTR.ERR
                CMP        #<ADR.HIGH
                BCS        PTR.ERR
                BCC        PTR.EXIT
*
* X BYTE = 80..8E
*
PTR.X808E       CPY        #$80
                BCC        PTR.X0
                CMP        #0
                BEQ        PTR.ERR
                CMP        #$FF
                BNE        PATCH
                INY                             ; $8N:FFXX --> $8N+1:7FXX
                LDA        #$7F
                BNE        PTR.EXIT
*
* X BYTE = 0
*
PTR.X0          CPY        #0
                BNE        PTR.ERR1
                CMP        #<ADR.LOW
                BCC        PTR.ERR
                CMP        #<ADR.HIGH
                BCS        PTR.ERR
                CMP        #<ADR.MID
                BCS        PTR.EXIT
*
                PHA
                LDA        B.SAVE
                AND        #$0F
                BNE        PTR030
```

```
                PLA                      ; $B=0:2000..9FFF --> $8F:2000.9FFF
                LDY     #$8F
                BNE     PTR.EXIT
*
PTR030          ORA     #$80             ; $B<>0:2000..9FFF --> $8B:0000..7FFF
                TAY
                PLA
                SEC
                SBC     #$20
                BNE     PATCH
                DEY                      ; $8B:00XX --> $8B-1:80XX
                LDA     #$80
*
PATCH           CPY     #$80             ; KLUDGE FOR BFM:  $8N:01XX --> $8N-1:81XX
                BCC     PTR.EXIT
                CMP     #1
                BNE     PTR.EXIT
                CPY     #$80
                BEQ     PTR.ERR          ; ERROR, $80:01XX NOT ALLOWED
                DEY
                LDA     #$81
*
PTR.EXIT        STA     1,X
                TYA
                STA     SXPAGE+1,X
                CLC
                RTS
*
*
PTR.ERR         LDA     #>BADSCBNDS
                JSR     SYSERR
PTR.ERR1        LDA     #>BADXBYTE
                JSR     SYSERR
*
                LST     ON
ZZEND           EQU     *
ZZLEN           EQU     ZZEND-ZZORG
                IFNE    ZZLEN-LENSCMGR
                FAIL    2,"SOSORG        FILE IS INCORRECT FOR SCMGR"
                FIN

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

```

End of File -- Lines: 607 Characters: 15630

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

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

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

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


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

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

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

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

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

End of File -- Lines: 26 Characters: 607

```
*
                LDA        K.DRIVES             ; LINK.INIT(A=K.DRIVES DIB1..4.IN, SDT.TBL BLKDLST.IO)
                JSR        LINK.INIT
                JSR        INIT.KRNL            ; INIT.KRNL()
                JSR        WELCOME              ; WELCOME()
*
                LDA        E.REG                ; ENABLE ROM BANK
                ORA        #$03
                STA        E.REG
                LDA        ROM.ADR              ; IF MONITOR ROM <> NEW
                CMP        #ROM.ID              ;    THEN
                BEQ        LDR020
                LDX        #ERR7X               ;       ERROR("ROM ERROR:  PLEASE NOTIFY YOUR DEALER")
                LDY        #ERR7L
                JSR        ERROR
LDR020          LDA        E.REG                ; DISABLE ROM BANK
                AND        #$F6
                STA        E.REG
                REP        100
* PROCESS INTERPRETER FILE
                REP        100
*
* OPEN SOS INTERPRETER FILE (DEFAULT='SOS.INTERP')
*
                LDY        I.PATH               ; OPEN(PATHNAME:=I.PATH
LDR030          LDA        I.PATH,Y             ;      REFNUM=OPEN.REF
                STA        PATH,Y               ;      SYSBUF.P:=80:LDREND-2000 )
                DEY
                BPL        LDR030
*
                LDA        #>LDREND-$2000
                STA        SYSBUF.P
                LDA        #<LDREND-$2000
                STA        SYSBUF.P+1
                LDA        #$80
                STA        CXPAGE+SYSBUF.P+1
*
*
                BRK
                DFB        OPEN
                DW         OPEN.PARMS
                BEQ        LDR040
                LDX        #ERR1X               ; ERROR("INTERPRETER FILE NOT FOUND")
                LDY        #ERR1L
                JSR        ERROR
LDR040          LDA        OPEN.REF
                STA        READ.REF
                STA        CLOSE.REF
*
* READ IN ENTIRE INTERPRETER FILE
*
                LDA        #$80                 ; READ(REFNUM=READ.REF
                STA        CXPAGE+RDBUF.P+1     ;      RDBUF.P:=80:FILE
                LDA        #>FILE               ;      BYTES=$FFFF-FILE+1
                STA        RDBUF.P              ;      BYTESRD=I.BYTESRD )
                LDA        #<FILE
                STA        RDBUF.P+1
*
                BRK
                DFB        READ
                DW         READ.PARMS
                BEQ        LDR050
                LDX        #ERR0X               ; ERROR("I/O ERROR")
                LDY        #ERR0L
                JSR        ERROR
*                                                                  +---------------+
* CLOSE INTERPRETER FILE AND CHECK LABEL                           ! SEE FIGURE 2. !
*                                                                  +---------------+
LDR050          BRK                             ; CLOSE(REFNUM=CLOSE.REF)
                DFB        CLOSE
                DW         CLOSE.PARMS
                LDY        #7                   ; CHECK LABEL
LDR051          LDA        (RDBUF.P),Y
                CMP        I.LABEL,Y
                BNE        LDR052
                DEY
                BPL        LDR051
                BMI        LDR053
LDR052          LDX        #ERR2X               ; ERROR("INVALID INTERPRETER FILE")
                LDY        #ERR2L
                JSR        ERROR
```

```
*
* MOVE INTERPRETER CODE
*
LDR053          LDA       #>I.HDR.CNT-2         ; WORK.P:=80:I.HDR.CNT-2
                STA       WORK.P
                LDA       #<I.HDR.CNT-2
                STA       WORK.P+1
                LDA       #$80
                STA       CXPAGE+WORK.P+1
*
                JSR       ADVANCE               ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
*
                LDA       DST.P                 ; I.BASE.P:=0:DST.P
                STA       I.BASE.P
                LDA       DST.P+1
                STA       I.BASE.P+1
                LDA       #0
                STA       CXPAGE+I.BASE.P+1
*
                CLC                             ; IF DST.P+CNT > K.BASE THEN ERROR
                LDA       CNT
                ADC       DST.P
                TAX
                LDA       CNT+1
                ADC       DST.P+1
                CPX       K.BASE
                SBC       K.BASE+1
                BEQ       LDR070
                BCC       LDR070
                LDX       #ERR3X                ; ERROR("INCOMPATIBLE INTERPRETER")
                LDY       #ERR3L
                JSR       ERROR
*
LDR070          LDA       SYSBANK               ; MOVE(SRC.P=RDBUF.P DST.P A=SYSBANK CNT.IN)
                JSR       MOVE
                REP       100
* PROCESS DRIVER FILE
                REP       100
*
* OPEN SOS DRIVER FILE (DEFAULT='SOS.DRIVER')
*
                LDY       D.PATH                ; OPEN(PATHNAME:=D.PATH
LDR080          LDA       D.PATH,Y              ;      REFNUM=OPEN.REF
                STA       PATH,Y                ;      SYSBUF.P:=80:LDREND-2000 )
                DEY
                BPL       LDR080
*
                BRK
                DFB       OPEN
                DW        OPEN.PARMS
                BEQ       LDR090
                LDX       #ERR4X                ; ERROR("DRIVER FILE NOT FOUND")
                LDY       #ERR4L
                JSR       ERROR
LDR090          LDA       OPEN.REF
                STA       READ.REF
                STA       CLOSE.REF
*
* READ IN ENTIRE DRIVER FILE INTO BANK 0
*
                BRK                             ; READ(REFNUM=READ.REF
                DFB       READ                  ;      RDBUF.P:=80:FILE
                DW        READ.PARMS            ;      BYTES=$FFFF-FILE+1
*                                       ;      BYTESRD=D.BYTESRD )
                BEQ       LDR100
                LDX       #ERR0X                ; ERROR("I/O ERROR")
                LDY       #ERR0L
                JSR       ERROR
*                                                                                  +---------------+
* CLOSE THE DRIVER FILE AND CHECK LABEL                                            ! SEE FIGURE 3. !
*                                                                                  +---------------+
LDR100          BRK                             ; CLOSE(REFNUM=CLOSE.REF)
                DFB       CLOSE
                DW        CLOSE.PARMS
                LDY       #$7                   ; CHECK LABEL
LDR101          LDA       (RDBUF.P),Y
                CMP       D.LABEL,Y
                BNE       LDR102
                DEY
                BPL       LDR101
                BMI       LDR103
```

```
LDR102          LDX        #ERR5X              ; ERROR("INVALID DRIVER FILE")
                LDY        #ERR5L
                JSR        ERROR
*
* MOVE CHARACTER SET TABLE
*
LDR103          LDA        #>D.CHRSET          ; MOVE(SRC.P=D.CHRSET DST.P=$C00 A=0 CNT=$400)
                STA        SRC.P
                LDA        #<D.CHRSET
                STA        SRC.P+1
                LDA        #>$C00
                STA        DST.P
                LDA        #<$C00
                STA        DST.P+1
                LDA        #>$400
                STA        CNT
                LDA        #<$400
                STA        CNT+1
                LDA        #0
                JSR        MOVE
*
* MOVE KEYBOARD TABLE
*
                LDA        #>D.KYBD            ; MOVE(SRC.P=D.KYBD DST.P=$1700 A=0 CNT=$100.IN)
                STA        SRC.P
                LDA        #<D.KYBD
                STA        SRC.P+1
                LDA        #>$1700
                STA        DST.P
                LDA        #<$1700
                STA        DST.P+1
                LDA        #>$100
                STA        CNT
                LDA        #<$100
                STA        CNT+1
                LDA        #0
                JSR        MOVE
*
* RE-INITIALIZE SDT TABLE
*
                LDY        #>D.DRIVES-D.FILE   ; LINK.INIT(A=D.DRIVES DIB1..4.IN, SDT.TBL BLKDLST.IO)
                LDA        (RDBUF.P),Y
                JSR        LINK.INIT
*
                LDA        #0                  ; DST.P:=0:I.BASE.P/256*256
                STA        CXPAGE+DST.P+1
                STA        DST.P
                LDA        I.BASE.P+1
                STA        DST.P+1
                CMP        #$A0                ; IF DST.P>=$A000 THEN DST.P:=$A000
                BCC        LDR105
                LDA        #$A0
                STA        DST.P+1
LDR105          LDA        SYSBANK             ; DSTBANK:=SYSBANK
                STA        DSTBANK
                JSR        REVERSE             ; REVERSE(D.HDR.CNT.IN, WORK.P.OUT)
*
* RELOCATE AND MOVE DRIVERS
*
NEXTDRIVER      JSR        DADVANCE            ; "NO DRIVERS LEFT":=DADVANCE(WORK.P.IO SRC.P CNT REL.P.OUT)
                BCS        LDR140
                JSR        FLAGS               ; "INACTIVE":=FLAGS(SRC.P.IN, PG.ALIGN FIRST.ADIB.OUT)
                BVS        NEXTDRIVER
                JSR        GETMEM              ; GETMEM(PG.ALIGN CNT.IN, DST.P DSTBANK DSEGLIST.IO, PREVBANK.OUT)
                JSR        RELOC               ; RELOC(SRC.P REL.P DST.P.IN)
*
                LDA        DSTBANK             ; IF DSTBANK < 0 OR DST.P < SRC.P THEN ERROR
                BMI        LDR120
                LDA        CXPAGE+SRC.P+1      ;    (CONVERT SRC.P TO BANK SWITCHED ADDRESS)
                AND        #$7F
                STA        TEMP.BANK
                LDA        SRC.P+1
                BPL        LDR110
                INC        TEMP.BANK
LDR110          AND        #$7F
                CLC
                ADC        #<$2000
                STA        TEMP.ADRH
                LDA        DST.P               ;    (NOW COMPARE)
                CMP        SRC.P
                LDA        DST.P+1
```

```
                SBC     TEMP.ADRH
                LDA     DSTBANK
                SBC     TEMP.BANK
                BCS     LDR130
LDR120          LDX     #ERR6X              ;    ERROR("DRIVER FILE TOO LARGE")
                LDY     #ERR6L
                JSR     ERROR
*
LDR130          LDA     DSTBANK             ; MOVE(SRC.P DST.P A=DSTBANK CNT.IN)
                JSR     MOVE
                JSR     LINK                ; LINK(DST.P DSTBANK PREVBANK FIRST.ADIB.IN, SDT.TBL BLKDLST.IO)
                JMP     NEXTDRIVER
                REP     100
* SETUP USER ENVIRONMENT
                REP     100
*
* RE-INITIALIZE KERNEL/DRIVERS, ALLOCATE SYSTEM SEGMENTS
*
LDR140          JSR     INIT.KRNL           ; INIT.KRNL()
                JSR     ALLOC.SEG           ; ALLOC.SEG(K.BASE I.BASE.P SYSBANK.IN)
                JSR     ALLOC.DSEG          ; ALLOC.DSEG(DSEGLIST.IN)
*
* SET PREFIX TO THE BOOT VOLUME
*
                LDA     #0                  ; TURN VIDEO OFF - PREVENTS CHAR "GROWTH" DURING DOWNLOAD
                STA     SCRNMODE
                BRK                         ; SET.PREFIX(PREFIXPATH=".D1")
                DFB     SETPREFIX
                DW      PREFX.PARMS
*
* LAUNCH CHARACTER SET DOWNLOAD (CONSOLE) AND CLEAR SCREEN
*
                CLI                         ; BEGIN CHARACTER SET DOWNLOAD (CONSOLE)
*
                LDA     #0                  ; CLEAR TEXT SCREENS
                STA     CXPAGE+SRC.P+1
                STA     CXPAGE+DST.P+1
                LDA     #$04
                STA     SRC.P+1
                STA     DST.P+1
                LDA     #$00
                STA     SRC.P
                LDA     #$80
                STA     DST.P
                LDA     #$A0
                LDX     #8
CLEAR0          LDY     #$77
CLEAR1          STA     (SRC.P),Y
                STA     (DST.P),Y
                DEY
                BPL     CLEAR1
                INC     SRC.P+1             ; NEXT PAGE
                INC     DST.P+1             ; NEXT PAGE
                DEX
                BNE     CLEAR0
*
WAIT            INC     SRC.P               ; WAIT FOR DOWNLOAD TO COMPLETE
                BNE     WAIT
                INX
                BNE     WAIT
*
                LDA     #$80                ; TURN VIDEO ON
                STA     SCRNMODE
                RTS
                REP     100

                CHN     SOSLDR.E.SRC

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


End of File -- Lines: 635 Characters: 19090
```

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

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

                PAGE
                REP        100
*
* SET.DRIVES ( IN:   A=# DRIVES
*              IN:   DIB1..4    )
* (INITIALIZES DIB LINKS IN KERNEL'S FLOPPY DRIVER)
                REP        100
*
SET.DRIVES      EQU        *
                TAY                             ; SAVE # OF DRIVES
                LDA        #>DIB2               ; DIB1:=ADR(DIB2)
                STA        DIB1
                LDA        #<DIB2
                STA        DIB1+1
                LDA        #>DIB3               ; DIB2:=ADR(DIB3)
                STA        DIB2
                LDA        #<DIB3
                STA        DIB2+1
                LDA        #>DIB4               ; DIB3:=ADR(DIB4)
                STA        DIB3
                LDA        #<DIB4
                STA        DIB3+1
*
                LDA        #0                   ; CASE (Y=# OF DRIVES)
                CPY        #2
                BCC        STDR010
                BEQ        STDR020
                CPY        #4
                BCC        STDR030
                BCS        STDR040
*
STDR010         STA        DIB1                 ;    1:  DIB1:=0
                STA        DIB1+1
                RTS
*
STDR020         STA        DIB2                 ;    2:  DIB2:=0
                STA        DIB2+1
                RTS
*
STDR030         STA        DIB3                 ;    3:  DIB3:=0
                STA        DIB3+1
                RTS
*
STDR040         STA        DIB4                 ;    4:  DIB4:=0
                STA        DIB4+1
                RTS                             ; RETURN
                PAGE
                REP        100
*
* INIT.KRNL ()
*
* (CALLS KERNEL INITIALIZATION MODULES)
                REP        100
*
INIT.KRNL       EQU        *
                LDA        E.REG                ; SWITCH IN I/O BANK AND SELECT PRIMARY STACK
                ORA        #$44                 ; E:=( 0.1.1.X:0.1.0.0 )
                STA        E.REG                ;    ( 1.I.S.R:W.P.R.R )
*
                LDA        #<SZPAGE             ; SWITCH TO SOS ZPAGE
                STA        Z.REG
*
                JSR        INT.INIT             ; CALL KERNEL INITIALIZATION ROUTINES
                JSR        EVQ.INIT
                JSR        BFM.INIT2
                BCS        INITK.ERR
                JSR        DMGR.INIT
                JSR        CFMGR.INIT
                JSR        MMGR.INIT
                JSR        BMGR.INIT
                JSR        BFM.INIT
```

```
                 JSR        CLK.INIT
*
                 LDA        E.REG                 ; SWITCH OUT I/O BANK AND RETURN TO ALTERNATE STACK
                 AND        #$BB                  ; E:=( 0.0.1.X:0.0.0.0 )
                 STA        E.REG                 ;    ( 1.I.S.R:W.P.R.R )
*
                 LDA        #<CZPAGE              ; SWITCH BACK TO USER ZPAGE
                 STA        Z.REG
*
                 RTS                              ; RETURN
*
*
INITK.ERR        LDX        #ERR0X                ; ERROR("I/O ERROR")
                 LDY        #ERR0L
                 JMP        ERROR
                 PAGE
                 REP        100
*
* ADVANCE ( I/O:  WORK.P
*           OUT:  SRC.P
*           OUT:  DST.P
*           OUT:  CNT     )
* (ADVANCES WORK.P TO NEXT INTERP.KERNEL MODULE.  INITS SRC.P, DST.P, CNT FOR MOVE)
                 REP        100
*
ADVANCE          EQU        *
                 CLC
                 LDY        #2                    ; Y:=0
                 LDA        WORK.P                ; WORK.P:=WORK.P+(WORK.P),Y + 4
                 ADC        (WORK.P),Y
                 TAX
                 INY
                 LDA        WORK.P+1
                 ADC        (WORK.P),Y
                 PHA
                 TXA
                 ADC        #4
                 STA        WORK.P
                 PLA
                 ADC        #0
                 STA        WORK.P+1
                 CLC                              ; SRC.P:=X:WORK.P+4
                 LDA        WORK.P
                 ADC        #>$0004
                 STA        SRC.P
                 LDA        WORK.P+1
                 ADC        #<$0004
                 STA        SRC.P+1
                 LDA        CXPAGE+WORK.P+1
                 STA        CXPAGE+SRC.P+1
                 LDY        #0                    ; DST.P:=0:(WORK.P)
                 STY        CXPAGE+DST.P+1
                 LDA        (WORK.P),Y
                 STA        DST.P
                 INY
                 LDA        (WORK.P),Y
                 STA        DST.P+1
                 INY                              ; Y:=2
                 LDA        (WORK.P),Y            ; CNT:=(WORK.P),Y
                 STA        CNT
                 INY
                 LDA        (WORK.P),Y
                 STA        CNT+1
                 RTS                              ; RETURN
                 PAGE
                 REP        100
*
* REVERSE ( IN:   D.HDR.CNT
*           IN:   SDT.SIZE = CONSTANT
*           I/O:  DRIVER FILE,
*           OUT:  WORK.P      )      )
*
*            LOCAL:  REV.SAVE, REV.TEMP
* (REVERSES TITLE/CODE/RELOC COUNTS TO ALLOW DRIVER FILE TO BE PROCESSED FROM BACK TO FRONT)
                 REP        100
REVERSE          EQU        *
                 LDA        #>D.HDR.CNT           ; WORK.P:=80:D.HDR.CNT
                 STA        WORK.P
                 LDA        #<D.HDR.CNT
                 STA        WORK.P+1
                 LDA        #$80
```

```
                 STA        CXPAGE+WORK.P+1
                 CLC                             ; WORK.P:=WORK.P+(WORK.P)+2
                 LDY        #0
                 LDA        WORK.P
                 ADC        (WORK.P),Y
                 TAX
                 INY
                 LDA        WORK.P+1
                 ADC        (WORK.P),Y
                 PHA
                 TXA
                 ADC        #2
                 STA        WORK.P
                 PLA
                 ADC        #0
                 STA        WORK.P+1
                 LDA        (WORK.P),Y           ; IF (WORK.P)=$FFFF
                 DEY
                 AND        (WORK.P),Y           ;    THEN
                 CMP        #$FF
                 BNE        REV010
                 LDX        #ERR10X              ;       ERROR("EMPTY DRIVER FILE")
                 LDY        #ERR10L
                 JSR        ERROR
REV010           LDA        #$FF
                 STA        REV.SAVE
                 STA        REV.SAVE+1
*
REV020           LDA        REV.SAVE             ;R1: STACK:=REV.SAVE
                 PHA
                 LDA        REV.SAVE+1
                 PHA
                 LDY        #0                   ;    REV.SAVE:=(WORK.P)
                 LDA        (WORK.P),Y
                 STA        REV.SAVE
                 INY
                 LDA        (WORK.P),Y
                 STA        REV.SAVE+1
                 PLA                             ;    (WORK.P):=STACK
                 STA        (WORK.P),Y
                 DEY
                 PLA
                 STA        (WORK.P),Y
                 LDA        REV.SAVE             ;    IF REV.SAVE = $FFFF THEN EXIT
                 AND        REV.SAVE+1
                 CMP        #$FF
                 BEQ        REV.EXIT
REV030           BIT        REV.SAVE+1           ;    IF REV.SAVE >= $8000 THEN ERROR
                 BMI        REV040
                 CLC                             ;    WORK.P:=WORK.P+REV.SAVE+2
                 LDA        WORK.P
                 ADC        REV.SAVE
                 TAX
                 LDA        WORK.P+1
                 ADC        REV.SAVE+1
                 PHA
                 BCS        REV040
                 TXA
                 ADC        #2
                 STA        WORK.P
                 PLA
                 ADC        #0
                 STA        WORK.P+1
                 BCC        REV020               ;    IF C=FALSE THEN R1
REV040           LDX        #ERR5X               ;            ELSE ERROR("INVALID DRIVER FILE")
                 LDY        #ERR5L
                 JSR        ERROR
*
REV.EXIT         RTS                             ; RETURN
                 PAGE
                 REP        100
*
* DADVANCE ( I/O:  WORK.P
*            OUT:  C="NO DRIVERS LEFT"
*            OUT:  SRC.P
*            OUT:  CNT
*            OUT:  REL.P )
* (ADVANCES WORK.P TO NEXT DRIVER MODULE.  INITS SRC.P, CNT, REL.P FOR RELOCATION AND MOVE)
                 REP        100
DADVANCE         EQU        *
                 LDY        #0                   ; IF (WORK.P)=$FFFF THEN EXIT "NO DRIVERS LEFT IN FILE"
```

```
                LDA     (WORK.P),Y
                INY
                AND     (WORK.P),Y
                CMP     #$FF
                BNE     DADV010
                SEC                     ; C:="NO DRIVERS LEFT"
                RTS                     ; RETURN
*
*
DADV010         LDA     WORK.P          ; REL.P:=X:WORK.P
                STA     REL.P
                LDA     WORK.P+1
                STA     REL.P+1
                LDA     CXPAGE+WORK.P+1
                STA     CXPAGE+REL.P+1
*
                JSR     DADD            ; ADVANCE TO CODE COUNT FIELD
*
                LDY     #0              ; CNT:=(WORK.P)
                LDA     (WORK.P),Y
                STA     CNT
                INY
                LDA     (WORK.P),Y
                STA     CNT+1
*
                JSR     DADD            ; ADVANCE TO TITLE CNT FIELD
*
                CLC                     ; SRC.P:=X:WORK.P+2
                LDA     WORK.P
                ADC     #2
                STA     SRC.P
                LDA     WORK.P+1
                ADC     #0
                STA     SRC.P+1
                LDA     CXPAGE+WORK.P+1
                STA     CXPAGE+SRC.P+1
*
                JSR     DADD            ; ADVANCE TO RELOC FIELD OF NEXT DRIVER
                CLC                     ; C:="DRIVERS LEFT"
                RTS                     ; RETURN
                PAGE
                REP     100
*
* DADD ( I/O:   WORK.P )
*
* (ADVANCES WORK.P TO NEXT FIELD IN DRIVER MODULE)
                REP     100
DADD            EQU     *
                SEC                     ; WORK.P:=WORK.P-(WORK.P)-2
                LDY     #0
                LDA     WORK.P
                SBC     (WORK.P),Y
                TAX
                INY
                LDA     WORK.P+1
                SBC     (WORK.P),Y
                PHA
                TXA
                SBC     #2
                STA     WORK.P
                PLA
                SBC     #0
                STA     WORK.P+1
                RTS                     ; RETURN
                PAGE
                REP     100
*
* FLAGS ( IN:   SRC.P
*         OUT:  PG.ALIGN
*         OUT:  FIRST.ADIB
*         OUT:  OV="ALL DIBS INACTIVE" )
*
*         LOCAL:  PREV.ADIB.P, DIB.P
* (PROCESSES "INACTIVE" & "PAGE ALIGN" FLAGS IN DRIVER MODULE'S DIBS"
                REP     100
FLAGS           EQU     *
                SEC                     ; C="FIRST DIB"
FLAG010         JSR     NEXT.DIB        ; NEXT.DIB(SRC.P.IN, DIB.P PG.ALIGN C OV.OUT)
                BVC     FLAG015         ; IF OV <> "INACTIVE" THEN ACTIVE DIB FOUND
                BCC     FLAG010         ; IF C <> "LAST DIB" THEN CHECK NEXT DIB
                RTS                     ; RETURN  (OV:="ALL DIBS INACTIVE")
```

```
*
FLAG015         PHP                                ; PUSH STATUS
                SEC                                ; FIRST.ADIB:=DIB.P-SRC.P
                LDA        DIB.P
                SBC        SRC.P
                STA        FIRST.ADIB
                LDA        DIB.P+1
                SBC        SRC.P+1
                STA        FIRST.ADIB+1
                LDA        DIB.P                   ; PREV.ADIB.P:=X:DIB.P
                STA        PREV.ADIB.P
                LDA        DIB.P+1
                STA        PREV.ADIB.P+1
                LDA        CXPAGE+DIB.P+1
                STA        CXPAGE+PREV.ADIB.P+1
                PLP                                ; PULL STATUS
                BCS        FLAG100                 ; IF C="LAST DIB" THEN EXIT
*
FLAG020         JSR        NEXT.DIB                ; NEXT.DIB(SRC.P.IN, DIB.P PG.ALIGN C OV.OUT)
                PHP                                ; PUSH STATUS
                LDY        #0                      ; IF OV="INACTIVE DIB"
                BVC        FLAG025
                SEC                                ;    THEN
                LDA        PREV.ADIB.P             ;       (PREV.ADIB.P):=PREV.ADIB.P-SRC.P
                SBC        SRC.P
                STA        (PREV.ADIB.P),Y
                INY
                LDA        PREV.ADIB.P+1
                SBC        SRC.P+1
                STA        (PREV.ADIB.P),Y
                JMP        FLAG050
*
FLAG025         SEC                                ;    ELSE
                LDA        DIB.P                   ;       (PREV.ADIB.P):=DIB.P-SRC.P
                SBC        SRC.P
                STA        (PREV.ADIB.P),Y
                INY
                LDA        DIB.P+1
                TAX
                SBC        SRC.P+1
                STA        (PREV.ADIB.P),Y
                STX        PREV.ADIB.P+1           ;       PREV.ADIB.P:=DIB.P
                LDA        DIB.P
                STA        PREV.ADIB.P
FLAG050         PLP                                ; PULL STATUS
                BCC        FLAG020                 ; IF C <> "LAST DIB" THEN PROCESS NEXT DIB
*
FLAG100         CLV                                ; OV:="ACTIVE DIBS"
                RTS                                ; RETURN
                PAGE
                REP        100
*
* NEXT.DIB ( IN:   C="FIRST DIB"
*            IN:   SRC.P
*            OUT:  DIB.P
*            OUT:  PG.ALIGN
*            OUT:  C="LAST DIB"
*            OUT:  OV="INACTIVE DIB" )
*
*            LOCAL:  DIB.FLAGS, DIB.DCB = CONSTANT
* (ADVANCES TO NEXT DIB IN DRIVER MODULE)
                REP        100
NEXT.DIB        EQU        *
                LDY        #0
                BCC        NXTD010                 ; IF C = "FIRST DIB"
                STY        PG.ALIGN                ;    THEN
                STY        PG.ALIGN+1              ;       PG.ALIGN:=0
                LDA        SRC.P                   ;       DIB.P:=X:SRC.P
                STA        DIB.P
                LDA        SRC.P+1
                STA        DIB.P+1
                LDA        CXPAGE+SRC.P+1
                STA        CXPAGE+DIB.P+1
                JMP        NXTD020
NXTD010         LDA        SRC.P                   ;    ELSE
                ADC        (DIB.P),Y               ;       DIB.P:=SRC.P+(DIB.P)
                TAX
                INY
                LDA        SRC.P+1
                ADC        (DIB.P),Y
                STA        DIB.P+1
```

```
                STX       DIB.P
*
NXTD020         LDY       #DIB.FLAGS           ; IF (DIB.P),DIB.FLAGS.BIT7 = "INACTIVE"
                LDA       (DIB.P),Y
                BMI       NXTD030
                BIT       NXTD999              ;    THEN
                BVS       NXTD040              ;       OV:="INACTIVE"
*                                           ELSE
NXTD030         AND       #$40                 ;       IF (DIB.P),DIB.FLAGS.BIT6 = "PAGE ALIGN"
                BEQ       NXTD040
                CLC                            ;          THEN
                LDA       #DIB.DCB+2           ;             PAGE.ALIGN:=DIB.DCB+2+(SRC.P),DIB.DCB
                TAY
                DEY
                DEY
                ADC       (SRC.P),Y
                STA       PG.ALIGN
                INY
                LDA       #0
                ADC       (SRC.P),Y
                STA       PG.ALIGN+1
                CLV                            ;       OV:="ACTIVE"
*
NXTD040         LDY       #0                   ; IF (DIB.P) = 0
                LDA       (DIB.P),Y
                INY
                ORA       (DIB.P),Y
                BNE       NXTD998
                SEC                            ;    THEN  C:="LAST DIB"
                BCS       NXTD999
NXTD998         CLC                            ;    ELSE  C:=NOT "LAST DIB"
NXTD999         RTS                            ; RETURN
                REP       100

                CHN       SOSLDR.F.SRC

                RTS                            ; RETURN

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

```

End of File -- Lines: 442 Characters: 12097

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

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

                PAGE
                REP        100
*
* GETMEM ( IN:   PG.ALIGN
*          IN:   CNT
*          I/O:  DST.P
*          I/O:  DSTBANK
*          I/O:  DSEGLIST
*          OUT:  PREVBANK )
*
*          LOCAL:  PREVDST
* (COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT)
                REP        100
GETMEM          EQU        *
                LDA        DSTBANK             ; PREVBANK:=DSTBANK
                STA        PREVBANK
                LDA        DST.P               ; PREVDST:=DST.P
                STA        PREVDST
                LDA        DST.P+1
                STA        PREVDST+1
                JSR        NEWDST              ; NEWDST(PG.ALIGN.IN, PREVDST.IN, CNT.IN, DST.P.OUT)
*
                LDA        DST.P+1             ; IF DST.P >= $2000
                CMP        #$20
                BCC        GETM010
                SEC                            ;    THEN
                LDA        PREVDST+1           ;       A=PAGES:=PREVDST-DST.P
                SBC        DST.P+1
                CLC
                JSR        BUILD.DSEG          ;       BUILD.DSEG(C="NEXT BANK".IN, A=PAGES.IN, DSEGLIST.IO)
                JMP        GETM.EXIT
*                                            ELSE
GETM010         DEC        DSTBANK             ;       DSTBANK:=DSTBANK-1
                LDA        #>$A000             ;       PREVDST:=$A000
                STA        PREVDST
                LDA        #<$A000
                STA        PREVDST+1
                JSR        NEWDST              ;       NEWDST(PG.ALIGN.IN, PREVDST.IN, CNT.IN, DST.P.OUT)
                SEC                            ;       A="PAGES":=PREVDST-DST.P
                LDA        PREVDST+1
                SBC        DST.P+1
                SEC
                JSR        BUILD.DSEG          ;       BUILD.DSEG(C="NEXTBANK".IN, A="PAGES".IN, DSEGLIST.IO)
*
GETM.EXIT       RTS                            ; RETURN
                PAGE
                REP        100
*
* NEWDST ( IN:   PG.ALIGN
*          IN:   PREVDST
*          IN:   CNT
*          I/O:  DST.P      )
* (COMPUTES DESTINATION BASE ADDRESS, ALIGNING ON PAGE BOUNDARY IF REQUESTED)
                REP        100
NEWDST          EQU        *
                SEC                            ; IF (PREVDST-$2000) < CNT
                LDA        PREVDST
                SBC        #>$2000
                TAX
                LDA        PREVDST+1
                SBC        #<$2000
                CPX        CNT
                SBC        CNT+1
                BCS        NEWD010
                LDA        #0                  ;    THEN
                STA        DST.P               ;      DST.P:=0
                STA        DST.P+1
                BEQ        NEWD.EXIT
NEWD010         SEC                            ;    ELSE
                LDA        PREVDST             ;       DST.P:=PREVDST-CNT
                SBC        CNT
```

```
* MARK ALL FILES BELONGING TO A VOLUME
* AS SWAPPED-IN OR SWAPPED-OUT.
*
* INPUT ARGS: DEVNUM -- DEVICE NUMBER OF MOUNTED VOLUME
*             A REGISTER - SWAP BYTE
*             CARRY -- CARRY FLAG SET MEANS SWAP OUT; ELSE SWAP IN
*
* OUTPUT ARGS: NONE
* GLOBALS AFFECTED: FCB, FCBPTR
* REGISTER STATUS: SCRAMBLED
*
FCBSCAN         EQU       *                    ; MARK FILES BELONGING TO VOLUME AS SWAPPED OR UNSWAPPED
*
                TAX                            ; SAVE SWAP BYTE
                LDY       FCBADDRH             ; POINT TO
                STY       FCBPTR+1             ; BEGINNING TO FCB
                LDY       #0
                STY       FCBPTR
                BCS       FCBOUT               ; SWAP OUT A VOLUMES FILES
FCBIN           EQU       *                    ; SWAPIN A VOLUMES FILES
                JSR       FCBFETCH             ; GET NEXT ACTIVE FCB CANDIDATE
                BCS       FCBRTS               ; NO MORE FILES TO PROCESS
                LDY       #FCBSWAP
                TXA
                CMP       (FCBPTR),Y           ; SWAP BYTES MATCH?
                BNE       FCBIN1               ; BRANCH IF NOT
                LDA       #0
                STA       (FCBPTR),Y           ; MARK FILE AS SWAPPED IN
FCBIN1          JSR       NEXTFCB              ; ADVANCE FCB POINTER
                BCS       FCBRTS               ; NO MORE TO LOOK AT
                JMP       FCBIN                ; AND LOOK AT NEXT FILE
*
FCBOUT          EQU       *                    ; SWAPPED OUT A VOLUMES FILES
                JSR       FCBFETCH             ; GET NEXT ACTIVE FILE IN FCB
                BCS       FCBRTS               ; NO MORE FILES -- RETURN TO USER
                LDY       #FCBSWAP             ; COMPARE
                LDA       (FCBPTR),Y
                BNE       FCBOUT1              ; ALREADY SWAPPED OUT
                TXA
                STA       (FCBPTR),Y           ; MARK AS SWAPPED
FCBOUT1         JSR       NEXTFCB              ; ADVANCE FCB POINTER
                BCS       FCBRTS
                JMP       FCBOUT               ; SWAP OUT NEXT FILE
*
FCBRTS          RTS
FCBFETCH        EQU       *                    ; GET NEXT ACTIVE FILE FROM FCB
* X REGISTER MUST NOT BE DISTURBED
* USES FCBPTR
                LDY       #FCBDEVN             ; MAKE
                LDA       (FCBPTR),Y           ; SURE DEVICE
                CMP       DEVNUM               ; MATCHES
                BNE       NEXTFCB
                LDY       #FCBREFN             ; MAKE SURE FILE IS ACTIVE
                LDA       (FCBPTR),Y
                BEQ       NEXTFCB              ; BRANCH IF NOT
                CLC
                RTS                            ; RETURN WITH CARRY CLEAR SHOWING AN ACTIVE FILE
NEXTFCB         LDA       FCBPTR
                CLC
                ADC       #$20                 ; FCB ENTRY SIZE
                STA       FCBPTR
                BCC       FCBFETCH             ; BRANCH IF NO PAGE CROSS
                LDA       FCBPTR+1
                INC       FCBPTR+1             ; SECOND PAGE
                CMP       FCBADDRH
                BEQ       FCBFETCH             ; LOOK AT PAGE TWO
NEXTEND         SEC
                RTS                            ; SHOW NO MORE FILES TO LOOK AT
USRREQ          EQU       *                    ; OPERATOR CONSOLE MESSAGE INTERFACE
* PRODUCES A MESSAGE REQUESTING
* THE SYSTEM OPERATOR TO MOUNT THE VOLUME
* SPECIFIED BY "VCBPTR" ON DEVICE SPECIFIED
* BY DEVNUM.  THIS MODULE INSISTS
* UPON THE CORRECT OPERATOR ACTION
* UPON THREE FAILURES TO COMPLY,
* THE MODULE WILL SIGNIFY FAILURE WITH
* CARRY SET.  IF THE CORRECT ACTION IS TAKEN,
* CARRY WILL BE RETURNED CLEAR
*
* INPUT ARGS: VOLUME NAME (VCBPTR)
*             DEVICE NUMBER (DEVNUM)
```

```
*
* OUTPUT ARGS: CC = OPERATOR COMPLIED WITH REQUESTED ACTION
*              CS = OPERATOR COULDN'T/DIDN'T COMPLY
*
* GLOBALS AFFECTED: NONE
*
* STATUS OF REGISTERS: UNCERTAIN
*
VNML            EQU       ZPGTEMP              ; VOLUME NAME LENGTH
                LDY       #VCBNML              ; IF ILLEGAL VCB
                LDA       (VCBPTR),Y           ; GET OUT QUICK
                BEQ       NEXTEND              ; BRANCH TO SEC RTS
                LDX       #$E                  ; LENGTH OF NAMED AREA-1
                LDA       #$0                  ; NULLS
UR1             STA       MDEV,X               ; BOTH CLEAR
                STA       MVOL,X               ; IN ONE LOOP
                DEX
                BPL       UR1
*
* DO A D-INFO TO FETCH THE DEVICE NAME
*
                LDA       #5                   ; DO ALL
                STA       $C0                  ; NECESSARY
                LDA       DEVNUM               ; HOUSKEEPING
                STA       $C1                  ; TO SET UP
                LDA       #>MDEV-1             ; A DEVICE MANAGER CALL
                STA       $C2
                LDA       #<MDEV-1
                STA       $C3
                LDA       #$8F                 ; EXTEND BYTE
                STA       $14C3
                LDA       #0
                STA       $14C2
                STA       $C4
                STA       $C5
                STA       $C6                  ; ZERO SUPERFLUOUS PARMS
                STA       URDERR               ; RESET FAILURE COUNT
                JSR       RPEATIO0             ; GET INFO FROM BOBS CODE
                LDA       #$20                 ; "SPACE" RESTORED
                STA       MDEV-1               ; RESTORED
                LDY       #VCBNML
                LDA       (VCBPTR),Y           ; LENGTH OF VOLUME NAME
                STA       VNML                 ; SAVED FOR WORK
                LDA       #0
                TAX
                LDY       #VCBNAM              ; POINT TO BEGINNING OF VOLUME NAME
UR2             LDA       (VCBPTR),Y
                STA       MVOL,X
                INX
                INY                            ; VOLUME NAME MOVED
                DEC       VNML                 ; TO MESSAGE BUFFER
                BNE       UR2                  ; CHARACTER BY CHARACTER
URDU            LDX       #>UMB                ; PASS THE AREA'S ADDR
                LDY       #<UMB                ; IN X AND Y REGS,LOW, HIGH)
                JSR       OPMSGRPLY            ; HAVE MESSAGE SYSTEM PRINT IT
                JSR       VERFYVOL             ; DID THE USER COMPLY?
                BCS       URDU1                ; BRANCH IF NOT
                RTS                            ; EXIT--CARRY IS CLEAR
URDU1           INC       URDERR               ; COLLECT USER ERRORS
                LDA       URDERR
                CMP       #3                   ; ONLY THREE TRIES ALLOWED
                BCC       URDU                 ; RETRY MESSAGE IF LESS THAN THREE TRIES
                RTS                            ; OTHERWISE RETURN WITH CARRY SET
*
*
*
*
*
* CLOSE UNCONDITIONAL
*
* (USER HAS REPLIED 'N' TO A VOLUME MOUNT REQUEST
* CLOSE ALL FILES ON VOLUME/UNLOG VOLUME
*
* INPUT ARGUMENT: (VCBPTR)
* OUTPUT ARGUMENT: NONE
*
CLOSEU          EQU       *
VSWA            EQU       ZPGTEMP              ; THE 'SWAP BYTE' STORED HERE
                LDY       #VCBDEV              ; FETCH
                LDA       (VCBPTR),Y           ; THE DEVICE NUMBER
                STA       DEVNUM               ; OF THIS VOLUME & SAVE IT
```

```
                LDY     #VCBSWAP            ; FETCH THE
                LDA     (VCBPTR),Y          ; SWAP BYTE
                STA     VSWA                ; SAVE FOR REFERENCE, TOO
                LDA     #0
                LDY     #VCBNML             ; UNLOG THE VOLUME
                STA     (VCBPTR),Y          ; BY SETTING LEN OF VOL NAME TO ZERO
                LDY     #VCBSWAP
                STA     (VCBPTR),Y          ; TURN OFF SWAP FLAG
                LDY     FCBADDRH            ; SET UP FCB SCAN FROM BEGINNING OF FCB
                STY     FCBPTR+1
                LDY     #0
                STY     FCBPTR
VFCBLOP         LDY     #FCBDEVN            ; FETCH
                LDA     (FCBPTR),Y          ; THE DEVICE
                CMP     DEVNUM              ; NUMBER AND SEE IF A MATCH
                BNE     VFCBNXT             ; BRANCH IF NO MATCH
                LDY     #FCBREFN            ; SEE EVEN IF FILE OPEN
                LDA     (FCBPTR),Y
                BEQ     VFCBNXT             ; BRANCH IF NOT
                LDY     #FCBSWAP            ; CHECK TO SEE IF ATTACHED
                LDA     (FCBPTR),Y          ; TO SAME VOLUME
                CMP     VSWA
                BNE     VFCBNXT             ; BRANCH IF NOT
                LDY     #FCBBUFN            ; RELEASE
                LDA     (FCBPTR),Y          ; ANY
                JSR     RELBUF              ; BUFFERS ASSOCIATED
                LDY     #FCBSWAP            ; AND CLEAR
                LDA     #0                  ; THE SWAP BYTE
                STA     (FCBPTR),Y
                LDY     #FCBREFN            ; AND FINALLY
                STA     (FCBPTR),Y          ; SAY 'CLOSED'
VFCBNXT         LDA     FCBPTR
                CLC
                ADC     #$20                ; FCB ENTRY SIZE
                STA     FCBPTR
                BCC     VFCBLOP
                LDA     FCBPTR+1
                INC     FCBPTR+1            ; LOOK AT SECOND PAGE
                CMP     FCBADDRH
                BEQ     VFCBLOP             ; CHECK PAGE TWO OF FCB
                RTS                         ; RETURN TO USER W/O ERROR
*
FCBUSED         EQU     *                   ; MARK AS FCB AS DIRTY SO
* THE DIRECTORY WILL BE FLUSHED ON 'FLUSH'
                STY     ZPGTEMP
                PHA                         ; SAVE REGS
                LDY     #FCBDIRTY
                LDA     (FCBPTR),Y          ; FETCH CURRENT FCBDIRTY BYTE
                ORA     #FCBMOD             ; MARK FCB AS DIRTY
                STA     (FCBPTR),Y          ; SAVE IT BACK
                PLA
                LDY     ZPGTEMP             ; AND RESTORE REGS
                RTS
*
URDERR          DS      1                   ; ERROR COUNT FOR USRREQ
*
*
UMB             EQU     *
                DFB     $49,$6E,$73,$65,$72,$74,$20
                DFB     $76,$6F,$6C,$75,$6D,$65
                DFB     $3A,$20             ; "INSERT VOLUME: "
MVOL            DS      15
                DFB     $0D                 ; CR LINE TERMINATOR
                DFB     $20,$20,$20,$20,$69,$6E,$20
                DFB     $64,$65,$76,$69,$63,$65
                DFB     $3A,$20             ; "    IN DEVICE: "
MDEV            DS      15
                DFB     $0D                 ; CR LINE TERMINATOR
                DFB     $74,$68,$65,$6E,$20,$70,$72
                DFB     $65,$73,$73,$20,$74,$68,$65,$20
                DFB     $41,$4C,$50,$48,$41,$20,$4C
                DFB     $4F,$43,$4B,$20,$6B,$65,$79
                DFB     $20,$74,$77,$69,$63,$65
* "THEN PRESS THE ALPHA LOCK KEY TWICE"
* FOLLOWED WITH $FF MESSAGE TERMINATOR (HIGH BIT SIGNIFICANT)
                DFB     $FF                 ; MESSAGE TERMINATOR (HIGH BIT)
*
ZZLEN           EQU     *-ZZORG
ZZEND           EQU     *
                IFNE    ZZLEN-LENBFM
                FAIL    2,"SOSORG           FILE IS INCORRECT FORMBFM"
```

```
                FIN

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

```

End of File -- Lines: 404 Characters: 15080

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

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

                SBTL       "SOS 1.1 SYSTEM ERROR ROUTINES"
                REL
                INCLUDE    SOSORG,6,1,254
                ORG        ORGSERR
ZZORG           EQU        *
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
* SYSTEM ERROR ROUTINES  (VERSION = 1.1O   )
*                        (DATE    = 12/02/81)
*
* THIS MODULE CONTAINS THE SYSTEM ERROR AND SYSTEM FAILURE ROUTINES.
*
                REP        60
*
                ENTRY      SYSERR
                ENTRY      SYSDEATH
*
                EXTRN      SERR
                EXTRN      SDEATH.REGS
                EXTRN      SCRNMODE
                PAGE
                REP        60
*
* DATA DECLARATIONS
*
                REP        60
*
E.REG           EQU        $FFDF
Z.REG           EQU        $FFD0
B.REG           EQU        $FFEF
*
S.SAVE          EQU        $09                  ; REGISTER SAVE AREA
PCH.SAVE        EQU        $08
PCL.SAVE        EQU        $07
P.SAVE          EQU        $06
A.SAVE          EQU        $05
X.SAVE          EQU        $04
Y.SAVE          EQU        $03
E.SAVE          EQU        $02
Z.SAVE          EQU        $01
B.SAVE          EQU        $00
*
NMI.VECTOR      EQU        $FFFA
*
TXT.CLR         EQU        $C050
MIX.CLR         EQU        $C052
HIRES.CLR       EQU        $C056
*
PG2.CLR         EQU        $C054
*
MSGBASE         EQU        $7E4
MSGBASE2        EQU        $BE4
MSG             ASC        ' SYSTEM FAILURE = $'
MSGLEN          EQU        *-MSG
                PAGE
                REP        60
*
* SYSTEM ERROR ROUTINE
*
* THIS ROUTINE IS CALLED WHEN AN ERROR CONDITION HAS BEEN
* ENCOUNTERED.  THE ERROR NUMBER IS PASSED IN THE A REG
* AND THE CALL TO THIS ROUTINE MUST ALWAYS BE A JSR.
*
                REP        60
SYSERR          EQU        *
*
                STA        SERR
```

```
                PLA
                STA     SDEATH.REGS+PCL.SAVE
                PLA
                STA     SDEATH.REGS+PCH.SAVE
                SEC
                LDA     SERR
                BNE     SERR.EXIT
                CLC
SERR.EXIT       RTS                          ; RETURNS ONE LEVEL BEYOND CALLER
                PAGE
                REP     60
*
* SYSTEM DEATH ROUTINE
*
* CALLED TO IMMEDIATELY TERMINATE EXECUTION OF THE MACHINE
* BECAUSE A FATAL ERROR HAS BEEN DETECTED BY THE OPERATING
* SYSTEM.  THE ERROR CODE IS PASSED IN THE A REG.  THE
* CALL TO THIS ROUTINE MUST ALWAYS BE A JSR.
*
                REP     60
SYSDEATH        EQU     *
*
                STA     SDEATH.REGS+A.SAVE   ; SAVE REGISTERS
                STX     SDEATH.REGS+X.SAVE
                STY     SDEATH.REGS+Y.SAVE
                PHP
                PLA
                STA     SDEATH.REGS+P.SAVE
                TSX
                STX     SDEATH.REGS+S.SAVE
                LDA     E.REG
                STA     SDEATH.REGS+E.SAVE
                LDA     Z.REG
                STA     SDEATH.REGS+Z.SAVE
                LDA     B.REG
                STA     SDEATH.REGS+B.SAVE
                PLA
                STA     SDEATH.REGS+PCL.SAVE
                PLA
                STA     SDEATH.REGS+PCH.SAVE
*
                SEI                          ; TURN OFF INTERRUPTS
                CLD
*
                LDX     #0                   ; SAVE SYSTEM STACK PAGE IN PAGE $17
SD005           LDA     $100,X
                STA     $1700,X
                DEX
                BNE     SD005
*
                LDA     $C059                ; ENSURE SILENTYPE PORT SHUT DOWN
                LDA     $C0DD
                LDA     $C0DF
                LDA     $C05F
                LDA     $C05A
*
                LDA     $C040                ; SOUND BELL
*
                LDA     #$74                 ; ENSURE RESET LOCK OFF & RAM SWITCHED IN.
                STA     E.REG
*
                LDA     TXT.CLR              ; SWITCH TO 40 COL B&W DISPLAY MODE
                LDA     MIX.CLR
                LDA     HIRES.CLR
                LDA     PG2.CLR              ; & SELECT PAGE 1
*
                LDA     #$02
                BIT     SCRNMODE
                BVS     SD015                ; IF GRAPHICS MODE THEN KEEP 40 COL MODE
                BEQ     SD015                ; IF 40 COL MODE THEN KEEP
                LDA     MIX.CLR+1            ; ELSE SWITCH TO 80 COL DISPLAY MODE
*
                LDX     #MSGLEN+1            ; ENSURE BKGRND SET TO INVERSE SPACES
                LDA     #$20                 ; SPACE CHAR W/INVERSE
SD010           STA     MSGBASE2-1,X
                DEX
                BPL     SD010
*
SD015           LDX     #0                   ; MOVE MSG TO TEXT SCREEN
SD020           LDA     MSG,X
                STA     MSGBASE-1,X
```

```
                 INX
                 CPX        #MSGLEN
                 BNE        SD020
*
                 LDA        SDEATH.REGS+A.SAVE   ; DISPLAY ERROR CODE (2 HEX DIGITS)
                 CLC
                 LSR        A
                 LSR        A
                 LSR        A
                 LSR        A
                 JSR        PRINT                ; FIRST DIGIT
                 INX
                 LDA        SDEATH.REGS+A.SAVE
                 AND        #$0F
                 JSR        PRINT                ; SECOND DIGIT
*
                 LDA        #>SD100
                 STA        NMI.VECTOR
                 LDA        #<SD100
                 STA        NMI.VECTOR+1
*
*
                 JMP        *                    ; HANG UNTIL REBOOT (CTRL/RESET)
                 REP        60
SD100            RTI                             ; NMI VECTOR POINT HERE TO MASK THEM OUT
*
*
* PRINT SUBROUTINE
*
PRINT            EQU        *
                 CMP        #$A
                 BCS        PRNT100
                 ADC        #$30                 ; "0"-"9"
                 BCC        PRNT110              ; ALWAYS TAKEN
PRNT100          ADC        #$36                 ; "A"-"F"
PRNT110          STA        MSGBASE-1,X
                 RTS
*
                 LST        ON
ZZEND            EQU        *
ZZLEN            EQU        ZZEND-ZZORG
                 IFNE       ZZLEN-LENSERR
                 FAIL       2,"SOSORG            FILE IS INCORRECT FOR SYSERR"
                 FIN

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

End of File -- Lines: 206 Characters: 5264

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

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

                SBTL       "SOS 1.1  GLOBAL EQUATES"
                REL
                ORG        $18FC
                MSB        OFF
                REP        60
*          COPYRIGHT (C) APPLE COMPUTER INC. 1980
*                    ALL RIGHTS RESERVED
                REP        60
*
*  SOS SYSTEM GLOBAL DATA & EQUATES
*
*  THIS MODULE CONTAINS THE SOS JUMP TABLE, AND ALL GLOBAL
*  DATA AND EQUATES.  THE JUMP TABLE, AND ALL DATA THAT IS
*  TO BE REFERENCED BY DEVICE HANDLERS, ARE ASSIGNED FIXED
*  ADDRESSES AT THE BEGINNING OF MEMORY PAGE $19.  DATA
*  THAT IS ONLY REFERENCED BY SOS BEGINS $1980, BUT MAY BE
*  MOVED WHENEVER SOS IS RELINKED.
*
                REP        60
*
                EXTRN      ALLOCSIR
                EXTRN      DEALCSIR
                EXTRN      NMIDSBL
                EXTRN      NMIENBL
                EXTRN      QUEEVENT
                EXTRN      SELC800
                EXTRN      SYSDEATH
                EXTRN      SYSERR
                EXTRN      REQBUF
                EXTRN      GETBUFADR
                EXTRN      RELBUF
                EXTRN      NMIDBUG
                EXTRN      NMICONT
                EXTRN      COLDSTRT
*
*
                ENTRY      MEMSIZE
                ENTRY      SYSBANK
                ENTRY      SUSPFLSH
                ENTRY      NMIFLAG
                ENTRY      SCRNMODE
                ENTRY      GRSIZE
*
                ENTRY      SERR
                ENTRY      DBUGBRK
                ENTRY      KYBDNMI
                ENTRY      NMISPSV
                ENTRY      SDEATH.REGS
*
                ENTRY      SOSVER
                ENTRY      SOSVERL
*
                ENTRY      SZPAGE
                ENTRY      SXPAGE
                ENTRY      SSPAGE
*
                ENTRY      CZPAGE
                ENTRY      CXPAGE
                ENTRY      CSPAGE
                ENTRY      CEVPRI
*
                ENTRY      SIRTEMP
                ENTRY      SIRARGSIZ
                ENTRY      IRQCNTR
                ENTRY      NMICNTR
                ENTRY      QEVTEMP
                ENTRY      QEV.THIS
                ENTRY      QEV.LAST
*
                ENTRY      BADBRK
                ENTRY      BADINT1
```

```
         ENTRY   BADINT2
         ENTRY   NMIHANG
         ENTRY   EVQOVFL
         ENTRY   STKOVFL
         ENTRY   BADSYSCALL
         ENTRY   DEV.OVFLOW
         ENTRY   MEM2SML
         ENTRY   VCBERR
         ENTRY   FCBERR
         ENTRY   ALCERR
         ENTRY   DIRERR
         ENTRY   TOOLONG
         ENTRY   BADBUFNUM
         ENTRY   BADBUFSIZ
         ENTRY   BITMAPADR
*
         ENTRY   BADSCNUM
         ENTRY   BADCZPAGE
         ENTRY   BADXBYTE
         ENTRY   BADSCPCNT
         ENTRY   BADSCBNDS
*
         ENTRY   NODNAME
         ENTRY   BADDNUM
*
         ENTRY   BADPATH
         ENTRY   CFCBFULL
         ENTRY   FCBFULL
         ENTRY   BADREFNUM
         ENTRY   PATHNOTFND
         ENTRY   VNFERR
         ENTRY   FNFERR
         ENTRY   DUPERR
         ENTRY   OVRERR
         ENTRY   DIRFULL
         ENTRY   CPTERR
         ENTRY   TYPERR
         ENTRY   EOFERR
         ENTRY   POSNERR
         ENTRY   ACCSERR
         ENTRY   BTSERR
         ENTRY   FILBUSY
         ENTRY   NOTSOS
         ENTRY   BADLSTCNT
         ENTRY   OUTOFMEM
         ENTRY   BUFTBLFULL
         ENTRY   BADSYSBUF
         ENTRY   DUPVOL
         ENTRY   NOTBLKDEV
         ENTRY   LVLERR
*
         ENTRY   BADJMODE
*
         ENTRY   BADBKPG
         ENTRY   SEGRQDN
         ENTRY   SEGTBLFULL
         ENTRY   BADSEGNUM
         ENTRY   SEGNOTFND
         ENTRY   BADSRCHMODE
         ENTRY   BADCHGMODE
         ENTRY   BADPGCNT
*
         ENTRY   XREQCODE
         ENTRY   XCTLCODE
         ENTRY   XCTLPARM
         ENTRY   XNOTOPEN
         ENTRY   XNOTAVAIL
         ENTRY   XNORESRC
         ENTRY   XBADOP
         ENTRY   XIOERROR
         ENTRY   XNODRIVE
         ENTRY   XNOWRITE
         ENTRY   XBYTECNT
         ENTRY   XBLKNUM
         ENTRY   XDISKSW
         ENTRY   BACKMASK             ; MASK BYTE FOR BACKUP BIT.
*
         ENTRY   E1908                ; DISK DRIVER IS READING/WRITING (SET) ELSE NOT (RESET)
*
         PAGE
         DW      SYSGLOB              ;SYSGLOB TARGET ADDRESS
```

```
                DW        $0100              ;  AND LENGTH
*
*  SYSTEM GLOBAL DATA
*    (ACCESSIBLE TO SOS AND DEVICE HANDLERS)
*
SYSGLOB         EQU       *
*
MEMSIZE         DFB       $08                ;MEMORY SIZE = 128K
SYSBANK         DFB       $02                ;SYSTEM BANK = 2
SUSPFLSH        DFB       $00                ;SYSOUT SUSPEND/FLUSH FLAG
NMIFLAG         DFB       $00                ;NMI PENDING FLAG
                DW        NMIEXIT            ;DEFAULT NMI VECTOR
SCRNMODE        DFB       $80                ;CURRENT SCREEN MODE
GRSIZE          DFB       $00
*
*
*  SOS JUMP TABLE
*
                DS        SYSGLOB+$10-*,$00  ; USED BY THE MOUSE DRIVER
USERNMI         JMP       NMIEXIT            ;KEYBOARD NMI VECTOR
                JMP       ALLOCSIR           ;ALLOCATE A SIR
                JMP       DEALCSIR           ;DEALLOCATE A SIR
                JMP       NMIDSBL            ;DISABLE NMI
                JMP       NMIENBL            ;ENABLE NMI
                JMP       QUEEVENT           ;QUEUE AN EVENT
                JMP       SELC800            ;SELECT I/O EXPANSION ROM
                JMP       SYSDEATH           ;SYSTEM DEATH
                JMP       SYSERR             ;SOS ERROR
                JMP       REQBUF             ;REQUEST BUFFER
                JMP       GETBUFADR          ;GET BUFFER'S ADDRESS
                JMP       RELBUF             ;RELEASE BUFFER
                JMP       CLRBMASK           ;VECTOR TO CLRBMASK
                PAGE
*
*  SOS DATA AND EQUATES
*    (ACCESSIBLE ONLY TO SOS)
*
                DS        SYSGLOB+$80-*,$00
SERR            DFB       $00                ;SYSTEM ERROR CODE
*
DBUGBRK         NOP                          ;TO ENABLE DEBUG BREAK POINTS,
                PLA                          ;  PATCH THESE BYTES TO
                PLA                          ;  JMP TO THE DEBUGGER
                RTS
*
KYBDNMI         JMP       USERNMI
                JMP       NMIDBUG
NMISPSV         DFB       0
                JMP       NMICONT
NMIEXIT         RTS
*
*
SOSVER          ASC       "SOS 1.3   01-DEC-82"
SOSVERL         EQU       *-SOSVER
*
                ASC       "(C) 1980, 1981 BY APPLE COMPUTER INC."
*
E1908           EQU       $1908              ; ALLOCATED TO STEPHEN SMITH (MOUSE DRIVER)
* ABOVE SET AND RESET IN DISK DRIVER
SZPAGE          EQU       $1800              ;SYSTEM ZERO PAGE
SXPAGE          EQU       $1400              ;SYSTEM EXTEND PAGE
SSPAGE          EQU       $0100              ;SYSTEM STACK PAGE
*
CZPAGE          EQU       $1A00              ;CALLER'S ZERO PAGE
CXPAGE          EQU       $1600              ;CALLER'S EXTEND PAGE
CSPAGE          EQU       $1B00              ;CALLER'S STACK PAGE
CEVPRI          DFB       $00                ;CALLER'S EVENT PRIORITY
*
SIRTEMP         DFB       $00                ;TEMP FOR ALLOCSIR & DEALCSIR
SIRARGSIZ       DFB       $00                ;ARGUMENT COUNT FOR ALLOCSIR & DEALCSIR
IRQCNTR         DW        $0000              ;FALSE IRQ COUNTER
NMICNTR         DW        $0000              ;COUNTER FOR NMILOCK
QEVTEMP         DFB       $00                ;TEMP FOR QUEEVENT
QEV.THIS        DFB       $00                ;POINTER FOR QUEEVENT
QEV.LAST        DFB       $00                ;POINTER FOR QUEEVENT
*
SOSQUIT         DS        COLDSTRT
BACKMASK        DFB       BACKBIT            ; MASK USED BY BFM TO UPDATE BACKUP BIT
*
* TO CLEAR THE BACKUP BIT, A PROGRAM MUST JSR TO CLRBMASK THRU 1934 THEN DO A
* SET-FILE-INFO WITH NO INTERVENING SOS CALLS.  ANY SOS CALL WILL
```

```
ADR060          CLI                            ;PROCESS AN INTERRUPT
                SEI
                BIT       AD.FLAG              ;STILL TIMING?
                BPL       ADR040               ;  NO -- START OVER
                JSR       ANLOG1               ;  YES -- CONTINUE
                BCS       ADR060
*
ADR070          PLP
                EOR       #$FF                 ;NORMALIZE RESULT
                BMI       ADR080               ;RESULT < 0
                STA       AD.TEMP
                TYA
                EOR       #$FF
                LSR       AD.TEMP
                ROR       A
                LSR       AD.TEMP
                ROR       A
                LSR       AD.TEMP
                BNE       ADR090               ;RESULT > 255
                ROR       A
                ADC       #0
                RTS
ADR080          LDA       #0
                RTS
ADR090          LDA       #$FF
                RTS
                PAGE
                REP       60
*
*  SYSTEM COLD START
*
*  THIS ROUTINE IS CALLED TO TELL THE USER TO REBOOT THE
*  SYSTEM.  IT CLEARS THE SCREEN, DISPLAYS A MESSAGE,
*  OVERWRITES BANKED MEMORY, AND HANGS UNTIL THE USER
*  PERFORMS A HARD RESET.
*
                REP       60
*
*
COLDSTRT        EQU       *
                SEI                            ;SHUT DOWN INTERRUPTS
                LDA       #$40                 ;  AND IGNORE NMI
                STA       $FFCA
                LDA       #$67
                STA       E.REG                ;DISABLE RESET
                LDA       #$00
                STA       Z.REG                ;USE PAGE ZERO
*
                LDX       SYSBANK
                LDA       #$BF
                LDY       #$00
                STY       MEMORY
CS010           STA       MEMORY+1
                STX       B.REG
                LDA       #$A0
CS020           STA       (MEMORY),Y           ;SET MEMORY TO BLANKS
                DEY
                BNE       CS020
                DEC       MEMORY+1
                BNE       CS020
                DEX
                BPL       CS010
*
                LDY       #6
CS030           STA       $C050,Y              ;SELECT 40 COLUMN
                DEY                            ;  BLACK & WHITE TEXT
                BPL       CS030
*
                LDY       #BOOTLEN
CS040           LDA       BOOTMSG-1,Y          ;PRINT BOOT MESSAGE
                STA       BOOTADR-1,Y
                DEY
                BNE       CS040
*
                LDA       #$77
                STA       E.REG                ;ENABLE RESET
                JMP       *                    ;HANG UNTIL RESET
                PAGE
                MSB       ON
BOOTMSG         ASC       "INSERT SYSTEM DISKETTE & REBOOT"
BOOTLEN         EQU       *-BOOTMSG
```

```
BOOTADR         EQU     40-BOOTLEN/2+$628
                MSB     OFF
                LST     ON
ZZEND           EQU     *
ZZLEN           EQU     ZZEND-ZZORG
                IFNE    ZZLEN-LENUMGR
                FAIL    2,"SOSORG           FILE IS INCORRECT FOR UMBR"
                FIN

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

```

End of File -- Lines: 818 Characters: 20770

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

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

                PAGE
                REP        40
* NAME    : VOLUME
* FUNCTION: RETURN VOLUME INFO
* INPUT   : DEVICE NAME
* OUTPUT  : THE INFO
* VOLATILE: ALL REGS
                REP        40
*
VOLUME          EQU        *
                LDA        C.DNAMP             ; TRANSFER DEVICE NAME
                STA        DVNAMP              ; NAME FOR DMGR
                LDA        C.DNAMP+1
                STA        DVNAMP+1
                LDA        SISTER+C.DNAMP+1    ; AND XTND
                STA        SISTER+DVNAMP+1
                JSR        GETDNUM             ; GET DEVNUM
                BCC        VOL7                ; =>SOME KINDA ERROR
                RTS                            ; RETURN ERROR
VOL7            BMI        VOL2                ; =>IT'S GOOD...
                LDA        #NOTBLKDEV          ; NOT BLOCKED
                JMP        VOLERR              ; =>RETURN THE ERROR
*
* UNCONDITIONALLY READ ROOT DIRECTORY:
*
VOL2            EQU        *
                LDA        SCRTCH+1
                STA        DEVNUM              ; SETUP DEV NUMBER
                LDA        #2                  ; BLKNUM=2
                LDX        #0
                JSR        GETROT0             ; GET IT PLEASE
                LDA        #VNFERR             ; ERROR CODE
                BCC        VOL8                ; BRANCH IF NO ERROR ON READ
                RTS                            ; =>ERROR, PASS IT ON.
*
VOL8            LDA        #>VCB               ; SET VCBPTR TO THE
                STA        VCBPTR              ; FIRST OF THEM
                LDA        #<VCB
                STA        VCBPTR+1
*
* IS THIS VOLUME SOS OR OTHER?
*
                JSR        TSTSOS              ; WHICH KIND?
                BCC        VLOGGED             ; =>IT'S SOS
                JMP        VNOTSOS             ; =>NOT SOS
*
* IS THIS SOS VOLUME LOGGED IN?
*
VLOGGED         EQU        *
                JSR        CMPVCB              ; DOES VOLNAME MATCH?
                BCC        VFOUND              ; =>YES, WE KNOW ABOUT IT.
                JSR        VNXTVCB             ; BUMP TO NEXT
                BCC        VLOGGED             ; =>TRY 'EM ALL...
                BCS        VNEW                ; =>NOT FOUND, IT'S NEW (BRANCH ALWAYS)
*
*
* IT'S BEEN LOGGED IN BEFORE:
*  IS IT SWAPPED IN OR OUT?
*
VFOUND          EQU        *
                LDY        #VCBSWAP            ; INDEX TO IT
                LDA        (VCBPTR),Y          ; SWAPPED?
                BPL        VFOUND1             ; =>IN. RETURN THE INFO
*
* SWAPPED OUT. BEFORE WE SWAP IT
*  IN, MAKE SURE IT BELONGS ON
*  THIS DEVICE!
*
                LDY        #VCBDEV             ; INDEX TO IT
                LDA        (VCBPTR),Y          ; GET ITS DEVICE
                CMP        DEVNUM              ; CORRECT DEVICE?
```

```
                BEQ     VSWAPIN             ; =>YES
                LDA     #DUPVOL             ; IF FOR ANOTHER DEV,
                JMP     VOLERR              ; THEN IT'S AN ERROR!
*
* NOW SWAP-IN THIS VOLUME:
*
VSWAPIN         EQU     *
                JSR     SWAPIN              ; SWAP IT IN
                JMP     VINFO               ; AND RETURN THE INFO
*
VFOUND1         LDY     #VCBDEV
                LDA     (VCBPTR),Y          ; SAME DEVICES?
                CMP     DEVNUM
                BEQ     VINFO               ; YES; RETURN THE INFORMATION
                LDY     #VCBSTAT
                LDA     (VCBPTR),Y          ; OPEN FILES?
                BPL     VFOUND2             ; BRANCH IF NOT
                LDA     #DUPVOL
                BNE     VOLERR              ; ELSE REPORT DUPLICATE VOLUME ERROR (BRANCH ALWAYS)
VFOUND2         LDY     #VCBNML             ; MOVE THE LOGIN TO THIS NEW DEVICE
                LDA     #0                  ; BY UNLOGGING THE OLD
                STA     (VCBPTR),Y          ; AND LOGGING IN THE NEW (DROP INTO VNEW)
                REP     40
*
* IT'S A BRAND NEW VOLUME.
*  GUESS WE'LL HAVE TO LOG IT IN:
*
VNEW            EQU     *
                LDA     DEVNUM              ; PASS A REG TO SWAPOUT
                JSR     SWAPOUT             ; SWAP ANY ACTIVE VOL ON THIS DEVICE
                BCC     VNEW1               ; BRANCH ON NO ERROR
                LDA     #XIOERROR
                RTS
VNEW1           LDA     #>VCB               ; FIND AN EMPTY VCB
                STA     VCBPTR
                LDA     #<VCB
                STA     VCBPTR+1
VFREE           LDY     #VCBNML
                LDA     (VCBPTR),Y          ; EMPTY VCB?
                BEQ     VLOGIN              ; ITS FREE, USE IT
                LDY     #VCBDEV
                LDA     (VCBPTR),Y          ; OR ONE WITH SAME DEVICE
                CMP     DEVNUM
                BNE     VFREEX              ; BRANCH IF NO DEVICE MATCH
                LDY     #VCBSTAT
                LDA     (VCBPTR),Y          ; AND NO OPEN FILES
                BPL     VLOGIN              ; BRANCH IF OK TO REUSE THIS VCB
                LDA     DEVNUM              ; THEN WE MUST SWAP OUT THIS VOLUME
                JSR     SWAPOUT
                BCC     VFREEX              ; SWAPOUT PROCEEDED OK
                LDA     #XIOERROR           ; ELSE REPORT ERROR
                RTS
VFREEX          JSR     VNXTVCB             ; TRY NEXT
                BCC     VFREE               ; MORE TO COME
* RAN OUT OF MT'S ... FIND W/O FILES
VNFIL           LDY     #VCBSTAT
                LDA     (VCBPTR),Y
                BPL     VLOGIN
                JSR     VNXTVCB
                BCC     VNFIL
* ALL OPEN ... REPORT VCBFULL
                LDA     #FCBFULL
                BNE     VOLERR
VLOGIN          EQU     *
                JSR     LOGVCB              ; AND LOGIN THIS ONE
                REP     40
*
* RETURN ALL THE NICE INFO:
*
VINFO           EQU     *
                LDA     #0
                LDY     #VCBTFRE            ; FETCH VOLUME FREE BLOCK COUNT
                STA     (VCBPTR),Y          ; FORCE RESCAN OF ALL
                INY                         ; BITMAPS
                STA     (VCBPTR),Y          ; TO MAKE SURE VCB INFO CURRENT
                STA     REQL                ; FREE BLOCKS
                STA     REQH
                JSR     TSFRBLK
*
                LDX     VCBPTR              ; GET VCB INDEX
                LDY     #0
```

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

                CHN       CREATE,4,1

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

End of File -- Lines: 225 Characters: 7048

SUMMARY:

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