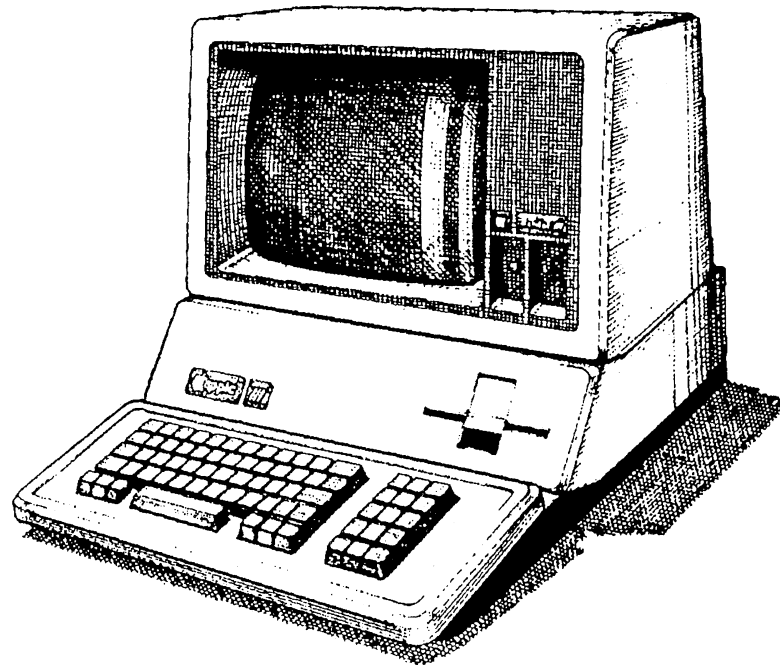




Apple /// Computer Technical Information

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



## How This Document Was Created

David T. Craig - 29 August 2001

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

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

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

### *DTCCatTextDocsRecursive*

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

### *DTCEntabNOT*

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

DTCEntabNOT 8 SOS.SOURCE.ALL.FILES.TEXT > SOS.SOURCE.ALL.FILES2.TEXT

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

Tab width = 8

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

That's all folks!

#### *Microsoft Word 5.1a*

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

#### *PDFWriter*

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

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

```

000001 =====
000002 DOCUMENT :SOS.SOURCE.INFO:SOS.AAA.1.README.TEXT
000003 =====
000004
000005
000006 =====
000007
000008             READ ME FILE FOR SOS SOURCE CODE DISK
000009
000010             Publicus / David T Craig  --  March 1993
000011
000012 =====
000013
000014 This Macintosh 800K HFS disk contains the complete source code listing for
000015 the Apple /// computer's operating system, SOS.  This source listing is for
000016 version 1.3 of SOS, the last released SOS.  Note that Apple had (to my
000017 knowledge) 3 SOS releases: 1.0, 1.1, 1.3 (version 1.2 appears to have not
000018 been released to the public).  Version 1.3's release date is February 1982.
000019
000020 SOS may be read as "Sophisticated Operating System" or "Sara's Operating
000021 System" since the Apple /// computer was code-named "SARA" by Apple Computer.
000022
000023 The Apple /// was Apple's premier business computer system for the time
000024 period 1980 to 1983.
000025
000026 This source listing is written in 6502 assembly language.  The assembler
000027 used by Apple was an Apple ][ computer assembler which ran on a networked
000028 collection of Apple ][ computers.  I have been told by knowledgeable ///
000029 owners that the SOS source code was never ported to an Apple /// even though
000030 the /// had a nice assembler (as part of the ///'s Pascal development system).
000031
000032 For a detailed discussion of SOS see Apple Computer's well-written
000033 "SOS Reference Manual" series (two volumes).
000034
000035 From a historical perspective this source code is of no real use today since
000036 it is for a discontinued computer system.  From a technical perspective this
000037 source is interesting since it provides a "real world" example of an
000038 operating system for a microcomputer.  From a legal perspective this source
000039 is rather sensitive since parts of it may be used by Apple in its ProDOS
000040 operating system for the Apple ][ series (includes the //e and //GS).
000041
000042 Due to the legal ramifications of the SOS source code the author of this
000043 READ ME file shall remain anonymous.
000044
000045 This author would very much like to learn a little about how Apple developed
  
```

000046 SOS. If any former /// development team members ever read this file, I hope  
000047 that one of them will write a short "SOS History" and place it in a publically  
000048 accessable area (e.g. CompuServe Information System).  
000049  
000050 Enjoy ...  
000051  
000052 =====  
000053



```

000103  PATH                33 31-Dec-89 19:30 Asciifile 497 34
000104  VOLUME                9 31-Dec-89 19:31 Asciifile 369 10
000105  CREATE                30 12-Jan-89 22:30 Asciifile 441 31
000106  8 files listed, 82 blocks available
000107
000108  /SOS1.3.FOUR          Size  Modified  Time  File type  Eof  Phys
000109  SWAPOUT.IN            21 31-Dec-89 19:47 Asciifile 303 22
000110  CLOSE.EOF             23 31-Dec-89 19:48 Asciifile 87 24
000111  READ.WRITE            38 31-Dec-89 19:49 Asciifile 86 39
000112  DESTROY               28 31-Dec-89 19:50 Asciifile 242 29
000113  POSN.OPEN             44 31-Dec-89 19:52 Asciifile 243 45
000114  5 files listed, 114 blocks available
000115
000116  /SOS1.3.FIVE          Size  Modified  Time  File type  Eof  Phys
000117  LCHK                  1 31-Dec-89 19:59 Asciifile 147 1
000118  LC                    1 31-Dec-89 20:00 Asciifile 74 1
000119  COMPILE.BFM           1 31-Dec-89 20:00 Asciifile 68 1
000120  COMPILE.SOS           2 31-Dec-89 20:01 Asciifile 97 3
000121  SOS.BLOAD             1 31-Dec-89 20:02 Asciifile 450 1
000122  SOS.LINK              1 31-Dec-89 20:03 Asciifile 170 1
000123  SOS.RENAME            2 31-Dec-89 20:03 Asciifile 11 3
000124  FEB01.1982           2 31-Dec-89 20:04 Asciifile 64 3
000125  PUBLICRELEASE         1 31-Dec-89 20:05 Asciifile 71 1
000126  COMP.SOS.NOLIST       2 31-Dec-89 20:05 Asciifile 79 3
000127  TCOMP.SOS             1 31-Dec-89 20:06 Asciifile 388 1
000128  SOSORG                5 31-Dec-89 20:07 Asciifile 428 6
000129  C.S                   1 31-Dec-89 20:08 Asciifile 116 1
000130  C.BI2                 1 31-Dec-89 20:09 Asciifile 76 1
000131  C3                    1 31-Dec-89 20:09 Asciifile 155 1
000132  COMP.OPR.IPL          1 31-Dec-89 20:10 Asciifile 124 1
000133  16 files listed, 244 blocks available
000134
000135  <<< END OF CATALOG LISTING >>>
000136
000137

```

```

000138 =====
000139 DOCUMENT :SOS.SOURCE.INFO:SOS.AAA.3.OPCODEFREQS.TEXT
000140 =====
000141
000142 APPLE /// SOS 1.3 OPCODE INFORMATION
000143
000144 ##### OPCODE LISTS
000145
000146     Sorted by Name:
000147
000148     Opcode count :    86
000149     Min frequency:    1
000150     Max frequency: 1864
000151
000152     #   Opcode Name   Freq   Histogram
000153     ---  -
000154     [ 1] .PAGE        2    **
000155     [ 2] ADC          125   ****
000156     [ 3] AND          147   *****
000157     [ 4] ASC          32    **
000158     [ 5] ASL          46    **
000159     [ 6] BCC          240   *****
000160     [ 7] BCS          274   *****
000161     [ 8] BEQ          291   *****
000162     [ 9] BIT          58    ***
000163     [10] BMI          59    ***
000164     [11] BNE          397   *****
000165     [12] BPL          121   ****
000166     [13] BRK          12    **
000167     [14] BVC          12    **
000168     [15] BVS          7     **
000169     [16] CHN          28    **
000170     [17] CHR          1     *
000171     [18] CLC          181   *****
000172     [19] CLD          3     **
000173     [20] CLI          5     **
000174     [21] CLV          4     **
000175     [22] CMP          286   *****
000176     [23] CPX          40    **
000177     [24] CPY          56    ***
000178     [25] DEC          73    ***
000179     [26] DEND         4     **
000180     [27] DEX          75    ***
000181     [28] DEY          126   ****
000182     [29] DFB          188   *****
000183     [30] DO           10    **
000184     [31] DS           210   *****
000185     [32] DSECT        4     **
000186     [33] DW           83    ***

```



```

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

```

```
000237 [ 84] TXS      6 **
000238 [ 85] TYA     58 ***
000239 [ 86] ZZLEN-LENLODR 1 *
```

Sorted by Static Frequency:

```
000243 Opcode count : 86
000244 Min frequency: 1
000245 Max frequency: 1864
```

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

000287	[ 39]	DEX	75	***
000288	[ 40]	DEC	73	***
000289	[ 41]	BMI	59	***
000290	[ 42]	BIT	58	***
000291	[ 43]	TYA	58	***
000292	[ 44]	CPY	56	***
000293	[ 45]	ASL	46	**
000294	[ 46]	CPX	40	**
000295	[ 47]	INX	33	**
000296	[ 48]	ASC	32	**
000297	[ 49]	SBTL	32	**
000298	[ 50]	CHN	28	**
000299	[ 51]	FIN	27	**
000300	[ 52]	ORG	24	**
000301	[ 53]	PHP	23	**
000302	[ 54]	ROR	23	**
000303	[ 55]	SEI	23	**
000304	[ 56]	MSB	22	**
000305	[ 57]	EOR	21	**
000306	[ 58]	INCLUDE	21	**
000307	[ 59]	PLP	20	**
000308	[ 60]	FAIL	17	**
000309	[ 61]	IFNE	16	**
000310	[ 62]	REL	16	**
000311	[ 63]	ROL	14	**
000312	[ 64]	LST	13	**
000313	[ 65]	BRK	12	**
000314	[ 66]	BVC	12	**
000315	[ 67]	DO	10	**
000316	[ 68]	BVS	7	**
000317	[ 69]	TSX	7	**
000318	[ 70]	TXS	6	**
000319	[ 71]	CLI	5	**
000320	[ 72]	ELSE	5	**
000321	[ 73]	CLV	4	**
000322	[ 74]	DEND	4	**
000323	[ 75]	DSECT	4	**
000324	[ 76]	CLD	3	**
000325	[ 77]	RTI	3	**
000326	[ 78]	SKP	3	**
000327	[ 79]	.PAGE	2	**
000328	[ 80]	NOP	2	**
000329	[ 81]	CHR	1	*
000330	[ 82]	ERROR	1	*
000331	[ 83]	IBUFSIZ	1	*
000332	[ 84]	SBUFSIZ	1	*
000333	[ 85]	TSTERR	1	*
000334	[ 86]	ZZLEN-LENLODR	1	*
000335				
000336				

000337 ### FINIS: Assembly Source Code File Beautifier [0.8] 28-Mar-93  
000338  
000339

```

000340 =====
000341 DOCUMENT :SOS1.3.1of5.ONE:SOS.BFM.INIT2.SRC.TEXT
000342 =====
000343
000344 *****
000345 * APPLE /// SOS 1.3 SOURCE CODE FILE: BFM.INIT2.SRC
000346 *****
000347 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000348
000349             SBTL          "SOS 1.1  BFM.INIT2"
000350             REL
000351             INCLUDE      SOSORG,6,1,254
000352             ORG          ORGBFMI
000353             MSB          OFF
000354             REP          60
000355 *             COPYRIGHT (C) APPLE COMPUTER INC.  1980
000356 *             ALL RIGHTS RESERVED
000357             REP          60
000358 *
000359 * BLOCK FILE MANAGER INIT2
000360 *
000361 * SECONDARY INITIALIZATION ROUTINE FOR BLOCK FILE MANAGER
000362 *
000363 * MODIFIED: 03/25/81 TO UTILIZE NEW
000364 * DISK DRIVER'S SEEKDSK3 ROUTINE.
000365 * CHANGES MARKED BY 'D3RRA81084'
000366 *
000367 * MODIFIED: 08/19/81 TO WORK WITH NEW
000368 * SOSLDR MODULE.
000369             REP          60
000370 *
000371             ENTRY        BFM.INIT2
000372 *
000373 *EXTRN I.BASE.P ; ENTRY IN SOSLDR
000374             EXTRN        SYSBANK
000375             EXTRN        SXPAGE
000376             EXTRN        CZPAGE
000377             EXTRN        SEEKDSK3             ;IN DISKDH/D3RRA81084
000378             EXTRN        NMIDSBL             ;/D3RRA81084
000379 I.BASE.P     EQU          $2
000380             PAGE
000381 *
000382 * CONSTANTS
000383 *
000384 KERNEL.BASE  EQU          $B800             ; BASE ADDRESS OF SOS KERNEL
000385 ROMID        EQU          $A0              ;$F1B9 OF NEW ROM/D3RRA81084
000386 SLOT        EQU          $60
000387 BEGTRK       EQU          $9
000388 BEGSECT      EQU          $2

```

```

000389 ENDSECT      EQU      $6
000390 *
000391 * ZERO PAGE
000392 *
000393 TRACK         EQU      $99
000394 SECTOR       EQU      $98
000395 VOLUME       EQU      $9A
000396 KEY          EQU      $E0          ; THRU $E7
000397 PREV.K      EQU      KEY+$8
000398 XIDX        EQU      KEY+$9
000399 I           EQU      KEY+$A          ; & $B
000400 *
000401 * ROM ROUTINES
000402 *
000403 RDADR        EQU      $F1B9          ;REV1
000404 RDADR        EQU      $F1BD          ;REV0
000405 *
000406 * HARDWARE LOCATIONS
000407 *
000408 E.REG         EQU      $FFDF
000409 B.REG         EQU      $FFEF
000410 MOTORON      EQU      $C089
000411 MOTOROFF     EQU      $C088
000412           PAGE
000413           REP      60
000414 *
000415 * BFM.INIT2 ENTRY POINT
000416 *
000417           REP      60
000418 *
000419 STATE        DFB      $FE          ; FF=1ST ENTRY, 0=2ND ENTRY, 1=PROT
000420 *
000421 BFM.INIT2     EQU      *
000422           INC      STATE
000423           BMI      BFMIO50
000424           JSR      GETK
000425           LDA      RETRY
000426           BEQ      BADNEWS
000427           BCC      BFMIO50
000428           JSR      NMIDSBL
000429           JSR      DC
000430           INC      STATE
000431 BFMIO50      CLC
000432           RTS
000433 BADNEWS     SEC          ; I/O ERROR
000434           RTS
000435           PAGE
000436           REP      60
000437 *
000438 * DECODE SUBROUTINE

```

```

000439 *
000440 * TO ENCODE:
000441 *   E0.E8:      - INIT KEY  & PREV.K
000442 *   B84E:4C 64 B8 - JUMPS AROUND INTERP'S 3 BYTE OVERWRITE
000443 *   1A02.1A03:  - NEW INTERP'S LOAD ADR (LO,HII)
000444 *   B81DG:    - JSR FROM MONITOR
000445 *
000446          REP          60
000447 DC      EQU          *
000448          LDA          B.REG          ; SAVE BANK REGISTER
000449          PHA
000450          LDA          SYSBANK        ; AND SWITCH TO SYSTEM BANK
000451          STA          B.REG
000452          CLC          ; FETCH LOADER'S INTERPRETER POINTER
000453          LDA          CZPAGE+I.BASE.P
000454          ADC          #3
000455          STA          I
000456          PHA
000457          LDA          CZPAGE+I.BASE.P+1
000458          ADC          #0
000459          STA          I+1
000460          PHA
000461          LDA          #0
000462          STA          SXPAGE+I+1
000463 *
000464          LDY          I          ; ALIGN I PTR TO PAGE BOUNDARY
000465          LDA          #0
000466          STA          I
000467          STA          PREV.K
000468 *
000469          JSR          DCLOOP        ; DECODE
000470 *
000471          PLA          ; RETRIEVE LOADER'S INTERPRETER POINTER
000472          STA          I+1
000473          PLA
000474          STA          I
000475 *
000476          LDY          #1          ; REPOSITION LOADER'S INTERPRETER POINTER (PUT ENCODE JMP HERE)
000477          LDA          (I),Y
000478          STA          CZPAGE+I.BASE.P
000479          INY
000480          LDA          (I),Y
000481          STA          CZPAGE+I.BASE.P+1
000482 *
000483          LDY          #2          ; WALK ON INTERPRETER'S FIRST INSTRUCTION (3 BYTES)
000484          LDA          #0
000485 DCA      STA          (I),Y
000486          DEY
000487          BPL          DCA
000488          PLA          ; RESTORE BANK REGISTER (ENCODE JMP JUMPS TO HERE)

```

```

000489          STA      B.REG
000490          RTS
000491          PAGE
000492          REP      60
000493 *
000494 * DECODE LOOP SUBROUTINE
000495 *
000496          REP      60
000497 DCLOOP      EQU      *
000498          LDX      #7          ; SHIFT LEFT ONE BIT
000499          CLC
000500          LDA      KEY
000501          BPL      DC1
000502          SEC
000503 DC1        ROL      KEY,X
000504          DEX
000505          BPL      DC1
000506 *
000507 DC2        TYA
000508          AND      #7
000509          EOR      #2
000510          TAX
000511          LDA      KEY,X
000512          PHA
000513          AND      #7
000514          TAX
000515          PLA
000516          CLC
000517          ADC      PREV.K
000518          CLC
000519          ADC      KEY,X
000520          STA      PREV.K
000521          EOR      (I),Y      ; DECODE BYTE
000522          STA      (I),Y      ; AND PUT IT BACK
000523          INY
000524          BNE      DC2
000525          INC      I+1
000526          LDA      I+1
000527          CMP      #<KERNEL.BASE
000528          BCC      DCLOOP
000529          RTS
000530          PAGE
000531          REP      60
000532 *
000533 * GETKEY SUBROUTINE
000534 *
000535          REP      60
000536 *
000537 RETRY        DFB      10+1      ;TEN RETRIES
000538 OURTRACK    DS       1          ;CURRENT TRACK/D3RRA81084

```



```

000539 *
000540 GETK      EQU      *
000541          LDX      #7
000542          STX      XIDX
000543          LDX      #SLOT
000544          LDA      MOTORON,X          ;ENSURE MOTOR STAYS ON
000545          LDA      E.REG              ; SELECT 1MHZ, ROM
000546          ORA      #$83
000547          STA      E.REG
000548 *
000549 * NOTE: THE SEEKDSK3 ROUTINE HAS THESE /D3RRA81084
000550 * CAVEATS: 1MHZ MODE, MOTOR IS ON, /D3RRA81084
000551 * DRIVE CURRENTLY SELECTED, ROM+I/O ENABLED! /D3RRA81084
000552 *
000553 GETK010    LDA      #BEGTRK
000554          STA      OURTRACK          ;WHERE WE SEEK TO /D3RRA81084
000555          JSR      SEEKDSK3        ;HAVE DISKDH SEEK FOR US /D3RRA81084
000556 GETK020    LDX      #SLOT
000557          JSR      DOREAD          ;FIND A SECTOR HEADER
000558          BCS      IOERROR        ;=>RETRY IF BAD
000559          LDA      SECTOR          ;WHERE ARE WE?
000560          CMP      #BEGSECT        ;AT THE RIGHT PLACE?
000561          BNE      GETK020        ;=>NO, GET THERE
000562 *
000563 GETK100    LDX      #1
000564          JSR      WAIT            ; (X * 1284) + 15 MILISECONDS
000565          LDX      XIDX
000566          LDA      VOLUME
000567          STA      KEY,X
000568          DEC      XIDX
000569          BMI      ENUFF
000570          INC      OURTRACK        ;BUMP FOR NEXT TRACK /D3RRA81084
000571          LDA      OURTRACK        ;WHERE TO GO /D3RRA81084
000572          LDX      #SLOT
000573          JSR      SEEKDSK3        ;DISKDH, PLEASE SEEK ME /D3RRA81084
000574          LDX      #SLOT
000575          JSR      DOREAD
000576          BCC      GETK100
000577          BCS      IOERROR
000578 *
000579 ENUFF     LDX      #SLOT
000580          LDA      MOTOROFF,X
000581          LDA      E.REG              ; SELECT 2MHZ, RAM
000582          AND      #$7C
000583          STA      E.REG
000584          PAGE
000585          LDA      SECTOR
000586          CMP      #ENDSECT        ;TRACKS SYNC'ED?
000587          BNE      NOTPROT
000588          LDA      KEY

```

```

000589          EOR          KEY+1
000590          BEQ          NOTPROT          ;IF FIRST 2 VOLS ARE EQUAL
000591          SEC
000592          RTS
000593          *
000594 NOTPROT          LDA          #0
000595          CLC
000596          RTS
000597          *
000598          *
000599 DOREAD          JSR          WHICHROM
000600          BCS          OLDREAD
000601          JMP          RDADR
000602 OLDREAD          JMP          RDADR
000603          *
000604          *
000605 WHICHROM          LDA          RDADR
000606          CMP          #ROMID
000607          CLC
000608          BEQ          NEWROM
000609          SEC
000610 NEWROM          RTS
000611          *
000612          *
000613 IOERROR          DEC          RETRY
000614          BEQ          ERR1
000615          JMP          GETK          ; TRY, TRY AGAIN
000616 ERR1          JMP          ENUFF          ; I/O ERROR, CLEANUP AND EXIT
000617          *
000618          *
000619 WAIT          LDY          #0
000620 W1          DEY
000621          BNE          W1
000622          DEX
000623          BNE          W1
000624          RTS
000625
000626 ZZLEN          EQU          $400
000627          IFNE          ZZLEN-LENBFMI
000628          FAIL          2,"SOSORG          FILE IS INCORRECT FOR BFM.INIT2"
000629          FIN
000630
000631 *****
000632 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: BFM.INIT2.SRC
000633 *****
000634
000635
000636

```

```

000637 =====
000638 DOCUMENT :SOS1.3.1of5.ONE:SOS.INIT.SRC.TEXT
000639 =====
000640
000641 *****
000642 * APPLE /// SOS 1.3 SOURCE CODE FILE: INIT.SRC
000643 *****
000644 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
000645
000646             SBTL             "SOS 1.1 INITIALIZATION"
000647             REL
000648             INCLUDE          SOSORG,6,1,254
000649             ORG              ORGINIT
000650 ZZORG             EQU              *
000651             MSB              OFF
000652             REP              100
000653 *             COPYRIGHT (C) APPLE COMPUTER INC. 1981
000654 *             ALL RIGHTS RESERVED
000655             REP              100
000656 *
000657 * SOS INIT MODULE (VERSION = 1.10 )
000658 *             (DATE        = 8/04/81)
000659 *
000660             REP              100
000661 *
000662             ENTRY           INT.INIT
000663             ENTRY           EVQ.INIT
000664             ENTRY           CLK.INIT
000665             ENTRY           MMGR.INIT
000666             ENTRY           BMGR.INIT
000667             ENTRY           DMGR.INIT
000668             ENTRY           CFMGR.INIT
000669             ENTRY           BFM.INIT
000670 *
000671 * EXTERNAL SUBROUTINES & DATA
000672 *
000673             EXTRN           SXPAGE
000674             EXTRN           SYSDEATH
000675 *
000676 * INTERRUPT SYSTEM INITIALIZATION
000677 *
000678             EXTRN           COLDSTRT
000679             EXTRN           IRQ.RCVR
000680             EXTRN           NMI.RCVR
000681             EXTRN           NMIFLAG
000682             EXTRN           SIRTABLE
000683             EXTRN           SIRTBSIZ
000684             EXTRN           ZPGSTACK
000685             EXTRN           ZPGSTART

```

```

000686 *
000687 * EVENT QUEUE INITIALIZATION
000688 *
000689         EXTRN     EV.QUEUE
000690         EXTRN     EVQ.LEN
000691         EXTRN     EVQ.CNT
000692         EXTRN     EVQ.SIZ
000693         EXTRN     EVQ.FREE
000694         EXTRN     EVQ.LINK
000695 *
000696 * CLOCK INITIALIZATION
000697 *
000698         EXTRN     PCLOCK
000699 *
000700 * CHARACTER FILE MANAGER INITIALIZATION
000701 *
000702         EXTRN     CFCB.MAX
000703         EXTRN     CFCB.DEV
000704 *
000705 * DEVICE MANAGER INITIALIZATION
000706 *
000707         EXTRN     DMGR
000708         EXTRN     MAX.DNUM
000709 *
000710 * BUFFER MANAGER INITIALIZATION
000711 *
000712         EXTRN     BUF.CNT
000713         EXTRN     PGCT.T
000714         EXTRN     XBYTE.T
000715         EXTRN     BUFREF
000716 *
000717 * MEMORY MANAGER INITIALIZATION
000718 *
000719         EXTRN     ST.CNT
000720         EXTRN     ST.ENTRY
000721         EXTRN     ST.FREE
000722         EXTRN     ST.FLINK
000723         EXTRN     VRT.LIM
000724         EXTRN     MEMSIZE
000725         EXTRN     MEM2SML
000726 *
000727 * BLOCK FILE MANAGER INITIALIZATION
000728 *
000729         EXTRN     FCBZPP
000730         EXTRN     PATHBUF
000731         EXTRN     VCB
000732         EXTRN     WORKSPC
000733         EXTRN     PFXPTR
000734         EXTRN     FCBADDRH
000735         EXTRN     BMAPAGE

```

```

000736          EXTRN      BMBPAGE
000737          EXTRN      BMAMADR
000738          EXTRN      BMBMADR
000739          EXTRN      BFMFCB1
000740          EXTRN      BFMFCB2
000741 *
000742 *  CONSTANT DECLARATIONS
000743 *
000744 TRUE          EQU      $80
000745 FALSE         EQU      $00
000746 BITON6        EQU      $40
000747 BITON7        EQU      $80
000748 *
000749 *  SYSTEM CONTROL REGISTERS
000750 *
000751 E.REG          EQU      $FFDF          ;ENVIRONMENT REGISTER
000752 Z.REG          EQU      $FFD0          ;ZERO PAGE REGISTER
000753          SBTL      "INTERRUPT SYSTEM INITIALIZATION"
000754 *
000755 *  6522 REGISTERS
000756 *
000757 D.DDRB         EQU      $FFD2
000758 D.DDRA         EQU      $FFD3
000759 D.ACR          EQU      $FFDB
000760 D.PCR          EQU      $FFDC
000761 D.IFR          EQU      $FFDD
000762 D.IER          EQU      $FFDE
000763 E.IORB         EQU      $FFE0
000764 E.DDRB         EQU      $FFE2
000765 E.DDRA         EQU      $FFE3
000766 E.ACR          EQU      $FFEB
000767 E.PCR          EQU      $FFEC
000768 E.IFR          EQU      $FFED
000769 E.IER          EQU      $FFEE
000770 ACIASTAT     EQU      $C0F1
000771 *
000772 *
000773          REP      60
000774 *
000775 *  THIS SUBROUTINE INITIALIZES THE INTERRUPT SYSTEM.
000776 *  ALL HARDWARE INTERRUPTS ARE MASKED AND THE
000777 *  INTERRUPT ALLOCATION TABLE IS CLEARED.
000778 *
000779          REP      60
000780 *
000781 *
000782 INT.INIT       EQU      *
000783          SEI          ;DISABLE INTERRUPTS
000784          LDA          #>ZPGSTART      ;SET UP MIH
000785          STA          ZPGSTACK        ; ZERO PAGE STACK POINTER

```

```

000786 *
000787     LDA     E.REG           ;SELECT $C000 I/O SPACE
000788     PHA     ; AND SET 1 MHZ
000789     ORA     #BITON7+BITON6
000790     STA     E.REG
000791 *
000792     STA     ACIASTAT       ;RESET ACIA
000793 *
000794     LDA     #$FF           ;SET UP 6522 D
000795     STA     D.DDRB
000796     STA     D.DDRA
000797     LDA     #$00
000798     STA     D.ACR
000799     LDA     #$76
000800     STA     D.PCR
000801     LDA     #$7F
000802     STA     D.IFR
000803     STA     D.IER
000804     LDA     #$82
000805     STA     D.IER
000806 *
000807     LDA     #$3F           ;SET UP 6522 E
000808     STA     E.DDRB
000809     LDA     #$0F
000810     STA     E.DDRA
000811     LDA     #$00
000812     STA     E.ACR
000813     LDA     #$63
000814     STA     E.PCR
000815     LDA     #$7F
000816     STA     E.IFR
000817     STA     E.IER
000818 *
000819     LDA     #$FF
000820     STA     E.IORB       ;SOUND PORT
000821     BIT     $C0D8       ;DISABLE GRAPHICS SCROLL
000822     BIT     $C0DA       ;DISABLE CHARACTER DOWNLOAD
000823     BIT     $C0DC       ;DISABLE ENSEL
000824     BIT     $C0DE       ;SET ENSIO FOR INPUT
000825 *
000826     PLA
000827     STA     E.REG       ;RESTORE E REGISTER
000828 *
000829     LDA     #FALSE
000830     STA     NMIFLAG      ;CLEAR NMI WAIT FLAG
000831     LDY     #>SIRTBLsiz-1
000832     INTI010 STA     SIRTABLE,Y ; ALLOCATION TABLE
000833     DEY
000834     BPL     INTI010
000835     LDA     #TRUE

```

```

000836          STA          SIRTABLE+$0A          ;LOCK DOWN ANY SLOT SIR
000837 *
000838          LDX          #$05
000839 INTI020      LDA          RAMVECT,X          ;SET UP VECTORS
000840          STA          $FFFA,X          ; AT $FFFA - $FFFF
000841          LDA          RAMJMPS,X          ;SET UP JMP INSTRUCTIONS
000842          STA          $FFCA,X          ; AT $FFCA - $FFCF
000843          DEX
000844          BPL          INTI020
000845          RTS
000846 *
000847 RAMVECT      DW          NMI.RCVR
000848          DW          COLDSTRT
000849          DW          IRQ.RCVR
000850 RAMJMPS      JMP          NMI.RCVR
000851          JMP          IRQ.RCVR
000852          SBTL         "EVENT QUEUE INITIALIZATION"
000853          REP          60
000854 *
000855 * THIS SUBROUTINE INITIALIZES THE EVENT QUEUE. ALL ENTRIES
000856 * ARE CLEARED AND LINKED INTO THE FREE LIST. THE ACTIVE
000857 * LIST IS EMPTY.
000858 *
000859          REP          60
000860 *
000861 *
000862 EVQ.INIT      EQU          *
000863 *
000864 * CLEAR ALL ENTRIES
000865 *
000866          LDY          #>EVQ.LEN
000867          LDA          #0
000868 EVQI010      STA          EV.QUEUE-1,Y
000869          DEY
000870          BNE          EVQI010
000871 *
000872 * SET UP FREE LIST
000873 *
000874          LDX          #>EVQ.CNT-2
000875          LDA          #>EVQ.SIZ
000876          STA          EVQ.FREE
000877 EVQI020      TAY
000878          CLC
000879          ADC          #>EVQ.SIZ
000880          STA          EVQ.LINK,Y
000881          DEX
000882          BNE          EVQI020
000883          RTS
000884          SBTL         "PSEUDO CLOCK INITIALIZATION"
000885          REP          60

```

```

000886 *
000887 * THIS SUBROUTINE INITIALIZES THE PSEUDO CLOCK. IF THE
000888 * RAM BEHIND THE "D" 6522 HAS THE PROPER CHECKSUM, IT
000889 * IS USED TO INITIALIZE THE PSEUDO CLOCK. OTHERWISE,
000890 * THE PSEUDO CLOCK IS SET TO ZERO.
000891 *
000892 * (ADDED 23 OCT 81)
000893 * BOTH THE CLOCK AND PSEUDO CLOCK ARE
000894 * ARE NOW INITIALIZED
000895 *
000896         REP         60
000897 *
000898 PCLK         EQU         $F0
000899 CKSUM        EQU         $F2
000900 CLKICR       EQU         $11           ; CLOCK INTERRUPT CONTROL REG
000901 CLKSTBY     EQU         $16           ; CLOCK STANDBY INTERRUPT
000902 CLOCK       EQU         $C070
000903 *
000904 CLK.INIT     EQU         *
000905             LDA         #$D0
000906             STA         PCLK           ;POINT (PCLK) TO 8F:FFD0
000907             LDA         #$FF
000908             STA         PCLK+1
000909             LDA         #$8F
000910             STA         SXPAGE+PCLK+1
000911             LDA         #$A5
000912             STA         CKSUM        ;INITIALIZE CHECKSUM
000913 *
000914             LDY         #$00
000915 CLK010      LDA         (PCLK),Y      ;COPY SAVED CLOCK DATA
000916             STA         PCLOCK,Y     ; TO PSEUDO CLOCK
000917             EOR         CKSUM
000918             STA         CKSUM        ;UPDATE CHECKSUM
000919             INY
000920             CPY         #$0A
000921             BCC         CLK010
000922 *
000923             CMP         (PCLK),Y     ;TEST CHECKSUM
000924             BEQ         CLK030
000925 *
000926             LDA         #$00
000927 CLK020      DEY
000928             STA         PCLOCK,Y     ;ZERO PSEUDO CLOCK
000929             BNE         CLK020
000930 CLK030      LDA         E.REG
000931             PHA
000932             ORA         #$80         ; SET 1 MHZ
000933             STA         E.REG
000934             LDA         #$00
000935             LDY         Z.REG

```



```

000936         LDX      #CLKICR
000937         STX      Z.REG
000938         STA      CLOCK          ; DISABLE CLOCK INTERRUPTS
000939         LDX      #CLKSTBY
000940         STX      Z.REG
000941         STA      CLOCK          ; DISABLE STANDBY INTERRUPT
000942         STY      Z.REG
000943         PLA
000944         STA      E.REG
000945         RTS
000946         SBTL     "CHARACTER FILE MANAGER INITIALIZATION"
000947         REP      60
000948         *
000949         * CHAR FILE MANAGER INITIALIZATION ROUTINE
000950         *
000951         * CFMGR.INIT INITIALIZES ALL ENTRIES IN THE CFCB TABLE TO
000952         * THE "FREE" STATE.
000953         *
000954         REP      60
000955         *
000956 CFMGR.INIT   EQU      *
000957         LDA      #$80
000958         LDX      #CFCB.MAX-1
000959 CFINIT010    STA      CFCB.DEV,X
000960         DEX
000961         BPL      CFINIT010
000962         RTS
000963         SBTL     "DEVICE MANAGER INITIALIZATION"
000964         REP      60
000965         *
000966         * DEVICE MANAGER INITIALIZATION ROUTINE
000967         *
000968         * INITIALIZES THE SYSTEM DEVICE TABLE (SDT) BY WALKING THE
000969         * DEVICE INFORMATION BLOCK (DIB) LINKS.  CALLED BY SYSLDR.
000970         *
000971         REP      60
000972         *
000973 D.TPARMX     EQU      $C0
000974 REQCODE     EQU      D.TPARMX+$00
000975 DNUM       EQU      D.TPARMX+$01
000976 DNUM.TEMP  DS      1
000977         *
000978         *
000979 DMGR.INIT   EQU      *
000980         LDX      MAX.DNUM
000981         INC      MAX.DNUM          ; MAX.DNUM:=MAX DEV NUMBER IN SYSTEM+1
000982         STX      DNUM.TEMP
000983 DMI110     LDA      #8          ; INITIALIZE ALL DEVICES IN SYSTEM (D.INIT)
000984         STA      REQCODE
000985         LDA      DNUM.TEMP

```

```

000986          STA          DNUM
000987          JSR          DMGR
000988          DEC          DNUM.TEMP
000989          BNE          DMI110
000990          RTS                      ; NORMAL EXIT
000991          SBTBL        "BUFFER MANAGER INITIALIZATION"
000992          REP          60
000993          *
000994          * BMGR.INIT
000995          *
000996          * THIS ROUTINE INITIALIZES THE BUFFER TABLE'S ENTRIES TO "FREE".
000997          * CALLED DURING SYSTEM BOOT.
000998          *
000999          REP          60
001000          *
001001          BMGR.INIT    EQU          *
001002          LDA          #$FF                      ; USED WHEN FINDING LOWEST BUFFER IN TBL (BUFCOMPACT)
001003          STA          XBYTE.T
001004          *
001005          LDX          #BUF.CNT-1
001006          LDA          #$80
001007          BUFI010     STA          PGCT.T,X      ;SET ALL ENTRIES "FREE"
001008          DEX
001009          BNE          BUFI010
001010          *
001011          STX          BUFREF                    ;ZERO COUNT BYTE IN BUFFER REFERENCE TABLE
001012          *
001013          CLC
001014          RTS
001015          SBTBL        "MEMORY MANAGER INITIALIZATION"
001016          REP          60
001017          *
001018          * MMGR.INIT
001019          *
001020          * THIS ROUTINE INITIALIZES THE MEMORY MANAGER'S SEGMENT TABLE
001021          * TO FREE ENTRIES, AND DETERMINES THE MEMORY SIZE OF THE
001022          * MACHINE (96K,128K,160K,192K,224K,256K,...,512K IN 32K STEPS).
001023          *
001024          REP          60
001025          *
001026          MMGR.INIT    EQU          *
001027          *
001028          * INIT SEGMENT TABLE
001029          *
001030          LDA          #0
001031          STA          ST.ENTRY
001032          LDA          #$81
001033          STA          ST.FREE
001034          *
001035          LDY          #ST.CNT-1

```

```

001036          LDA      #$80          ; SET LAST LINK TO NULL
001037          STA      ST.FLINK,Y
001038 MEMI010      TYA
001039          ORA      #$80
001040          DEY
001041          STA      ST.FLINK,Y
001042          BNE      MEMI010
001043 *
001044 * COMPUTE VIRTUAL LIMIT FROM MEMORY SIZE
001045 * VRT.LIM := NUMBER OF PAGES IN BANK SWITCHED MEMORY - 1
001046 *          := (MEMSIZ-2)*64 - 1
001047 *          := (MEMSIZ-4)*64 + 127
001048 *
001049          SEC
001050          LDA      MEMSIZE
001051          SBC      #4
001052          BCC      MEMI.ERR
001053          LSR      A
001054          LSR      A
001055          STA      VRT.LIM+1
001056          LDA      #$FE
001057          ROR      A
001058          STA      VRT.LIM
001059          CLC
001060          RTS          ; NORMAL EXIT
001061 *
001062 MEMI.ERR      LDA      #MEM2SML      ; FATAL ERR - MEM < 64K
001063          JSR      SYSDEATH
001064          PAGE
001065          REP      60
001066 *
001067 * BLOCK FILE MANAGER INITIALIZATION
001068 *
001069          REP      60
001070 *
001071 SISTER      EQU      $1400          ;BFM XPAGE
001072 BFM.INIT     EQU      *
001073          LDA      #BFMFCB1          ; ADDRESS OF PAGE 1 OF FCB
001074          STA      >FCBZPP+1
001075          LDA      #BFMFCB2          ; AND PAGE 2
001076          STA      >FCBZPP+3
001077          LDA      #0
001078          STA      >FCBZPP          ; FCB PAGE ALIGNED
001079          STA      >FCBZPP+2
001080          STA      SISTER+FCBZPP+1    ; PREPARE PART OF EXTEND BYTE
001081          STA      SISTER+FCBZPP+3
001082          TAY          ; MAKE ZERO INTO INDEX
001083 CLRBUFFS     EQU      *
001084          STA      PATHBUF,Y          ; PATHNAME BUFFER PAGE
001085          STA      VCB,Y             ; VOLUME CONTROL BLOCK PAGE

```

```

001086          STA      (>FCBZPP),Y          ; BOTH FILE CONTROL BLOCK PAGES
001087          STA      (>FCBZPP+2),Y
001088          INY
001089          BNE      CLRBUFFS
001090          LDX      #$3F                    ; SIZE OF MY ZERO PAGE STUFF
001091 CLRZWRK    STA      0,X                    ; ZERO PAGE ZEROED
001092          STA      WORKSPC,X
001093          DEX
001094          BPL      CLRZWRK
001095          LDA      #<PATHBUF
001096          STA      PFIXPTR+1
001097          LDA      #BFMFCB1
001098          STA      FCBADDRH
001099          LDA      #BMAPAGE                ; BIT MAP A PAGE NUMBER
001100          STA      BMAMADR
001101          LDA      #BMBPAGE                ; BIT MAP B PAGE NUMBER
001102          STA      BMBMADR
001103          CLC
001104          RTS
001105 *
001106          LST      ON
001107 ZZEND      EQU      *
001108 ZZLEN      EQU      ZZEND-ZZORG
001109          IFNE     ZZLEN-LENINIT
001110          FAIL     2,"SOSORG                FILE IS INCORRECT FOR INIT"
001111          FIN
001112
001113 *****
001114 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: INIT.SRC
001115 *****
001116
001117

```

```

001118 =====
001119 DOCUMENT :SOS1.3.1of5.ONE:SOS.IPL.SRC1.TEXT
001120 =====
001121
001122 *****
001123 * APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC1
001124 *****
001125 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
001126
001127         SBTL          "SOS 1.1  INTRPTS. & PROC. LAUNCH"
001128         REL
001129         INCLUDE      SOSORG,6,1,254
001130         ORG          ORGIPL
001131 ZZORG         EQU          *
001132         MSB          OFF
001133         REP          60
001134 *           COPYRIGHT (C) APPLE COMPUTER INC. 1980
001135 *           ALL RIGHTS RESERVED
001136         REP          60
001137 *
001138 * THIS MODULE IS RESPONSIBLE FOR FIELDING ALL INTERRUPTS
001139 * AND RELAUNCHING THE INTERRUPTED CODE AFTER THE INTERRUPTS
001140 * HAVE BEEN PROCESSED.  THE MAJOR FUNCTIONAL AREAS ARE:
001141 *
001142 *     GENERAL INTERRUPT RECEIVER
001143 *     NMI INTERRUPT RECEIVER
001144 *     DISPATCHER
001145 *     INTERRUPT ALLOCATION & DEALLOCATION
001146 *     EVENT QUEUE MANAGER
001147 *     TABLE INITIALIZATION
001148 *
001149         REP          60
001150 *
001151 * SUBROUTINE ENTRY POINTS
001152 *
001153         ENTRY        IRQ.RCVR          ;GENERAL INTERRUPT RECEIVER
001154         ENTRY        NMI.RCVR          ;NON-MASKABLE INTRPT RCVR
001155         ENTRY        DISPATCH          ;DISPATCHER
001156         ENTRY        ALLOCSIR         ;SIR ALLOCATION
001157         ENTRY        DEALCSIR         ;SIR DEALLOCATION
001158         ENTRY        SELC800          ;SELECT I/O EXPANSION ROM
001159         ENTRY        NMIDSBL          ;DISABLE NMI
001160         ENTRY        NMIEENBL         ;ENABLE NMI
001161         ENTRY        NMIDBUG          ;NMI DEBUG ENTRY
001162         ENTRY        NMICONT          ;NMI DEBUG CONTINUATION
001163         ENTRY        QUEEVENT         ;QUEUE AN EVENT
001164 *
001165 * EXTERNAL SUBROUTINES & DATA
001166 *

```

```

001167          EXTRN   SCMGR
001168          EXTRN   CHKBUF
001169 *
001170 *  SYSTEM DEATH ERRORS
001171 *
001172          EXTRN   SYSDEATH
001173          EXTRN   BADBRK
001174          EXTRN   BADINT1
001175          EXTRN   BADINT2
001176          EXTRN   NMIHANG
001177          EXTRN   EVQOVFL
001178          EXTRN   STKOVFL
001179 *
001180 *  LINKAGE DATA FOR INITIALIZATION ROUTINES
001181 *
001182          ENTRY   EV.QUEUE
001183          ENTRY   EVQ.CNT
001184          ENTRY   EVQ.SIZ
001185          ENTRY   EVQ.LEN
001186          ENTRY   EVQ.FREE
001187          ENTRY   EVQ.LINK
001188          ENTRY   SIRTABLE
001189          ENTRY   SIRTBLSIZ
001190          ENTRY   ZPGSTACK
001191          ENTRY   ZPGSTART
001192 *
001193 *  SYSGLOB DATA
001194 *
001195          EXTRN   SERR
001196          EXTRN   CEVPRI           ;CALLER'S EVENT PRIORITY
001197          EXTRN   SYSBANK        ;SYSTEM BANK
001198          EXTRN   KYBDNMI
001199          EXTRN   NMISPSV
001200          EXTRN   NMIFLAG       ;NMI PENDING FLAG
001201          EXTRN   SCRNMODE     ;CURRENT SCREEN MODE
001202          EXTRN   SIRTEMP      ;FOR ALLOCSIR & DEALCSIR
001203          EXTRN   SIRARGSIZ
001204          EXTRN   IRQCNTR       ;FLASE IRQ COUNTER
001205          EXTRN   NMICNTR      ;TWO BYTE COUNTER
001206          EXTRN   QEVTEMP
001207          EXTRN   QEV.THIS
001208          EXTRN   QEV.LAST
001209          EXTRN   BACKMASK
001210 *
001211 *  CONSTANT DECLARATIONS
001212 *
001213 FALSE          EQU          $00
001214 BITON0         EQU          $01
001215 BITON1         EQU          $02
001216 BITON2         EQU          $04

```

```

001217 BITON4      EQU      $10
001218 BITON5      EQU      $20
001219 BITON6      EQU      $40
001220 BITON7      EQU      $80
001221 BITOFF3     EQU      $F7
001222 BITOFF4     EQU      $EF
001223 BITOFF5     EQU      $DF
001224 BITOFF6     EQU      $BF
001225 BITOFF7     EQU      $7F
001226 BACKBIT    EQU      $20          ; BACKUP BIT MASK
001227 *
001228 * SYSTEM CONTROL REGISTERS
001229 *
001230 B.REG        EQU      $FFEF      ;BANK REGISTER
001231 E.REG        EQU      $FFDF      ;ENVIRONMENT REGISTER
001232 Z.REG        EQU      $FFD0      ;ZERO PAGE REGISTER
001233 *
001234 * 6522 REGISTERS
001235 *
001236 D.IFR        EQU      $FFDD
001237 D.IER        EQU      $FFDE
001238 E.IORB       EQU      $FFE0
001239 E.IFR        EQU      $FFED
001240 E.IER        EQU      $FFEE
001241 E.IORA      EQU      $FFEF
001242          PAGE
001243 *
001244 * REGISTER PRESERVATION EQUATES
001245 * FOR USE DURING INTERRUPT PROCESSING
001246 *
001247 A.SAVE       EQU      $103
001248 S.SAVE       EQU      $104
001249 SP.SAVE     EQU      $1FF
001250 E.SAVE       EQU      $1FE
001251 Z.SAVE       EQU      $1FD
001252 B.SAVE       EQU      $1FC
001253 EXPNSLOT    DFB      $00          ;CURRENT I/O EXPANSION SLOT
001254 *
001255 * STATUS LOCATIONS FOR INTERRUPT POLLING
001256 *
001257 ACIASTAT     EQU      $C0F1
001258 ANYSLOT     DFB      BITON1
001259 SLOT1       EQU      $C065
001260 SLOT2       EQU      $C064
001261 SLOT3       DFB      BITON5
001262 SLOT4       DFB      BITON4
001263 *
001264 * INTERRUPT ZERO PAGE STORAGE & EQUATES
001265 *
001266 SIRARGS      EQU      $F9          ;AND $FA

```

```

001267 QEVARGS      EQU      $FB      ;AND $FC
001268 IRQADDR     EQU      $FD      ;AND $FE
001269 ZPGSP        EQU      $FF
001270 ZPGSTART      EQU      $F8
001271 ZPGSTOP       EQU      $28
001272 ZPGSPACE     EQU      $20
001273 ZPGSTACK     DFB      ZPGSTART
001274 *
001275 * SYSTEM INTERNAL RESOURCE
001276 * TABLE STORAGE AND EQUATES
001277 *
001278 SIRTBLSIZ      EQU      $18
001279 SIRTABLE       DS      SIRTBLSIZ
001280 SIRADR.L       DS      SIRTBLSIZ
001281 NMIADR.L       DS      1          ;MUST PRECEED SIRADR.H
001282 SIRADR.H       DS      SIRTBLSIZ
001283 SIRADR.B       DS      SIRTBLSIZ
001284 *
001285 * EVENT QUEUE STORAGE AND EQUATES
001286 *
001287 EVQ.SIZ        EQU      6          ;ENTRY SIZE
001288 EVQ.CNT        EQU      $07        ;ENTRY COUNT
001289 EVQ.LEN        EQU      $2A        ;(EVQ.SIZ*EVQ.CNT)
001290 EV.QUEUE       DS      EVQ.LEN
001291 EVQ.FREE       EQU      EV.QUEUE+2 ;FIRST FREE ENTRY INDEX
001292 EVQ.LINK       EQU      EV.QUEUE+0 ;NEXT ACTIVE ENTRY INDEX
001293 EVQ.PRI        EQU      EV.QUEUE+1 ;EVENT PRIORITY
001294 EVQ.ID         EQU      EV.QUEUE+2 ;EVENT IDENTIFICATION
001295 EVQ.ADR.L      EQU      EV.QUEUE+3 ;EVENT ADDRESS: LOW BYTE
001296 EVQ.ADR.H     EQU      EV.QUEUE+4 ;EVENT ADDRESS: HIGH BYTE
001297 EVQ.BANK      EQU      EV.QUEUE+5 ;EVENT ADDRESS: BANK
001298             SBTL      "GENERAL INTERRUPT RECEIVER"
001299             REP      60
001300 *
001301 * THIS IS THE GENERAL INTERRUPT RECEIVER. WHEN AN
001302 * INTERRUPT OCCURS, THE CPU PASSES CONTROL TO THE GIR
001303 * THROUGH THE IRQ VECTOR. THE GIR IS RESPONSIBLE FOR
001304 * SAVING THE CURRENT ENVIRONMENT, SETTING UP THE SOS
001305 * ENVIRONMENT, AND CALLING THE APPROPRIATE CODE MODULE.
001306 * IF THE INTERRUPT WAS CAUSED BY A BRK, THE GIR CALLS
001307 * THE SYSTEM CALL MANAGER. OTHERWISE, THE GIR POLLS THE
001308 * I/O DEVICES AND CALLS THE APPROPRIATE MASTER INTERRUPT
001309 * HANDLER. WHEN THE SCM OR MIH RETURNS, THE GIR PASSES
001310 * CONTROL TO THE DISPATCHER.
001311 *
001312             REP      60
001313 *
001314 IRQ.RCVR       EQU      *
001315 *
001316 * SAVE CPU REGISTERS A, X, & Y ON CURRENT STACK

```



```

001317 *
001318             PHA
001319             TXA
001320             PHA
001321             TYA
001322             PHA
001323 *
001324 * CHECK FOR STACK OVERFLOW AND
001325 * SAVE INTERRUPTED STATUS IN Y REGISTER.
001326 *
001327             TSX
001328             CPX             #$FA
001329             BCC             GIR005
001330             LDA             #>STKOVFL
001331             JSR             SYSDEATH
001332 GIR005      LDY             S.SAVE,X
001333 *
001334 * SET UP INTERRUPT ENVIRONMENT:
001335 *     BINARY ARITHMETIC, 2 MHZ, I/O ENABLED,
001336 *     RAM WRITE ENABLED, PRIMARY STACK,
001337 *     AND $F000 RAM SELECTED. PRESERVE
001338 *     USER STATE OF SCREEN AND RESET LOCK.
001339 *
001340             CLD
001341             LDA             E.REG
001342             TAX
001343             AND             #BITON5+BITON4
001344             ORA             #BITON6+BITON2
001345             STA             E.REG
001346 *
001347 * IF NOT ALREADY ON PRIMARY STACK, SAVE USER'S STACK
001348 * POINTER AND SET UP SOS STACK POINTER.
001349 *
001350             TXA
001351             AND             #BITON2
001352             BNE             GIR010
001353             TXA
001354             TSX
001355             STX             SP.SAVE
001356             LDX             #>E.SAVE
001357             TXS
001358             TAX
001359 *
001360 * SAVE E, Z, B, & I/O EXPANSION SLOT ON SOS STACK
001361 * IF BRK THEN CALL SCMGR ELSE POLL I/O DEVICES
001362 *
001363 GIR010      TXA
001364             PHA
001365             LDA             Z.REG
001366             PHA

```

```

001367          LDA      B.REG
001368          PHA
001369          LDA      EXPNSLOT
001370          PHA
001371          BIT      $CFFF
001372          BIT      $C020          ;RESET I/O SPACE
001373          LDA      #$00
001374          STA      EXPNSLOT
001375          TYA
001376          AND      #BITON4
001377          BEQ      POLL.IO
001378          *
001379          * CALL SYSTEM CALL MANAGER; ON RETURN, PUT ERROR CODE IN
001380          * USER'S A REGISTER AND SET RETURN STATUS, THEN DISPATCH.
001381          *
001382          TSX          ;CHECK FOR
001383          CPX      #>B.SAVE-2      ; REENRANT
001384          BEQ      GIR020          ; SYSTEM CALL
001385          LDA      #>BADBRK
001386          JSR      SYSDEATH
001387 GIR020      LDA      E.REG          ;SELECT $C000 RAM
001388          AND      #BITOFF6
001389          STA      E.REG
001390          CLI          ;ENABLE INTERRUPTS
001391          JSR      SCMGR          ;CALL THE SYSTEM CALL MGR
001392          LDA      #BACKBIT        ; GET THE MASK
001393          STA      BACKMASK        ; SET IT IN SYSGLOB
001394          JSR      CHKBUF
001395          SEI
001396          LDX      SP.SAVE
001397          LDA      Z.SAVE
001398          EOR      #BITON0        ;SET ZERO PAGE TO
001399          STA      Z.REG          ; CALLER'S STACK
001400          LDA      SERR
001401          STA      >A.SAVE,X
001402          PHP
001403          LDA      >S.SAVE,X
001404          AND      #$7D
001405          STA      >S.SAVE,X
001406          PLA
001407          AND      #$82
001408          ORA      >S.SAVE,X
001409          STA      >S.SAVE,X
001410          JMP      DISPATCH
001411          PAGE
001412          *
001413          * SET INTERRUPT ZERO PAGE AND SOS BANK
001414          * THEN POLL I/O DEVICES
001415          *
001416 POLL.IO      BIT      E.IORA          ;VERIFY THAT 'IRQ IS LOW

```

```

001417          BPL          PIO006
001418          INC          IRQCNTR          ;BUMP FALSE IRQ COUNTER
001419          BNE          PIO004
001420          INC          IRQCNTR+1
001421 PIO004      JMP          DISPATCH
001422 PIO006      LDA          #0          ;SET INTERRUPT ZERO PAGE
001423          STA          Z.REG
001424          LDA          E.REG
001425          ORA          #BITON7          ;FORCE 1 MHZ FOR
001426          STA          E.REG          ; READING ACIA STATUS
001427          AND          #BITOFF7
001428          LDX          #$01
001429          LDY          ACIASTAT          ;ANY INTERRUPT ON ACIA?
001430          STA          E.REG
001431          BMI          PIO070
001432          LDA          E.IFR          ;ANY INTERRUPT ON E-6522?
001433          BPL          PIO020          ; NO
001434          AND          E.IER
001435          LDY          #7
001436          LDX          #$02
001437 PIO010     LSR          A          ;CHECK FLAG BITS
001438          BCS          PIO070
001439          INX
001440          DEY
001441          BNE          PIO010
001442          BEQ          PIO035
001443 PIO020     LDA          D.IFR          ;ANY INTERRUPT ON D-6522?
001444          BPL          PIO035
001445          AND          D.IER
001446          BIT          ANYSLOT          ;ANY SLOT INTERRUPT?
001447          BNE          PIO040          ; YES
001448          LDY          #7
001449          LDX          #$09
001450 PIO030     LSR          A          ;CHECK FLAG BITS
001451          BCS          PIO070
001452          INX
001453          DEY
001454          BNE          PIO030
001455 PIO035     LDX          #$10          ;INTERRUPT NOT FOUND
001456          BNE          PIO050
001457 PIO040     LDX          #$11
001458          BIT          SLOT1          ;SLOT 1?
001459          BPL          PIO070
001460          INX
001461          BIT          SLOT2          ;SLOT 2?
001462          BPL          PIO070
001463          LDA          E.IORA
001464          INX
001465          BIT          SLOT3          ;SLOT 3?
001466          BEQ          PIO070

```

```

001467             INX
001468             BIT          SLOT4             ;SLOT 4?
001469             BEQ          PIO070
001470             LDX          #$0A
001471 *
001472 *  BAD INTERRUPT -- SYSTEM DEATH
001473 *
001474 PIO050         LDA          #>BADINT1         ;INTERRUPT NOT FOUND
001475             JSR          SYSDEATH
001476 PIO060         LDA          #>BADINT2         ;BAD ZERO PAGE ALLOCATION
001477             JSR          SYSDEATH
001478 *
001479 *  INTERRUPTING DEVICE FOUND
001480 *    ALLOCATE ZERO PAGE AND CALL MASTER INTERRUPT HANDLER
001481 *
001482 *  NOTE:
001483 *    SINCE READING THE ACIA'S STATUS REGISTER RESETS THE
001484 *    DSR AND DCD BITS, THE STATUS READ BY THE POLLING
001485 *    ROUTINE MUST BE PASSED TO THE INTERRUPT HANDLER;
001486 *    THE Y REGISTER HAS BEEN SELECTED FOR THIS PURPOSE.
001487 *    THE CURRENT IMPLEMENTATION DOES NOT USE Y IN CALLING
001488 *    THE INTERRUPT HANDLER.  IF SUBSEQUENT REVISIONS
001489 *    NEED TO USE Y, THE STATUS MUST BE PRESERVED AND
001490 *    RESTORED BEFORE CALLING THE INTERRUPT HANDLER.
001491 *
001492 CALLMIH         JMP          (IRQADDR)
001493 *
001494 PIO070         LDA          SIRTABLE,X         ;INTERRUPT ALLOCATED?
001495             BPL          PIO050             ; NO
001496             LDA          SIRADR.L,X         ;GET INTERRUPT ADDRESS
001497             STA          IRQADDR
001498             ORA          SIRADR.H,X         ;CHECK FOR ADDRESS = $00
001499             BEQ          PIO050             ; BAD ADDRESS
001500             LDA          SIRADR.H,X
001501             STA          IRQADDR+1
001502             LDA          SIRADR.B,X
001503             STA          B.REG
001504             LDA          ZPGSTACK         ;ALLOCATE MIH ZERO PAGE
001505             CMP          #ZPGSTOP+ZPGSPACE
001506             BCC          PIO060             ;TOO MANY NESTED INTERRUPTS
001507             SBC          #ZPGSPACE
001508             STA          ZPGSTACK
001509             STA          ZPGSP
001510             TAX
001511             JSR          CALLMIH         ;CALL INTERRUPT HANDLER
001512             SEI
001513             LDA          #$00
001514             STA          Z.REG
001515             CLC
001516             LDA          ZPGSTACK         ;DEALLOCATE MIH ZERO PAGE

```

```

001517         ADC      #ZPGSPACE
001518         STA      ZPGSTACK
001519         STA      ZPGSP
001520         LDA      #BITON1
001521         STA      D.IFR          ;CLEAR ANY SLOT INTERRUPT
001522         JMP      DISPATCH
001523         SBTL     "NON-MASKABLE INTERRUPT RECEIVER"
001524         REP      60
001525 *
001526 * THIS IS THE NON-MASKABLE INTERRUPT RECEIVER.  WHEN AN
001527 * NMI OCCURS, THE CPU PASSES CONTROL TO THE NMI RECEIVER
001528 * THROUGH THE NMI VECTOR.  THE OPERATION OF THE NMI
001529 * RECEIVER IS ESSENTIALLY THE SAME AS THE GIR EXCEPT
001530 * THAT IT IS NOT CONCERNED WITH BRK, AND THE ONLY VALID
001531 * SOURCE OF AN NMI IS THE KEYBOARD OR THE I/O DEVICE THAT
001532 * HAS ALLOCATED THE NMI RESOURCE.
001533 *
001534         REP      60
001535 *
001536 *
001537 NMI.RCVR     EQU      *
001538 *
001539 * SAVE CPU REGISTERS A, X, & Y ON CURRENT STACK
001540 *
001541         PHA
001542         TXA
001543         PHA
001544         TYA
001545         PHA
001546 *
001547 * CHECK FOR STACK OVERFLOW
001548 *
001549         TSX
001550         CPX      #$FA
001551         BCC      NMI005
001552         LDA      #>STKOVFL
001553         JSR      SYSDEATH
001554 *
001555 * SET UP INTERRUPT ENVIRONMENT:
001556 *   BINARY ARITHMETIC, 2 MHZ, I/O ENABLED,
001557 *   RAM WRITE ENABLED, PRIMARY STACK,
001558 *   AND $F000 RAM SELECTED.  PRESERVE
001559 *   USER STATE OF SCREEN AND RESET LOCK.
001560 *
001561 NMI005      CLD
001562         LDA      E.REG
001563         TAX
001564         AND      #BITON5+BITON4
001565         ORA      #BITON6+BITON2
001566         STA      E.REG

```

```

001567 *
001568 * IF NOT ALREADY ON PRIMARY STACK, SAVE USER'S
001569 * STACK POINTER AND SET UP SOS STACK POINTER.
001570 *
001571 TXA
001572 AND #BITON2
001573 BNE NMI010
001574 TXA
001575 TSX
001576 STX SP.SAVE
001577 LDX #>E.SAVE
001578 TXS
001579 TAX
001580 *
001581 * SAVE SYSTEM CONTROL REGISTERS E, Z, & B ON SOS STACK
001582 *
001583 NMI010 TXA
001584 PHA
001585 LDA Z.REG
001586 PHA
001587 LDA B.REG
001588 PHA
001589 LDA EXPNSLOT
001590 PHA
001591 BIT $CFFF
001592 BIT $C020 ;RESET I/O SPACE
001593 LDA #$00
001594 STA EXPNSLOT
001595 *
001596 * SET INTERRUPT ZERO PAGE
001597 *
001598 LDA #0
001599 STA Z.REG
001600 *
001601 * SEE IF NMI IS FROM KEYBOARD OR I/O DEVICE
001602 *
001603 LDA E.IORB
001604 BMI NMI030
001605 *
001606 * NMI IS FROM I/O DEVICE
001607 *
001608 LDA SIRTABLE ;NMI ALLOCATED?
001609 BPL NMI020
001610 JSR CALLNMI
001611 SEI
001612 JMP DISPATCH
001613 CALLNMI LDA SIRADR.L
001614 STA NMIADR.L
001615 LDA SIRADR.B
001616 STA B.REG

```

```

001617             JMP          (NMIADR.L)
001618 *
001619 *   BAD INTERRUPT -- SYSTEM DEATH
001620 *
001621 NMI020         LDA          #>BADINT1           ;NMI NOT ALLOCATED
001622             JSR          SYSDEATH
001623 *
001624 *   NMI IS FROM THE KEYBOARD
001625 *
001626 NMI030         LDA          SYSBANK
001627             STA          B.REG
001628             JSR          KYBDNMI
001629             SEI
001630             JMP          DISPATCH
001631             SBTL         "DISPATCHER"
001632             REP          60
001633 *
001634 *   THIS IS THE DISPATCHER.  UPON COMPLETION, ALL SOS CALLS
001635 *   AND INTERRUPT HANDLERS RETURN CONTROL TO THE DISPATCHER.
001636 *   ITS PURPOSE IS TO SET UP THE APPROPRIATE ENVIRONMENT AND
001637 *   PASS CONTROL TO WHATEVER CODE SHOULD RUN NEXT.
001638 *
001639 *   WHEN SOS IS INTERRUPTED, CONTROL ALWAYS RETURNS TO THE
001640 *   INTERRUPTED CODE.  HOWEVER, WHEN THE USER IS INTERRUPTED,
001641 *   BY EITHER A SOS CALL OR AN INTERRUPT, THE DISPATCHER
001642 *   MUST CHECK THE EVENT QUEUE.  IF THERE IS AN ACTIVE EVENT
001643 *   WITH A PRIORITY HIGHER THAN THE CURRENT EVENT FENCE,
001644 *   CONTROL IS PASSED TO THE EVENT CODE.  OTHERWISE, CONTROL
001645 *   RETURNS TO THE INTERRUPTED CODE.
001646 *
001647             REP          60
001648 *
001649 DISPATCH        EQU          *
001650 *
001651 *   DISABLE INTERRUPTS AND RESTORE
001652 *   SYSTEM CONTROL REGISTERS B & Z
001653 *
001654             SEI
001655             LDA          E.REG
001656             ORA          #BITON6           ;ENABLE I/O
001657             STA          E.REG
001658             PLA
001659             JSR          SELC800           ;RESTORE I/O SPACE
001660             PLA
001661             STA          B.REG
001662             PLA
001663             STA          Z.REG
001664 *
001665 *   CHECK SAVED ENVIRONMENT REGISTER
001666 *   IF RETURNING TO PRIMARY STACK

```

```

001667 *      THEN RESTORE E REG AND RELAUNCH SOS
001668 *      ELSE RESET STACK POINTER & RESTORE E REG
001669 *
001670          PLA
001671          ORA          #BITON5          ;SET SCREEN STATE TO
001672          BIT          SCRNMODE        ;  CURRENT SCREEN MODE
001673          BMI          DSP005
001674          AND          #BITOFF5
001675 DSP005      TAY
001676          AND          #BITON2
001677          BEQ          DSP010
001678          STY          E.REG
001679          BNE          DSP030
001680 DSP010     PLA
001681          TAX
001682          TXS
001683          STY          E.REG
001684 *
001685 *      CHECK FOR ACTIVE EVENT WITH PRIORITY > FENCE
001686 *
001687 DSP020     LDA          CEVPRI
001688          LDX          EVQ.LINK
001689          CMP          EVQ.PRI,X
001690          BCS          DSP030
001691 *
001692 *      PROCESS ACTIVE EVENT TRAP
001693 *      SAVE E, Z, B, & CALLER'S PRIORITY ON STACK THEN CALL
001694 *      EVENT. UPON RETURN, RESTORE PRIORITY, B, Z, & E THEN
001695 *      CHECK FOR MORE EVENTS.
001696 *
001697          LDA          E.REG
001698          PHA
001699          LDA          Z.REG
001700          PHA
001701          LDA          B.REG
001702          PHA
001703          LDA          CEVPRI
001704          PHA
001705          JSR          DO.EVENT
001706          SEI
001707          PLA
001708          STA          CEVPRI
001709          PLA
001710          STA          B.REG
001711          PLA
001712          STA          Z.REG
001713          PLA
001714          ORA          #BITON5          ;SET SCREEN STATE TO
001715          BIT          SCRNMODE        ;  CURRENT SCREEN MODE
001716          BMI          DSP025

```



```

001717          AND          #BITOFF5
001718 DSP025      STA          E.REG
001719          JMP          DSP020
001720 *
001721 *  RESTORE CPU REGISTERS Y, X, & A AND LAUNCH
001722 *
001723 DSP030      PLA
001724          TAY
001725          PLA
001726          TAX
001727          PLA
001728          RTI
001729          PAGE
001730          REP          60
001731 *
001732 *  THIS SUBROUTINE CALLS THE HIGHEST PRIORITY ACTIVE EVENT.
001733 *  FIRST, IT DELINKS THE FIRST ENTRY ON THE ACTIVE LIST AND
001734 *  LINKS IT TO THE FREE LIST.  THEN, IT SETS UP THE BANK,
001735 *  ADDRESS, ID, & STATUS AND CALLS THE EVENT VIA AN RTI.
001736 *
001737          REP          60
001738 *
001739 DO.EVENT      EQU          *
001740 *
001741 *  WRITE ENABLE RAM
001742 *
001743          LDY          E.REG
001744          TYA
001745          AND          #BITOFF3
001746          STA          E.REG
001747 *
001748 *  DELINK ENTRY FROM ACTIVE LIST AND RELINK IT TO FREE LIST
001749 *
001750          LDX          EVQ.LINK
001751          LDA          EVQ.LINK,X
001752          STA          EVQ.LINK
001753          LDA          EVQ.FREE
001754          STA          EVQ.LINK,X
001755          STX          EVQ.FREE
001756 *
001757 *  SET FENCE TO EVENT PRIORITY THEN RESTORE E REG
001758 *
001759          LDA          EVQ.PRI,X
001760          STA          CEVPRI
001761          STY          E.REG
001762 *
001763 *  SET UP B, Z, E, ADDRESS, ID, & STATUS
001764 *
001765          LDA          EVQ.BANK,X
001766          STA          B.REG

```

```
001767      LDA      EVQ.ADRH,X
001768      PHA
001769      LDA      EVQ.ADRL,X
001770      PHA
001771      LDY      EVQ.ID,X
001772      PHP
001773      PLA
001774      AND      #$82
001775      PHA
001776      TYA
001777      RTI
001778
001779      CHN      IPL.SRC2
001780
001781 *****
001782 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC1
001783 *****
001784
001785
001786
```

```

001787 =====
001788 DOCUMENT :SOS1.3.1of5.ONE:SOS.IPL.SRC2.TEXT
001789 =====
001790
001791 *****
001792 * APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC2
001793 *****
001794 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
001795
001796             SBTL             "SYSTEM INTERNAL RESOURCES"
001797             REP              60
001798 *
001799 * SYSTEM INTERNAL RESOURCE NUMBERS
001800 *
001801 *
001802 * SIR RESOURCE
001803 *
001804 * 0 SOUND PORT / I/O NMI
001805 * 1 ACIA
001806 * 2 E.CA2 -- KEYBOARD
001807 * 3 E.CA1 -- CLOCK
001808 * 4 E.SR
001809 * 5 E.CB2 -- VBL +
001810 * 6 E.CB1 -- VBL -
001811 * 7 E.T2
001812 * 8 E.T1
001813 * 9 D.CA2 -- CSP INPUT FLAG / INPUT SWITCH 1
001814 * A D.CA1 -- ANY SLOT (RESERVED FOR SOS)
001815 * B D.SR -- CSP DATA REGISTER
001816 * C D.CB2 -- CSP DATA I/O / ENSIO
001817 * D D.CB1 -- CSP CLOCK / ENSEL / A/D SELECT / INPUT SW3
001818 * E D.T2
001819 * F D.T1
001820 * 10 DISK STEPPER / GRAPHICS SCROLL / CHARACTER DOWNLOAD
001821 * 11 SLOT 1
001822 * 12 SLOT 2
001823 * 13 SLOT 3
001824 * 14 SLOT 4
001825 * 15 (UNASSIGNED)
001826 * 16 (UNASSIGNED)
001827 * 17 (UNASSIGNED)
001828 *
001829             REP              60
001830             SBTL             "RESOURCE ALLOCATION & DEALLOCATION"
001831             REP              60
001832 *
001833 * RESOURCE ALLOCATION AND DEALLOCATION
001834 *
001835 * SIRS ARE ALLOCATED AND DEALLOCATED BY THE SUBROUTINES

```

```

001836 * 'ALLOCSIR' AND 'DEALCSIR'.  THE RESOURCE PARAMETERS ARE
001837 * PASSED IN A TABLE THAT CONTAINS ONE FIVE-BYTE ENTRY FOR
001838 * EACH SIR THAT IS TO BE ALLOCATED OR DEALLOCATED.  THE
001839 * TABLE ENTRY FORMAT IS SHOWN BELOW:
001840 *
001841 *           0           1           2           3           4
001842 *   +-----+-----+-----+-----+-----+
001843 *   | SIR # | ID   | ADR.L | ADR.H | ADR.B |
001844 *   +-----+-----+-----+-----+-----+
001845 *
001846 * SIR # -- SYSTEM INTERNAL RESOURCE NUMBER
001847 * ID   -- IDENTIFICATION BYTE
001848 *       SUPPLIED BY ALLOCSIR, CHECKED BY DEALCSIR
001849 * ADR  -- INTERRUPT ADDRESS (LOW, HIGH, BANK)
001850 *       ZERO IF NO INTERRUPT HANDLER
001851 *
001852 *
001853 * ALLOCSIR -- ALLOCATE SYSTEM INTERNAL RESOURCES
001854 *
001855 *   PARAMETERS:
001856 *   A:  NUMBER OF BYTES IN TABLE
001857 *   X:  TABLE ADDRESS (LOW BYTE)
001858 *   Y:  TABLE ADDRESS (HIGH BYTE)
001859 *
001860 *   NORMAL EXIT -- SIRS ALLOCATED
001861 *   CARRY:  CLEAR
001862 *   A, X, Y:  UNDEFINED
001863 *
001864 *   ERROR EXIT -- SIRS NOT ALLOCATED
001865 *   CARRY:  SET
001866 *   X:  SIR NUMBER
001867 *   A, Y:  UNDEFINED
001868 *
001869 *
001870 * DEALCSIR -- DEALLOCATE SYSTEM INTERNAL RESOURCES
001871 *
001872 *   PARAMETERS:
001873 *   A:  NUMBER OF BYTES IN TABLE
001874 *   X:  TABLE ADDRESS (LOW BYTE)
001875 *   Y:  TABLE ADDRESS (HIGH BYTE)
001876 *
001877 *   NORMAL EXIT -- SIRS DEALLOCATED
001878 *   CARRY:  CLEAR
001879 *   A, X, Y:  UNDEFINED
001880 *
001881 *   ERROR EXIT -- SIRS NOT DEALLOCATED
001882 *   CARRY:  SET
001883 *   X:  SIR NUMBER
001884 *   A, Y:  UNDEFINED
001885 *

```

```

001886          REP          60
001887          PAGE
001888      *
001889  IDBYTE          DFB          $81
001890      *
001891  ALLOCSIR        EQU          *
001892          CLC
001893          PHP
001894          SEI
001895          STA          SIRARGSIZ          ;SAVE TABLE SIZE
001896          LDA          E.REG
001897          STA          SIRTEMP
001898          ORA          #BITON2          ;FORCE PRIMARY STACK
001899          AND          #BITOFF3          ; AND WRITE ENABLE
001900          STA          E.REG
001901          LDA          SIRTEMP
001902          PHA
001903          LDA          Z.REG
001904          PHA
001905          LDA          #$00
001906          STA          Z.REG          ;SET ZERO PAGE := $00
001907          STX          SIRARGS
001908          STY          SIRARGS+1          ;SET POINTER TO TABLE
001909      *
001910          LDY          #$00
001911  ASIR010          LDA          (SIRARGS),Y          ;GET SIR NUMBER
001912          CMP          #SIRTBSIZ
001913          TAX
001914          BCS          ASIR020
001915          LDA          SIRTABLE,X          ;CHECK ALLOCATION
001916          BMI          ASIR020
001917          LDA          IDBYTE
001918          STA          SIRTABLE,X          ;ALLOCATE SIR
001919          INY
001920          STA          (SIRARGS),Y          ;RETURN ID BYTE
001921          INY
001922          LDA          (SIRARGS),Y
001923          STA          SIRADR.L,X          ;SAVE INTERRUPT ADDRESS
001924          INY
001925          LDA          (SIRARGS),Y
001926          STA          SIRADR.H,X
001927          INY
001928          LDA          (SIRARGS),Y
001929          STA          SIRADR.B,X
001930          INY
001931          CPY          SIRARGSIZ
001932          BCC          ASIR010
001933      *
001934          CLC
001935          INC          IDBYTE          ;BUMP ID BYTE

```

```

001936          BMI          SIREXIT
001937          LDA          #$81
001938          STA          IDBYTE
001939          BMI          SIREXIT
001940          *
001941 ASIR020      STX          SIRTEMP          ;SAVE BAD SIR NUMBER
001942 ASIR030      SEC
001943          TYA
001944          SBC          #5
001945          TAY
001946          BCC          ASIR040
001947          LDA          (SIRARGS),Y          ;GET SIR NUMBER
001948          TAX
001949          LDA          #FALSE
001950          STA          SIRTABLE,X          ;RELEASE ALLOCATED SIRs
001951          BEQ          ASIR030
001952          *
001953 ASIR040      LDX          SIRTEMP          ;RETURN BAD SIR
001954          SEC
001955          *
001956          *
001957          *
001958 SIREXIT      PLA
001959          STA          Z.REG          ;RESTORE Z REGISTER
001960          PLA
001961          STA          E.REG          ;RESTORE E REGISTER
001962          BCC          SIREXIT1
001963          PLA
001964          ORA          #BITON0
001965          PHA
001966 SIREXIT1    PLP
001967          RTS
001968          *
001969          *
001970          *
001971 DEALCSIR    EQU          *
001972          CLC
001973          PHP
001974          SEI
001975          STA          SIRARGSIZ          ;SAVE TABLE SIZE
001976          LDA          E.REG
001977          STA          SIRTEMP
001978          ORA          #BITON2          ;FORCE PRIMARY STACK
001979          AND          #BITOFF3          ; AND WRITE ENABLE
001980          STA          E.REG
001981          LDA          SIRTEMP
001982          PHA
001983          LDA          Z.REG
001984          PHA
001985          LDA          #$00

```

```

001986          STA      Z.REG          ;SET ZERO PAGE := $00
001987          STX      SIRARGS
001988          STY      SIRARGS+1      ;SET POINTER TO TABLE
001989 *
001990          LDY      #$00
001991 DSIR010     LDA      (SIRARGS),Y  ;GET SIR NUMBER
001992          TAX
001993          CPX      #SIRTBSIZ
001994          BCS      DSIR030
001995          INY
001996          LDA      SIRTABLE,X
001997          BPL      DSIR030          ;VERIFY ALLOCATION
001998          CMP      (SIRARGS),Y
001999          BNE      DSIR030
002000          INY
002001          INY
002002          INY
002003          INY
002004          CPY      SIRARGSIZ
002005          BCC      DSIR010
002006 *
002007          LDY      SIRARGSIZ
002008 DSIR020     SEC
002009          TYA
002010          SBC      #5
002011          TAY
002012          BCC      SIREXIT
002013          LDA      (SIRARGS),Y  ;GET SIR NUMBER
002014          TAX
002015          LDA      #FALSE
002016          STA      SIRTABLE,X
002017          BEQ      DSIR020
002018 *
002019 DSIR030     SEC
002020          BCS      SIREXIT
002021          SBTLE "SELECT I/O EXPANSION ROM"
002022          REP      60
002023 *
002024 * SUBROUTINE 'SEL800' IS CALLED TO SELECT THE C800 I/O EX-
002025 * PANSION ADDRESS SPACE FOR A PERIPHERAL SLOT. ON ENTRY,
002026 * THE SLOT NUMBER IS PASSED IN THE ACCUMULATOR. IF NO
002027 * ERROR OCCURS, CARRY IS CLEARED; OTHERWISE, CARRY IS SET
002028 * AND THE PREVIOUS SLOT REMAINS SELECTED.
002029 *
002030 * PARAMETERS:
002031 *   A:  SLOT NUMBER
002032 *
002033 * NORMAL EXIT -- NEW SLOT SELECTED
002034 *   CARRY: CLEAR
002035 *   A:  UNDEFINED

```

```

002036 *   X, Y:  UNCHANGED
002037 *
002038 *  ERROR EXIT -- SLOT NOT CHANGED
002039 *    CARRY:  SET
002040 *    A, X, Y:  UNCHANGED
002041 *
002042 *  WARNING !!!
002043 *    'SELC800' USES SELF-MODIFYING CODE!
002044 *
002045         REP          60
002046 *
002047 SELC800      EQU          *
002048         CMP          #$05          ;CHECK SLOT NUMBER
002049         BCS          SC8EXIT      ;  INVALID
002050         PHP
002051         SEI
002052         STA          EXPNSLOT
002053         ORA          #$C0          ;MAKE SLOT INTO $CN00
002054         STA          CNADDR+2      ;  AND MODIFY BIT ADDRESS
002055         BIT          $C020
002056         BIT          $CFFF          ;DESELECT PREVIOUS SLOT
002057 CNADDR     BIT          $C0FF      ;  AND SELECT CURRENT SLOT
002058         PLP
002059 SC8EXIT     RTS
002060         SBTL         "NMI DISABLE / ENABLE"
002061         REP          60
002062 *
002063 *  THE SUBROUTINES NMIDSBL AND NMIENBL ARE CALLED TO
002064 *  DISABLE AND ENABLE NMI, RESPECTIVELY.  THERE ARE NO
002065 *  INPUT PARAMETERS.  ON EXIT, THE REGISTERS ARE UN-
002066 *  DEFINED.  NMIDSBL CLEARS THE CARRY FLAG IF NMI WAS
002067 *  SUCCESSFULLY DISABLED; OTHERWISE, CARRY IS SET.
002068 *
002069         REP          60
002070 *
002071 NMIDSBL      EQU          *
002072         LDX          E.REG
002073         BIT          NMIFLAG
002074         BPL          NDS020
002075         TXA
002076         ORA          #BITON7
002077         STA          E.REG          ;SET 1MHZ
002078         LDA          #$00
002079         STA          NMICNTR
002080         STA          NMICNTR+1
002081 NDS010     BIT          NMIFLAG      ;NMI PENDING?
002082         BPL          NDS020          ;  NO
002083         INC          NMICNTR        ;BUMP NMI COUNTER
002084         BNE          NDS010        ;  AND RECHECK NMI FLAG
002085         INC          NMICNTR+1

```



```

002086          BNE      NDS010
002087          LDA      #>NMIHANG          ;CAN'T LOCK NMI
002088          JSR      SYSDEATH
002089 NDS020     TXA      ;GET E.REG
002090          AND      #BITOFF4          ;DISABLE NMI
002091          STA      E.REG
002092          RTS
002093 *
002094 *
002095 *
002096 NMIENBL     EQU      *
002097          LDA      E.REG
002098          ORA      #BITON4          ;ENABLE NMI
002099          STA      E.REG
002100          RTS
002101          SBTL     "KEYBOARD NMI HANDLER"
002102          REP      60
002103 *
002104 * BY DEFAULT, KEYBOARD NMI IS IGNORED.  THE USER MAY
002105 * PROCESS NMI BY CHANGING THE ADDRESS IN SYSTEM GLOBAL.
002106 *
002107          REP      60
002108 *
002109 NMIDBUG     EQU      *
002110          TSX      ;SAVE THE STACK POINTER
002111          STX      NMISPSV
002112          LDA      #$03          ;SELECT MONITOR'S ZERO PAGE
002113          STA      Z.REG
002114          LDA      E.REG
002115          ORA      #$03          ;SELECT MONITOR ROM
002116          STA      E.REG
002117          JSR      $F901          ;CALL THE MONITOR
002118 *
002119 NMICONT     EQU      *
002120          LDA      E.REG
002121          ORA      #BITON2          ;FORCE PRIMARY STACK
002122          STA      E.REG
002123          LDX      NMISPSV
002124          TXS      ;RESTORE STACK POINTER
002125          RTS
002126          SBTL     "EVENT QUEUE MANAGER"
002127          REP      60
002128 *
002129 * THE EVENT QUEUE IS USED TO HOLD THE PARAMETERS OF EVENTS
002130 * THAT HAVE BEEN DETECTED BUT NOT YET RECOGNIZED.  EVENT
002131 * QUEUE ENTRIES ARE ORGANIZED INTO TWO LINKED LISTS; A FREE
002132 * LIST AND AN ACTIVE LIST.  EACH ENTRY IS SIX BYTES LONG,
002133 * WITH THE FIRST BYTE (BYTE 0) USED AS A LINK.  THE LINK
002134 * BYTE CONTAINS THE TABLE INDEX OF THE NEXT ENTRY IN THE
002135 * LIST.  BECAUSE OF THE INDEXING METHOD, THE EVENT QUEUE

```

```

002136 * MUST NOT EXCEED 256 BYTES.
002137 *
002138 * ENTRY ZERO IS A SPECIAL ENTRY. BYTE 0 IS THE INDEX OF
002139 * THE FIRST ACTIVE ENTRY; BYTE 1 CONTAINS A ZERO, ALLOWING
002140 * ENTRY 0 TO BE USED AS THE ACTIVE EVENT LIST TERMINATOR;
002141 * BYTE 2 CONTAINS THE INDEX OF THE FIRST FREE ENTRY; AND
002142 * BYTES 4 THROUGH 6 ARE UNUSED.
002143 *
002144 * THE FREE LIST IS LINKED LIFO. THE ONLY VALID BYTE IN A
002145 * FREE ENTRY IS THE LINK BYTE; THE REMAINING BYTES ARE
002146 * UNDEFINED. THE FREE LIST IS TERMINATED BY A LINK BYTE
002147 * CONTAINING A ZERO.
002148 *
002149 * THE ACTIVE LIST IS LINKED IN DECREASING PRIORITY ORDER
002150 * WITH ENTRIES OF EQUAL PRIORITY LINKED FIFO. BYTES 1
002151 * THROUGH 5 CONTAIN THE EVENT PRIORITY, EVENT ID, LOW BYTE
002152 * OF THE EVENT ADDRESS, HIGH BYTE OF THE EVENT ADDRESS, AND
002153 * THE ADDRESS BANK. THE ACTIVE LIST IS TERMINATED BY AN
002154 * ENTRY WITH AN EVENT PRIORITY OF ZERO.
002155 *
002156 *           REP           60
002157 *           PAGE
002158 *           REP           60
002159 *
002160 * SUBROUTINE 'QUEEVENT' IS USED TO ENTER AN EVENT INTO THE
002161 * EVENT QUEUE. ACTIVE EVENTS ARE LINKED IN DECREASING
002162 * PRIORITY ORDER WITH EVENTS OF EQUAL PRIORITY LINKED FIFO.
002163 * EVENTS ARE REMOVED FROM THE QUEUE AS THEY ARE RECOGNIZED
002164 * BY THE DISPATCHER.
002165 *
002166 * PARAMETERS:
002167 *   X: EVENT PARAMETER ADDRESS (LOW BYTE)
002168 *   Y: EVENT PARAMETER ADDRESS (HIGH BYTE)
002169 *
002170 *   EVENT      0      1      2      3      4
002171 *   PARS:  +-----+-----+-----+-----+
002172 *           |  PRI  |  ID  |  ADR.L |  ADR.H |  ADR.B |
002173 *           +-----+-----+-----+-----+
002174 *           PRI:  EVENT PRIORITY
002175 *           ID:   EVENT ID BYTE
002176 *           ADR:  EVENT ADDRESS (LOW, HIGH, BANK)
002177 *
002178 * EXIT CONDITIONS:
002179 *   CARRY:  CLEAR
002180 *   A, X, Y:  UNDEFINED
002181 *
002182 *           REP           60
002183 *
002184 * QUEEVENT      EQU      *
002185 *               CLC

```

```

002186      PHP
002187      SEI
002188      LDA      E.REG
002189      STA      QEVTEMP
002190      ORA      #BITON2          ;FORCE PRIMARY STACK
002191      AND      #BITOFF3         ; AND WRITE ENABLE
002192      STA      E.REG
002193      LDA      QEVTEMP
002194      PHA
002195      LDA      Z.REG
002196      PHA
002197      LDA      #0
002198      STA      Z.REG          ;SET ZERO PAGE := 0
002199  *
002200      STX      QEVARGS
002201      STY      QEVARGS+1       ;SET ARGUMENT POINTER
002202      LDY      #0
002203      LDA      (QEVARGS),Y     ;GET PRIORITY
002204      BEQ      Q.EXIT         ; IGNORE IF ZERO
002205  *
002206      LDX      EVQ.FREE
002207      BEQ      Q.FULL
002208      STX      QEV.THIS        ;GET FIRST FREE ENTRY
002209      LDA      EVQ.LINK,X     ; AND DELINK IT
002210      STA      EVQ.FREE
002211  *
002212      LDY      #EVQ.SIZ-2
002213  QEV010  LDA      (QEVARGS),Y ;COPY ARGUMENTS
002214      STA      EVQ.BANK,X     ; INTO NEW ENTRY
002215      DEX
002216      DEY
002217      BPL      QEV010
002218  *
002219      LDX      QEV.THIS
002220      LDY      #0
002221  QEV020  STY      QEV.LAST
002222      LDA      EVQ.LINK,Y
002223      TAY
002224      LDA      EVQ.PRI,Y     ;SCAN EVENT QUEUE
002225      CMP      EVQ.PRI,X     ; FOR PROPER POSITION
002226      BCS      QEV020
002227  *
002228      TYA
002229      STA      EVQ.LINK,X     ;RELINK EVENT INTO QUEUE
002230      TXA
002231      LDY      QEV.LAST
002232      STA      EVQ.LINK,Y
002233  *
002234  Q.EXIT  PLA
002235      STA      Z.REG          ;RESTORE Z REGISTER

```

```
002236          PLA
002237          STA      E.REG          ;RESTORE E REGISTER
002238          PLP
002239          RTS
002240 *
002241 Q.FULL      LDA      #>EVQOVFL    ;EVENT QUEUE OVERFLOW
002242          JSR      SYSDEATH
002243          LST      ON
002244
002245 ZZEND      EQU      *
002246 ZZLEN      EQU      ZZEND-ZZORG
002247          IFNE    ZZLEN-LENIPL
002248          FAIL    2,"SOSORG        FILE IS INCORRECT FOR IPL"
002249          FIN
002250
002251 *****
002252 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: IPL.SRC2
002253 *****
002254
002255
002256
```

```

002257 =====
002258 DOCUMENT :SOS1.3.1of5.ONE:SOS.OPRMSG.SRC.TEXT
002259 =====
002260
002261 *****
002262 * APPLE /// SOS 1.3 SOURCE CODE FILE: OPRMSG.SRC
002263 *****
002264 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
002265
002266             SBTL           "SOS 1.1 OPERATOR MESSAGE/REPLY"
002267             REL
002268             INCLUDE       SOSORG,6,1,254
002269             ORG           ORGOMSG
002270 ZZORG        EQU         *
002271             MSB          OFF
002272             REP          60
002273 *
002274 *             COPYRIGHT (C) APPLE COMPUTER INC. 1981
002275 *             ALL RIGHTS RESERVED
002276 *
002277             REP          60
002278 *
002279 * THIS MODULE CONTAINS THE BLOCK FILE MANAGERS'S OPERATOR
002280 * INTERFACE. IT DISPLAYS A MESSAGE IN A FOUR LINE BY
002281 * FOURTY COLUMN WINDOW, THEN WAITS FOR THE USER TO TOGGLE
002282 * THE ALPHA-LOCK KEY BEFORE RETURNING.
002283 *
002284 * THE VERTICAL BLANKING FLAGS AND COMPOSITE BLANKING
002285 * TIMER ARE USED TO MAINTAIN THE DISPLAY. MEMORY PAGE
002286 * $02 IS USED FOR TEMPORARY STORAGE. ON EXIT, ALL
002287 * RESOURCES ARE RESTORED TO THEIR PREVIOUS STATES.
002288 *
002289 * ENTRY POINT: OPMSGRPLY
002290 *
002291 * PARAMETERS: X -- MESSAGE ADDRESS (LOW BYTE)
002292 *             Y -- MESSAGE ADDRESS (HIGH BYTE)
002293 *             (THE MESSAGE MUST RESIDE IN THE CURRENT BANK)
002294 *
002295 * RESULT:   A -- RESPONSE KEYSTROKE
002296 *           X, Y -- UNDEFINED
002297 *
002298             REP          60
002299 *
002300 *
002301             ENTRY        OPMSGRPLY
002302 *
002303             EXTRN        SCRNMODE
002304             PAGE
002305 *

```

```

002306 *   HARDWARE EQUATES
002307 *
002308 Z.REG      EQU      $FFD0
002309 E.REG      EQU      $FFDF
002310 *
002311 KBPORT     EQU      $C008
002312 *
002313 BELL        EQU      $C040
002314 *
002315 VM0         EQU      $C050
002316 VM1         EQU      $C052
002317 VM2         EQU      $C054
002318 VM3         EQU      $C056
002319 *
002320 E.T2         EQU      $FFE8
002321 E.ACR        EQU      $FFEB
002322 E.PCR        EQU      $FFEC
002323 E.IFR        EQU      $FFED
002324 E.IER        EQU      $FFEE
002325 *
002326 *   ZERO PAGE DECLARATIONS
002327 *
002328                DSECT
002329 ZPBASE       EQU      $200
002330                ORG      $0000                ;ZERO PAGE DECLARATIONS
002331 MSGPTR       DS      2                      ;MESSAGE POINTER
002332 MSGIDX       DS      1
002333 *
002334 SCRNIIDX     DS      1
002335 SCRNPTR      DS      2
002336 DATAPTR      DS      2
002337 DATABUF     DS      160
002338 *
002339 SV.ZREG       DS      1
002340 SV.EREG       DS      1
002341 SV.SMODE     DS      1
002342 SV.EACR       DS      1
002343 SV.EPCR       DS      1
002344 SV.EIER       DS      1
002345 *
002346 FLAG         DS      1
002347                DEND
002348                PAGE
002349 OPMSGRPLY     EQU      *
002350 *
002351 *
002352 *   SAVE CURRENT VALUES AND SET UP ZERO PAGE,
002353 *   ENVIRONMENT, SCREEN MODE, AND E.6522 REGISTERS.
002354 *
002355                PHP

```

```

002356      SEI
002357      LDA      Z.REG
002358      STA      ZPBASE+SV.ZREG      ;SAVE ZERO PAGE
002359      LDA      #<ZPBASE
002360      STA      Z.REG
002361      STX      MSGPTR      ;SAVE MESSAGE ADDRESS
002362      STY      MSGPTR+1
002363      LDA      E.REG
002364      STA      SV.EREG      ;SAVE ENVIRONMENT
002365      AND      #$5F
002366      ORA      #$40
002367      STA      E.REG      ;SCREEN OFF, I/O SPACE ON
002368      LDA      SCRNMODE
002369      STA      SV.SMODE      ;SAVE SCREEN MODE
002370      LDA      #$00
002371      STA      SCRNMODE
002372      BIT      VM0      ;SET 40 COLUMN
002373      BIT      VM1      ; BLACK & WHITE TEXT
002374      BIT      VM2
002375      BIT      VM3
002376      LDX      E.ACR
002377      TXA
002378      AND      #$20
002379      STA      SV.EACR      ;SAVE AUXILIARY CONTROL REG
002380      TXA
002381      ORA      #$20
002382      STA      E.ACR      ;SET UP BL TIMER
002383      LDX      E.PCR
002384      TXA
002385      AND      #$F0
002386      STA      SV.EPCR      ;SAVE PERIPHERAL CONTROL REG
002387      TXA
002388      AND      #$0F
002389      ORA      #$60
002390      STA      E.PCR      ;SET UP VBL FLAGS
002391      LDA      E.IER
002392      AND      #$38
002393      STA      E.IER      ;MASK VBL & BL INTERRUPTS
002394      STA      SV.EIER      ;SAVE INTERRUPT MASKS
002395      PLP
002396      *
002397      *
002398      *  SAVE SCREEN DATA AND CLEAR MESSAGE WINDOW
002399      *
002400      LDX      #3
002401  OPR010      JSR      SETPTRS
002402      LDY      #39
002403  OPR020      LDA      (SCRNPTR),Y      ;SAVE SCREEN DATA
002404      STA      (DATAPTR),Y
002405      LDA      #$A0

```

```

002406          STA      (SCRNPTR),Y      ;BLANK SCREEN
002407          DEY
002408          BPL      OPR020
002409          DEX
002410          BPL      OPR010
002411 *
002412 *
002413 *  MOVE MESSAGE TO WINDOW
002414 *
002415          BIT      BELL
002416          LDX      #$00
002417          STX      MSGIDX
002418 OPR100     JSR      SETPTRS
002419          LDY      #$00
002420          STY      SCRNIIDX
002421 OPR110     LDY      MSGIDX
002422          INC      MSGIDX
002423          LDA      (MSGPTR),Y      ;SET UP MESSAGE
002424          BEQ      OPR110
002425          BMI      OPR200
002426          CMP      #$0D
002427          BEQ      OPR120
002428          LDY      SCRNIIDX
002429          INC      SCRNIIDX
002430          ORA      #$80
002431          STA      (SCRNPTR),Y
002432          CPY      #39
002433          BCC      OPR110
002434 OPR120     INX
002435          CPX      #4
002436          BCC      OPR100
002437 *
002438 *
002439 *  DISPLAY MESSAGE UNTIL ALPHA-LOCK KEY TOGGLES
002440 *
002441 OPR200     LDY      #2
002442          LDA      KBPORT
002443          AND      #$08
002444          STA      FLAG
002445 OPR210     JSR      VIDEO
002446          LDA      KBPORT
002447          AND      #$08
002448          CMP      FLAG
002449          BEQ      OPR210
002450          STA      FLAG
002451          DEY
002452          BNE      OPR210
002453 *
002454 *
002455 *  RESTORE PREVIOUS CONTENTS OF WINDOW

```



```

002456 *
002457          LDX          #3
002458 OPR400      JSR          SETPTRS
002459          LDY          #39
002460 OPR410      LDA          (DATAPTR),Y
002461          STA          (SCRNPTR),Y
002462          DEY
002463          BPL          OPR410
002464          DEX
002465          BPL          OPR400
002466 *
002467 *
002468 * RESTORE E.6522, SCREEN MODE, ENVIRONMENT, & ZERO PAGE
002469 * THEN RETURN TO CALLER
002470 *
002471          PHP
002472          SEI
002473          LDA          E.ACR
002474          AND          #$DF
002475          ORA          SV.EACR          ;RESTORE AUXILIARY CONTROL REG
002476          STA          E.ACR
002477          LDA          E.PCR
002478          AND          #$0F
002479          ORA          SV.EPCR          ;RESTORE PERIPHERAL CONTROL REG
002480          STA          E.PCR
002481          LDA          SV.EIER          ;RESTORE INTERRUPT ENABLE REG
002482          ORA          #$80
002483          STA          E.IER
002484          LDA          SV.SMODE          ;RESTORE SCREEN MODE
002485          STA          SCRNMODE
002486          LSR          A
002487          BCC          OPR500
002488          BIT          VM0+1          ;RESTORE VIDEO MODE
002489 OPR500      LSR          A
002490          BCC          OPR510
002491          BIT          VM1+1
002492 OPR510      LSR          A
002493          BCC          OPR520
002494          BIT          VM2+1
002495 OPR520      BIT          SCRNMODE
002496          BVC          OPR530
002497          BIT          VM3+1
002498 OPR530      LDA          SV.EREG          ;RESTORE ENVIRONMENT
002499          STA          E.REG
002500          LDA          SV.ZREG          ;RESTORE ZERO PAGE
002501          STA          Z.REG
002502          PLP
002503          RTS
002504          PAGE
002505          REP          60

```

```

002506 *
002507 * SUBROUTINE VIDEO
002508 *
002509 * THIS SUBROUTINE POLLS THE VERTICAL-BLANKING AND
002510 * COMPOSITE-BLANKING-TIMER FLAGS AND TURNS THE SCREEN
002511 * OFF AND ON SO THAT ONLY THE MESSAGE WINDOW WILL BE
002512 * DISPLAYED.
002513 *
002514 * THE E.6522 MUST BE INITIALIZED SO THAT E.CB2 FLAGS THE
002515 * POSITIVE EDGE OF VBL AND E.T2 COUNTS BL PULSES. THE
002516 * INTERRUPTS MUST BE MASKED AND THE PROPER COUNT MUST
002517 * ALREADY BE STORED IN THE LOW ORDER BYTE OF E.T2.
002518 *
002519 * ENTRY: VIDEO
002520 *
002521 * PARAMETERS: INTERRUPT SYSTEM DISABLED
002522 *
002523 * EXIT: A -- UNDEFINED
002524 *        X, Y -- PRESERVED
002525 *
002526         REP        60
002527 *
002528 VIDEO        EQU        *
002529         LDA        E.IFR
002530         AND        #$28                ;GET VBL & BL FLAGS
002531         BEQ        VID030
002532         STA        E.IFR                ;CLEAR FLAGS
002533         AND        #$20                ;WHICH FLAG?
002534         BNE        VID010                ; BL
002535 *
002536         LDA        #$1F
002537         STA        E.T2                ;SET UP BL TIMER
002538         LDA        #$00
002539         STA        E.T2+1
002540         LDA        E.REG
002541         ORA        #$20                ;SET UP FOR SCREEN ON
002542         SEC
002543         BCS        VID020
002544 *
002545 VID010        LDA        E.REG
002546         AND        #$DF                ;SET UP FOR SCREEN OFF
002547         CLC
002548 *
002549 VID020        STA        E.REG
002550         LDA        #$00
002551         ROR        A
002552         STA        SCRNMODE
002553 VID030        RTS
002554         PAGE
002555         REP        60

```

```

002556 *
002557 * SUBROUTINE SETPTRS
002558 *
002559 * THIS SUBROUTINE SETS UP THE POINTERS TO THE MESSAGE
002560 * WINDOW AND DATA SAVE AREA.
002561 *
002562 * ENTRY:  SETPTRS
002563 *
002564 * PARAMETERS:  X -- LINE NUMBER [0..3]
002565 *
002566 * EXIT:  A -- UNDEFINED
002567 *        X, Y -- PRESERVED
002568 *
002569 *                REP        60
002570 *
002571 SETPTRS        EQU        *
002572                TXA
002573                LSR        A
002574                ORA        #$04
002575                STA        SCRNPTR+1
002576                LDA        #$00
002577                ROR        A
002578                STA        SCRNPTR
002579                LDA        #<DATABUF
002580                STA        DATAPTR+1
002581                LDA        DBUFADR,X
002582                STA        DATAPTR
002583                RTS
002584 *
002585 DBUFADR        EQU        *
002586                DFB        >0*40+DATABUF
002587                DFB        >1*40+DATABUF
002588                DFB        >2*40+DATABUF
002589                DFB        >3*40+DATABUF
002590                LST        ON
002591
002592 ZZEND          EQU        *
002593 ZZLEN          EQU        ZZEND-ZZORG
002594                IFNE        ZZLEN-LENOMSG
002595                FAIL        2,"SOSORG          FILE IS INCORRECT FOR OPRMSG"
002596                FIN
002597
002598 *****
002599 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: OPRMSG.SRC
002600 *****
002601
002602
002603

```

```

002604 =====
002605 DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.A.SRC.TEXT
002606 =====
002607
002608 *****
002609 * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.A.SRC
002610 *****
002611 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
002612
002613             PAGE
002614             REP      110
002615 *
002616 * $1E00 +-----+
002617 * !      SOSLDR      !<-ENTRY      SOS MEMORY MAP
002618 * $1FFF +-----+ ----- (128K APPLE ///)
002619 *
002620 *             BANK 0             BANK 1             BANK 2
002621 * $2000 +-----+ +-----+ +-----+
002622 * !             !             !             !             !
002623 * !             !             !             !             !
002624 * !             !             !             !             !
002625 * !             !             !             !             !
002626 * !             !             !             !             !
002627 * !             !             !             !             !
002628 * !             !             !             !             !
002629 * !             !             !             !             !
002630 * !             !             !             !             !
002631 * !             !             !             !             !
002632 * !             !             !             !             !
002633 * !             !             !             !             !
002634 * !             !             !             !             !
002635 * !             !             !             !             !
002636 * !             !             !             !             !
002637 * !             !             !             !             !
002638 * !             !             !             !             !
002639 * !             !             !             !             !
002640 * !             !             !             !             !
002641 * !             !             !             !             !
002642 * !             !             !             !             !
002643 * !             !             !             !             !
002644 * !             !             !             !             !
002645 * !             !             !             !             !
002646 * !             !             !             !             !
002647 * !             !             !             !             !
002648 * !             !             !             !             !
002649 * !             !             !             !             !
002650 * !             !             !             !             !
002651 * !             !             !             !             !
002652 * !             !             !             !             !

```

```

002653 *      !           !           !           !           !           !
002654 *  $9FFF +-----+ +-----+ +-----+
002655 *
002656 *
002657 *  $A000 +-----+
002658 *      !   SOSBOOT   !
002659 *      +-----+
002660 *
002661 *
002662 *  FIGURE 1.  SOS KERNEL FILE READ INTO $2:1E00..9FFF BY SOS BOOT IN BLOCKS 0,1.
002663 *             SOS LOADER BEGINS EXECUTION AT THIS POINT.
002664 *
002665 *
002666 *
002667 *
002668 *             REP      110
002669 *             PAGE
002670 *             REP      110
002671 *
002672 *  $1E00 +-----+
002673 *      !   SOSLDR   !           SOS MEMORY MAP
002674 *  $1FFF +-----+           (128K APPLE ///)
002675 *
002676 *             BANK 0           BANK 1           BANK 2
002677 *  $2000 +-----+ +-----+ +-----+
002678 *      !           !           !           !           !
002679 *      !   SOSLDR   !           !           !           !
002680 *      !     &     !           !           !           !
002681 *      !  INIT MODULE !           !           !           !
002682 *      !           !           !           !           !
002683 *  LDREND ! - - - - - !           !           !           !
002684 *      !  FILE BUFFER !           !           !           !
002685 *      ! - - - - - !           !           !           !
002686 *      !           !           !           !           !
002687 *      !           !           !           !           !
002688 *      !           !           !           !           !
002689 *      !           !           !           !           !
002690 *      !           !           !           !           !
002691 *      !           !           !           !           !
002692 *      !  INTERPRETER !           !  INTERPRETER !           !
002693 *      !     FILE     !           !     FILE     !           !
002694 *      !           !           !           !           !
002695 *      !           !           !           !           !
002696 *      !           !           !           !           !
002697 *      !           !           !           !           !
002698 *      !           !           !           !           !
002699 *      !           !           !           !           !
002700 *      !           !           !           !           !
002701 *      !           !           !           !           !
002702 *      !           !           !           !           !

```

```

002703 *      !      !      !      !      !      !      !
002704 *      !      !      !      !      !      !      !
002705 *      !      !      !      !      !      !      !
002706 *      !      !      !      !      !      !      !
002707 *      !      !      !      !      !      !      !
002708 *      !      !      !      !      !      !      !
002709 *      !      !      !- - - EOF - - -!      !      !
002710 *      $9FFF +-----+      +-----+      +-----+
002711 *
002712 *
002713 *
002714 *
002715 *      FIGURE 2.  SOS INTERPRETER FILE READ INTO BANKS 0 AND 1
002716 *                  USING EXTENDED ADDRESSING (X=$80).
002717 *
002718 *
002719 *
002720 *
002721 *
002722 *                  REP      110
002723 *
002724 *                  CHN      SOSLDR.B.SRC
002725 *
002726 * *****
002727 *      * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.A.SRC
002728 * *****
002729 *
002730 *
002731 *

```

```

002732 =====
002733 DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.C.SRC.TEXT
002734 =====
002735
002736 *****
002737 * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.C.SRC
002738 *****
002739 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
002740
002741             PAGE
002742             REP          100
002743 *
002744 * SUBROUTINES:
002745 *
002746 * SOSLDR             "MAIN PROGRAM"
002747 *
002748 *   SOSLDR1         "PROCESSES KERNEL/INTERPRETER/DRIVER FILES"
002749 *
002750 * (1)  MOVE         "MOVES SRC.P..SRC.P+CNT-1 TO DST.P..DST.P+CNT-1"
002751 *
002752 *   INIT.KRNL      "CALLS KERNEL INITIALIZATION MODULES"
002753 *
002754 *   WELCOME        "PRINTS WELCOME MESSAGE ("APPLE ///", VERSION, DATE/TIME, COPYRIGHT)"
002755 *
002756 *   ADVANCE        "ADVANCES WRK.PTR TO NEXT INTERP/KERNEL MODULE.  INITS SRC.P, DST.P, CNT FOR MOVE"
002757 *
002758 *   REVERSE        "REVERSES TITLE/CODE/RELOC COUNTS TO ALLOW DRIVER FILE TO BE PROCESSED FM BACK TO FRONT"
002759 *
002760 *   DADVANCE       "ADVANCES WORK.P TO NEXT DRIVER MODULE.  INITS SRC.P, CNT, REL.P FOR MOVE"
002761 *
002762 *   DADD           "ADVANCES WORK.P TO NEXT DRIVER FIELD"
002763 *
002764 *   FLAGS          "PROCESSES "INACTIVE" & "PAGE ALIGN" FLAGS IN DRIVER MODULE'S DIBS"
002765 *
002766 *   NEXT.DIB       "ADVANCES TO NEXT DIB IN DRIVER MODULE"
002767 *
002768 *   GETMEM         "COMPUTES DESTINATION BASE ADDRESS FOR NEXT DRIVER MODULE"
002769 *
002770 *   NEWDST        "COMPUTES DESTINATION BASE ADDRESS, ALIGNING ON PAGE BOUNDARY IF REQUESTED"
002771 *
002772 *   BUILD.DSEG     "COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT"
002773 *
002774 *   RELOC          "RELOCATES DRIVER MODULE'S CODE FIELD USING RELOCATION FIELD"
002775 *
002776 * (1)  LINK         "LINKS FIRST DIB TO PREVIOUS DRIVER'S LAST "ACTIVE" DIB, AND ADDS SDT ENTRY"
002777 *
002778 *   SET.DRIVES     "INITIALIZES DIB LINKS IN KERNEL'S FLOPPY DRIVER"
002779 *
002780 * (1)  ALLOC.DEV   "ADDS A NEW ENTRY TO THE DEVICE MANAGER'S SYSTEM DEVICE TABLE (SDT)"

```

```

002781 *
002782 *      ALLOC.SEG      "ALLOCATES SEGMENTS FOR KERNEL, INTERPRETER AND SYSTEM WORK AREA"
002783 *
002784 *      RSEG          "CALLS MEMORY MANAGER TO ALLOCATE SEGMENTS FOR THE KERNEL AND INTERPRTER"
002785 *
002786 *      ALLOC.DSEG    "ALLOCATES SEGMENTS FOR DRIVER MODULES"
002787 *
002788 *      ERROR          "DISPLAYS ERROR MESSAGE, SOUNDS BELL AND LOOPS UNTIL CONTROL/RESET PRESSED"
002789 *
002790 * (1) - INDICATES THAT THE ROUTINE PERFORMS BANK SWITCHING AND MUST(!) BE OUTSIDE THE 32K RAM BANKS.
002791 *           REP      100
002792 *           PAGE
002793 *           REP      100
002794 *
002795 * SOS.KERNEL FILE FORMAT
002796 *
002797 * (8)  LABEL          <----+
002798 *      = "SOS KRNL"   !
002799 *                      !
002800 * (2)  HEADER COUNT   !
002801 *      HEADER        !
002802 *      = # OF FLOPPY DRIVES !   CONTAINED IN THIS LISTING
002803 *      = INTERPRETER PATHNAME !
002804 *      = DRIVER PATHNAME  !
002805 *                      !
002806 * (4)  ADR & COUNT    !
002807 *      SOSLDR CODE     <----+
002808 *
002809 * (4)  ADR & COUNT
002810 *      GLOBALS
002811 *
002812 * (4)  ADR & COUNT
002813 *      KERNEL CODE
002814 *
002815 *           REP      100
002816 *
002817 * SOS.INTERP FILE FORMAT
002818 *
002819 * (8)  LABEL
002820 *      = "SOS NTRP"
002821 *
002822 * (2)  HEADER COUNT
002823 *
002824 * (4)  ADR & COUNT
002825 *      INTERPRETER CODE
002826 *
002827 *           REP      100
002828 *
002829 * SOS.DRIVER FILE FORMAT
002830 *

```



```

002831 * (8) LABEL
002832 *      = "SOS DRVR"
002833 *
002834 * (2)  HEADER COUNT
002835 *      = # OF FLOPPY DRIVES
002836 *      = CHARACTER SET TABLE
002837 *      = KEYBOARD TABLE
002838 *      ...
002839 *
002840 * (2)  DM #N TITLE COUNT      <----+      +-----+
002841 *      TITLE FIELD           !              RELOCATION FIELD FORMAT      !
002842 * (2)  DM #N CODE COUNT      !              +-----+              !
002843 *      CODE FIELD            ! DRIVER MODULE #N      ! CONSISTS OF A LIST OF 2 BYTE POINTERS !
002844 * (2)  DM #N RELOC COUNT      !              ! WHICH POINT TO THE LOW BYTE OF A TWO !
002845 *      RELOC FIELD           <----+      +-----+              !
002846 *      ...
002847 *
002848 *      $FFFF = THE END
002849 *
002850 *      REP      100
002851 *      PAGE
002852 *      REP      100
002853 *
002854 * SOSLDR - EXTERNAL DECLARATIONS
002855 *
002856 *      REP      100
002857 *      EXTRN    SYSBANK
002858 *      EXTRN    MEMSIZE
002859 *      EXTRN    SCRNMODE
002860 *      EXTRN    SOSVER
002861 *      EXTRN    SOSVERL
002862 *
002863 *      EXTRN    INT.INIT          ; (IPL) INTERRUPT INIT
002864 *      EXTRN    EVQ.INIT         ; (IPL) EVENT QUEUE INIT
002865 *      EXTRN    DMGR.INIT        ; DEVICE MANAGER INIT
002866 *      EXTRN    MAX.DNUM          ;
002867 *      EXTRN    SDT.SIZE
002868 *      EXTRN    SDT.DIBL
002869 *      EXTRN    SDT.DIBH
002870 *      EXTRN    SDT.ADRL
002871 *      EXTRN    SDT.ADRH
002872 *      EXTRN    SDT.BANK
002873 *      EXTRN    SDT.UNIT
002874 *      EXTRN    BLKD.SIZE
002875 *      EXTRN    BLKDLST
002876 *      EXTRN    CFMGR.INIT        ; CHAR FILE MANAGER INIT
002877 *      EXTRN    MMGR.INIT         ; MEMORY MANAGER INIT
002878 *      EXTRN    BMGR.INIT        ; BUFFER FILE MANAGER INIT
002879 *      EXTRN    BFM.INIT         ; BLOCK FILE MANAGER INIT
002880 *      EXTRN    BFM.INIT2        ; BLOCK FILE MANAGER INIT2

```

```

002881          EXTRN      CLK.INIT          ; CLOCK SYSTEM CALL INIT
002882 *
002883          EXTRN      DIB1                ; ON BOARD DISK DRIVER'S DIBS (1-4)
002884          EXTRN      DIB2
002885          EXTRN      DIB3
002886          EXTRN      DIB4
002887 *
002888 *ENTRY I.BASE.P ; USED BY BFM.INIT2 (HARDWIRED!)
002889          PAGE
002890          REP          100
002891 *
002892 * FILE DATA DECLARATIONS
002893 *
002894          REP          100
002895 * KERNEL FILE
002896          REP          100
002897 K.FILE          ASC          "SOS KRNL"
002898 K.HDR.CNT       DW           LDR.ADR-K.DRIVES
002899 K.DRIVES       DFB          $1
002900 K.FLAGS        DFB          $0          ; RESERVED FOR FUTURE USE
002901 I.PATH        DFB          $E
002902          ASC          ".D1/SOS.INTERP"
002903          DS           $30-$F
002904 D.PATH         DFB          $E
002905          ASC          ".D1/SOS.DRIVER"
002906          DS           $30-$F
002907 LDR.ADR        DW           $0
002908 LDR.CNT        DW           ZZEND-SOSLDR
002909          REP          100
002910 * INTERPRETER/DRIVER FILES      <--+
002911 * ERROR MESSAGES                !      DEFINED IN BACK OF THIS LISTING
002912 * WELCOME MESSAGES              <--+
002913          REP          100
002914          PAGE
002915          REP          100
002916 *
002917 * SOSLDR - DATA DECLARATIONS (1)
002918 *
002919          REP          100
002920 TRUE          EQU          $80
002921 FALSE        EQU          $0
002922 *
002923 Z.REG         EQU          $FFD0
002924 E.REG         EQU          $FFDF
002925 B.REG         EQU          $FFEF
002926 *
002927 CZPAGE       EQU          $1A00
002928 CSPAGE       EQU          $1B00
002929 CXPAGE       EQU          $1600
002930 SZPAGE       EQU          $1800

```

```

002931 SXPAGE      EQU      $1400
002932 SSPAGE      EQU      $0100
002933 *
002934 ROM.ADR      EQU      $F1B9
002935 ROM.ID        EQU      $A0
002936              PAGE
002937              REP      100
002938 *
002939 * SOSLDR - DATA DECLARATIONS (2)
002940 *
002941              REP      100
002942 ZPAGE          EQU      $00
002943 *
002944 K.BASE          EQU      ZPAGE+$0      ; SOSLDR1 SUBROUTINE      +-----+
002945 I.BASE.P        EQU      ZPAGE+$2      ;                          ! <VARNAME>.P ::= 3 BYTE ZPAGE POINTER !
002946 RDBUF.P        EQU      ZPAGE+$4      ;                          +-----+
002947 SYSBUF.P       EQU      ZPAGE+$6
002948 TEMP.BANK       EQU      ZPAGE+$8
002949 TEMP.ADRH       EQU      ZPAGE+$9
002950 WORK.P         EQU      ZPAGE+$A
002951 *
002952 REV.SAVE        EQU      ZPAGE+$C      ; REVERSE SUBROUTINE
002953 *
002954 FIRST.ADIB      EQU      ZPAGE+$10     ; FLAGS SUBROUTINE
002955 PREV.ADIB.P     EQU      ZPAGE+$12
002956 DIB.P           EQU      ZPAGE+$14
002957 PG.ALIGN        EQU      ZPAGE+$16
002958 DIB.FLAGS       EQU      $14
002959 DIB.DCB         EQU      $20
002960 *
002961 PREVBANK        EQU      ZPAGE+$18     ; GETMEM SUBROUTINE
002962 PREVDST         EQU      ZPAGE+$19
002963 *
002964 CODE.P          EQU      ZPAGE+$1C     ; RELOCATION SUBROUTINE
002965 REL.P           EQU      ZPAGE+$1E
002966 REL.END         EQU      ZPAGE+$20
002967 *
002968 SRC.P           EQU      ZPAGE+$22     ; MOVE SUBROUTINE
002969 DST.P           EQU      ZPAGE+$24
002970 CNT             EQU      ZPAGE+$26
002971 *
002972 DSTBANK         EQU      ZPAGE+$2A     ; LINK SUBROUTINE
002973 LINK.P          EQU      ZPAGE+$2C
002974 *
002975 DIB.ENTRY       EQU      2            ; ALLOC.DEV SUBROUTINE
002976 DIB.UNIT        EQU      4+16+2
002977 DIB.DTYPE       EQU      4+16+3
002978 *
002979 ETEMP           EQU      ZPAGE+$2E     ; ERROR SUBROUTINE
002980 *

```

```
002981 WTEMP          EQU          ZPAGE+$2F          ; WELCOME SUBROUTINE
002982
002983 *****
002984 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.C.SRC
002985 *****
002986
002987
002988
```

```

002989 =====
002990 DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.D.SRC.TEXT
002991 =====
002992
002993 *****
002994 * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.D.SRC
002995 *****
002996 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
002997
002998             PAGE
002999             REP          100
003000 *
003001 * SOS LOADER -
003002 *
003003 * (MAIN PROGRAM)
003004             REP          100
003005 SOSLDR             EQU          *                ;
003006             LDA          #0                ; ZERO SOS/USER X, Z AND STACK PAGES      ! SEE FIGURE 1. !
003007             TAX                ;
003008 SLDR010           STA          CZPAGE,X
003009             STA          CXPAGE,X
003010             STA          CSPAGE,X
003011             STA          SZPAGE,X
003012             STA          SXPAGE,X
003013             STA          SSPAGE,X
003014             DEX
003015             BNE          SLDR010
003016 *
003017             LDA          #$30                ; SETUP SOS CALL ENVIRONMENT (WRITE PROTECT=OFF)
003018             STA          E.REG                ; E:=( 0.0.1.1:0.0.0.0 )
003019 *
003020             LDX          #$FB                ; CONSOLE 1.0 MODIFIES STACK DURING D.INIT CALL
003021             TXS
003022             LDA          #<CZPAGE            ; ZREG:=CALLER'S Z PAGE
003023             STA          Z.REG
003024 *
003025             JSR          SOSLDR1            ; +-----+
003026 *
003027             LDA          E.REG
003028             AND          #$10                ; SETUP SOS CALL ENVIRONMENT (WRITE PROTECT=ON)
003029             ORA          #$28                ; E:=( 0.0.1.X:1.0.0.0 )
003030             STA          E.REG                ; ( 1.I.S.R:W.P.R.R )
003031 *
003032             LDX          #$FF                ; STACK.REG:=$FF
003033             TXS
003034             LDA          #<CZPAGE            ; ZREG:=CALLER'S Z PAGE
003035             STA          Z.REG
003036 *
003037             LDA          SYSBANK            ; BREG:=SYSBANK
003037

```

```

003038          STA      B.REG          ;
003039          JMP      (I.BASE.P)      ; SOS LOAD COMPLETE - JMP TO INTERPRETER
003040          *
003041          *THE END.
003042          REP      100
003043          PAGE
003044          REP      100
003045          *
003046          * MOVE ( IN:  SRC.P
003047          *           IN:  DST.P
003048          *           IN:  A="BANK"
003049          *           IN:  CNT      )
003050          *
003051          *           LOCAL:  END
003052          * (MOVES SRC.P..SRC.P+CNT-1 TO DST.P..DST.P+CNT-1)          "CNT PARM IS DESTROYED"
003053          REP      100
003054          MOVE     EQU      *
003055          TAX
003056          LDA      B.REG          ; SAVE BANK REGISTER
003057          PHA
003058          STX      B.REG          ; BREG:=A
003059          LDA      CNT+1          ; IF CNT <> 0
003060          ORA      CNT            ; THEN
003061          BEQ      MOVE.EXIT
003062          LDA      CNT            ; CNT:=CNT-1
003063          BNE      MOVE010
003064          DEC      CNT+1
003065          MOVE010  DEC      CNT
003066          CLC          ; SRC.P:=SRC.P+PAGE.CNT
003067          LDA      SRC.P+1
003068          ADC      CNT+1
003069          STA      SRC.P+1
003070          LDA      DST.P+1          ; DST.P:=DST.P+PAGE.CNT
003071          ADC      CNT+1
003072          STA      DST.P+1
003073          INC      CNT+1          ; PAGE.CNT:=PAGE.CNT+1
003074          LDY      CNT            ; Y:=BYTE.CNT
003075          BEQ      MOVE020          ; IF Y=0 THEN M2
003076          *
003077          MOVE.PAGE  LDA      (SRC.P),Y          ;M1:  DO
003078          STA      (DST.P),Y          ;           (DST.P),Y:=(SRC.P),Y
003079          DEY          ;           Y:=Y-1
003080          BNE      MOVE.PAGE          ; UNTIL Y=0
003081          MOVE020  LDA      (SRC.P),Y          ;M2:  (DST.P),Y:=(SRC.P),Y
003082          STA      (DST.P),Y
003083          DEY          ; Y:=Y-1
003084          DEC      SRC.P+1          ; SRC.P:=SRC.P-256
003085          DEC      DST.P+1          ; DST.P:=DST.P-256
003086          DEC      CNT+1          ; PAGE.CNT:=PAGE.CNT-1
003087          BNE      MOVE.PAGE          ; IF PAGE.CNT <> 0 THEN M1

```

```

003088 *
003089         INC      SRC.P+1          ; RESTORE SRC.P
003090         INC      DST.P+1          ;      "      DST.P
003091 *
003092 MOVE.EXIT  PLA                      ; RESTORE BANK REGISTER
003093         STA      B.REG
003094         RTS
003095         PAGE
003096         REP      100
003097 *
003098 * LINK ( IN:  DST.P
003099 *         IN:  DSTBANK
003100 *         IN:  PREVBANK
003101 *         IN:  FIRST.ADIB
003102 *         I/O: SDT.TBL
003103 *         I/O: BLKDLST
003104 *         OUT: LINKED DRIVER MODULE )
003105 *
003106 *         OWN:  LINK.P
003107 * (LINKS FIRST DIB TO PREVIOUS DRIVER'S LAST "ACTIVE" DIB, AND ADDS SDT ENTRY)
003108         REP      100
003109 LINK      EQU      *
003110         CLC                      ; FIRST.ADIB:=0:DST.P+FIRST.ADIB
003111         LDA      DST.P
003112         ADC      FIRST.ADIB
003113         STA      FIRST.ADIB
003114         LDA      DST.P+1
003115         ADC      FIRST.ADIB+1
003116         STA      FIRST.ADIB+1
003117         LDA      #0
003118         STA      CXPAGE+FIRST.ADIB+1
003119         LDA      PREVBANK          ; BREG:=PREVBANK
003120         STA      B.REG
003121         LDY      #0                ; (LINK.P):=FIRST.ADIB
003122         LDA      FIRST.ADIB
003123         STA      (LINK.P),Y
003124         INY
003125         LDA      FIRST.ADIB+1
003126         STA      (LINK.P),Y
003127         LDA      DSTBANK          ; BREG:=DSTBANK
003128         STA      B.REG
003129         LDA      FIRST.ADIB        ; LINK.P:=FIRST.ADIB
003130         STA      LINK.P
003131         LDA      FIRST.ADIB+1
003132         STA      LINK.P+1
003133 WALKLINKS JSR      ALLOC.DEV        ; ALLOC.DEV(LINK.P BREG.IN, SDT.TBL BLKDLST.IO)
003134 LINK010  LDY      #0                ; WHILE (LINK.P) <> 0 AND (LINK.P) <> LINK.P
003135         LDA      (LINK.P),Y
003136         INY
003137         ORA      (LINK.P),Y

```

```

003138      BEQ      LINK100
003139      LDA      (LINK.P),Y
003140      CMP      LINK.P+1
003141      BNE      LINK030
003142      DEY
003143      LDA      (LINK.P),Y
003144      CMP      LINK.P
003145      BEQ      LINK100
003146 LINK030  LDY      #0          ; DO LINK.P:=(LINK.P)
003147      LDA      (LINK.P),Y
003148      TAX
003149      INY
003150      LDA      (LINK.P),Y
003151      STX      LINK.P
003152      STA      LINK.P+1
003153      JSR      ALLOC.DEV      ; " ALLOC.DEV(LINK.P BREG.IN, SDT.TBL BLKDLST.IO)
003154      JMP      LINK010
003155      *
003156 LINK100  LDY      #0          ; (LINK.P):=0
003157      TYA
003158      STA      (LINK.P),Y
003159      INY
003160      STA      (LINK.P),Y
003161      DEY          ; BREG:=0
003162      STY      B.REG
003163      RTS
003164      *
003165      *
003166      *
003167      *
003168      * LINK.INIT ( IN:  A=# DRIVES
003169      *                IN:  DIB1..4
003170      *                I/O:  SDT.TBL
003171      *                I/O:  BLKDLST  )
003172      *
003173 LINK.INIT  EQU      *
003174      JSR      SET.DRIVES      ; SET.DRIVES(A=#DRIVES.IN, DIB1..4.IN)
003175      LDA      #0
003176      STA      MAX.DNUM      ; MAXDNUM:=0
003177      STA      BLKDLST      ; BLKDLST:=0
003178      STA      CXPAGE+LINK.P+1 ; LINK.P:=0:DIB1
003179      LDA      #>DIB1
003180      STA      LINK.P
003181      LDA      #<DIB1
003182      STA      LINK.P+1
003183      JMP      WALKLINKS
003184      PAGE
003185      REP      100
003186      *
003187      * ALLOC.DEV ( IN:  LINK.P

```



```

003188 *           IN:  B.REG
003189 *           I/O:  SDT.TBL                               (SYSTEM DEVICE TABLE)
003190 *                   IN:  SDT.SIZE = CONSTANT
003191 *                   IN:  DIB.ENTRY = CONSTANT           DEV  DIB  ADR  BANK  UNIT
003192 *                   IN:  DIB.UNIT = CONSTANT           !-----!-----!-----!-----!
003193 *                   IN:  DIB.DTYPE = CONSTANT           1  !      !      !      !      !
003194 *                   I/O:  MAX.DNUM                       2  !      !      !      !      !
003195 *                   OUT:  SDT.BANK                       .  !      !      !      !      !
003196 *                   OUT:  SDT.DIB                        .  !      !      !      !      !
003197 *                   OUT:  SDT.ADR                       .  !-----!-----!-----!-----!
003198 *                   OUT:  SDT.UNIT                       MAX.DNUM
003199 *           I/O:  BLKDLST
003200 *                   IN:  BLKD.SIZE = CONSTANT
003201 * (ADDS A NEW ENTRY TO THE DEVICE MANAGER'S SYSTEM DEVICE TABLE (SDT))
003202 *                   REP      100
003203 ALLOC.DEV      EQU      *
003204 *                   INC      MAX.DNUM           ; MAX.DNUM:=MAX.DNUM+1
003205 *                   LDX      MAX.DNUM           ; IF MAX.DNUM >= SDT.SIZE
003206 *                   CPX      #>SDT.SIZE         ; THEN
003207 *                   BCC      ADEV010
003208 *                   LDX      #ERR8X             ; ERROR("TOO MANY DEVICES")
003209 *                   LDY      #ERR8L
003210 *                   JSR      ERROR
003211 ADEV010        LDA      B.REG                 ; SDT.BANK,X:=BREG
003212 *                   STA      SDT.BANK,X
003213 *                   CLC                               ; SDT.DIB,X:=LINK.P+4
003214 *                   LDA      LINK.P
003215 *                   ADC      #4
003216 *                   STA      SDT.DIBL,X
003217 *                   LDA      LINK.P+1
003218 *                   ADC      #0
003219 *                   STA      SDT.DIBH,X
003220 *                   SEC                               ; SDT.ADR,X:=(LINK.P),DIB.ENTRY-1
003221 *                   LDY      #DIB.ENTRY
003222 *                   LDA      (LINK.P),Y
003223 *                   SBC      #1
003224 *                   STA      SDT.ADRL,X
003225 *                   INY
003226 *                   LDA      (LINK.P),Y
003227 *                   SBC      #0
003228 *                   STA      SDT.ADRH,X
003229 *                   LDY      #DIB.UNIT           ; SDT.UNIT,X:=(LINK.P),DIB.UNIT
003230 *                   LDA      (LINK.P),Y
003231 *                   STA      SDT.UNIT,X
003232 *                   LDY      #DIB.DTYPE         ; IF (LINK.P),DIB.DTYPE = "BLOCK DEVICE"
003233 *                   LDA      (LINK.P),Y
003234 *                   BPL      ADEV.EXIT
003235 *                   TXA                               ; THEN
003236 *                   INC      BLKDLST             ; BLKDLST:=BLKDLST+1
003237 *                   LDX      BLKDLST           ; IF BLKDLST >= BLKD.SIZE

```

```

003238          CPX      #>BLKD.SIZE      ;          THEN
003239          BCC      ADEV020
003240          LDX      #ERR9X              ;          ERROR("TOO MANY BLOCK DEVICES")
003241          LDY      #ERR9L
003242          JSR      ERROR
003243 ADEV020     STA      BLKDLST,X        ;          BLKDLST,X:=MAX.DNUM
003244 ADEV.EXIT   RTS                      ; RETURN
003245          PAGE
003246          REP      100
003247 *
003248 * SOSLDR1 ( )
003249 *
003250 * (PROCESSES KERNEL/INTERPRETER/DRIVER FILES)
003251          REP      100
003252 SOSLDR1     EQU      *
003253          LDX      #$1F              ; COPY ROM'S DISK CORE ROUTINE ZPAGE VARS TO SOS ZPAGE
003254 LDR010     LDA      $380,X
003255          STA      SZPAGE,X
003256          DEX
003257          BPL      LDR010
003258          REP      100
003259 * PROCESS KERNEL FILE
003260          REP      100
003261 *
003262 * MOVE AND INITIALIZE SOS GLOBALS
003263 *
003264          LDA      #>LDR.ADR          ; WORK.P:=0:LDR.ADR
003265          STA      WORK.P
003266          LDA      #<LDR.ADR
003267          STA      WORK.P+1
003268          JSR      ADVANCE            ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
003269 *
003270          LDA      B.REG              ; MOVE(SRC.P DST.P A=BREG CNT.IN)
003271          JSR      MOVE
003272 *
003273          LDA      B.REG              ; SYSBANK:=BREG
003274          AND      #$0F
003275          STA      SYSBANK
003276          ASL      A                  ; MEMSIZ:=SYSBANK*2+4 "16K CHUNKS"
003277          CLC
003278          ADC      #4
003279          STA      MEMSIZE            ; AND, MEMSIZE (SIZE IN 16K BYTE "CHUNKS")
003280 *
003281 * MOVE KERNAL CODE
003282 *
003283          JSR      ADVANCE            ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
003284 *
003285          LDA      DST.P              ; K.BASE:=DST.P
003286          STA      K.BASE
003287          LDA      DST.P+1

```

```

003288          STA      K.BASE+1
003289          LDA      B.REG              ; MOVE(SRC.P DST.P A=BREG CNT.IN)
003290          JSR      MOVE
003291      *
003292      * MOVE LOADER TO BANK 0 AND SWITCH FROM SYSTEM BANK TO BANK 0
003293      *
003294          LDA      #>$2000              ; MOVE(SRC.P=0:2000 DST.P=8F:2000 A=BREG CNT=LDR.END-$2000)
003295          STA      SRC.P
003296          STA      DST.P
003297          LDA      #<$2000
003298          STA      SRC.P+1
003299          STA      DST.P+1
003300          LDA      #$8F
003301          STA      CXPAGE+DST.P+1
003302          LDA      #>LDREND-$2000
003303          STA      CNT
003304          LDA      #<LDREND-$2000
003305          STA      CNT+1
003306          LDA      B.REG
003307          JSR      MOVE
003308          LDA      #0                  ; BREG:=0
003309          STA      B.REG
003310      *
003311      * INITIALIZE SDT TABLE, KERNEL AND PRINT WELCOME MESSAGE
003312      *
003313          LDA      K.DRIVES              ; LINK.INIT(A=K.DRIVES DIB1..4.IN, SDT.TBL BLKDLST.IO)
003314          JSR      LINK.INIT
003315          JSR      INIT.KRNL            ; INIT.KRNL()
003316          JSR      WELCOME             ; WELCOME()
003317      *
003318          LDA      E.REG                ; ENABLE ROM BANK
003319          ORA      #$03
003320          STA      E.REG
003321          LDA      ROM.ADR              ; IF MONITOR ROM <> NEW
003322          CMP      #ROM.ID             ; THEN
003323          BEQ      LDR020
003324          LDX      #ERR7X              ; ERROR("ROM ERROR: PLEASE NOTIFY YOUR DEALER")
003325          LDY      #ERR7L
003326          JSR      ERROR
003327      LDR020          LDA      E.REG      ; DISABLE ROM BANK
003328          AND      #$F6
003329          STA      E.REG
003330          REP      100
003331      * PROCESS INTERPRETER FILE
003332          REP      100
003333      *
003334      * OPEN SOS INTERPRETER FILE (DEFAULT='SOS.INTERP')
003335      *
003336          LDY      I.PATH                ; OPEN(PATHNAME:=I.PATH
003337      LDR030          LDA      I.PATH,Y   ; REFNUM=OPEN.REF

```

```

003338          STA      PATH,Y          ;      SYSBUF.P:=80:LDREND-2000 )
003339          DEY
003340          BPL      LDR030
003341 *
003342          LDA      #>LDREND-$2000
003343          STA      SYSBUF.P
003344          LDA      #<LDREND-$2000
003345          STA      SYSBUF.P+1
003346          LDA      #$80
003347          STA      CXPAGE+SYSBUF.P+1
003348 *
003349 *
003350          BRK
003351          DFB      OPEN
003352          DW      OPEN.PARMS
003353          BEQ      LDR040
003354          LDX      #ERR1X          ; ERROR("INTERPRETER FILE NOT FOUND")
003355          LDY      #ERR1L
003356          JSR      ERROR
003357 LDR040      LDA      OPEN.REF
003358          STA      READ.REF
003359          STA      CLOSE.REF
003360 *
003361 * READ IN ENTIRE INTERPRETER FILE
003362 *
003363          LDA      #$80          ; READ(REFNUM=READ.REF
003364          STA      CXPAGE+RDBUF.P+1 ;      RDBUF.P:=80:FILE
003365          LDA      #>FILE          ;      BYTES=$FFFF-FILE+1
003366          STA      RDBUF.P          ;      BYTESRD=I.BYTESRD )
003367          LDA      #<FILE
003368          STA      RDBUF.P+1
003369 *
003370          BRK
003371          DFB      READ
003372          DW      READ.PARMS
003373          BEQ      LDR050
003374          LDX      #ERROX          ; ERROR("I/O ERROR")
003375          LDY      #ERR0L
003376          JSR      ERROR
003377 *
003378 * CLOSE INTERPRETER FILE AND CHECK LABEL          +-----+
003379 *
003380 LDR050      BRK          ; CLOSE(REFNUM=CLOSE.REF)
003381          DFB      CLOSE
003382          DW      CLOSE.PARMS
003383          LDY      #7          ; CHECK LABEL
003384 LDR051      LDA      (RDBUF.P),Y
003385          CMP      I.LABEL,Y
003386          BNE      LDR052
003387          DEY

```

```

003388          BPL          LDR051
003389          BMI          LDR053
003390 LDR052      LDX          #ERR2X          ; ERROR("INVALID INTERPRETER FILE")
003391          LDY          #ERR2L
003392          JSR          ERROR
003393 *
003394 * MOVE INTERPRETER CODE
003395 *
003396 LDR053      LDA          #>I.HDR.CNT-2      ; WORK.P:=80:I.HDR.CNT-2
003397          STA          WORK.P
003398          LDA          #<I.HDR.CNT-2
003399          STA          WORK.P+1
003400          LDA          #$80
003401          STA          CXPAGE+WORK.P+1
003402 *
003403          JSR          ADVANCE          ; ADVANCE(WORK.P.IO, SRC.P DST.P CNT.OUT)
003404 *
003405          LDA          DST.P          ; I.BASE.P:=0:DST.P
003406          STA          I.BASE.P
003407          LDA          DST.P+1
003408          STA          I.BASE.P+1
003409          LDA          #0
003410          STA          CXPAGE+I.BASE.P+1
003411 *
003412          CLC          ; IF DST.P+CNT > K.BASE THEN ERROR
003413          LDA          CNT
003414          ADC          DST.P
003415          TAX
003416          LDA          CNT+1
003417          ADC          DST.P+1
003418          CPX          K.BASE
003419          SBC          K.BASE+1
003420          BEQ          LDR070
003421          BCC          LDR070
003422          LDX          #ERR3X          ; ERROR("INCOMPATIBLE INTERPRETER")
003423          LDY          #ERR3L
003424          JSR          ERROR
003425 *
003426 LDR070      LDA          SYSBANK          ; MOVE(SRC.P=RDBUF.P DST.P A=SYSBANK CNT.IN)
003427          JSR          MOVE
003428          REP          100
003429 * PROCESS DRIVER FILE
003430          REP          100
003431 *
003432 * OPEN SOS DRIVER FILE (DEFAULT='SOS.DRIVER')
003433 *
003434          LDY          D.PATH          ; OPEN(PATHNAME:=D.PATH
003435 LDR080      LDA          D.PATH,Y      ; REFNUM=OPEN.REF
003436          STA          PATH,Y          ; SYSBUF.P:=80:LDREND-2000 )
003437          DEY

```

```

003438          BPL          LDR080
003439  *
003440          BRK
003441          DFB          OPEN
003442          DW          OPEN.PARMS
003443          BEQ          LDR090
003444          LDX          #ERR4X          ; ERROR("DRIVER FILE NOT FOUND")
003445          LDY          #ERR4L
003446          JSR          ERROR
003447  LDR090          LDA          OPEN.REF
003448          STA          READ.REF
003449          STA          CLOSE.REF
003450  *
003451  * READ IN ENTIRE DRIVER FILE INTO BANK 0
003452  *
003453          BRK          ; READ(REFNUM=READ.REF
003454          DFB          READ          ; RDBUF.P:=80:FILE
003455          DW          READ.PARMS          ; BYTES=$FFFF-FILE+1
003456  *          ; BYTESRD=D.BYTESRD )
003457          BEQ          LDR100
003458          LDX          #ERR0X          ; ERROR("I/O ERROR")
003459          LDY          #ERR0L
003460          JSR          ERROR
003461  *          +-----+
003462  * CLOSE THE DRIVER FILE AND CHECK LABEL          ! SEE FIGURE 3. !
003463  *          +-----+
003464  LDR100          BRK          ; CLOSE(REFNUM=CLOSE.REF)
003465          DFB          CLOSE
003466          DW          CLOSE.PARMS
003467          LDY          #$7          ; CHECK LABEL
003468  LDR101          LDA          (RDBUF.P),Y
003469          CMP          D.LABEL,Y
003470          BNE          LDR102
003471          DEY
003472          BPL          LDR101
003473          BMI          LDR103
003474  LDR102          LDX          #ERR5X          ; ERROR("INVALID DRIVER FILE")
003475          LDY          #ERR5L
003476          JSR          ERROR
003477  *
003478  * MOVE CHARACTER SET TABLE
003479  *
003480  LDR103          LDA          #>D.CHRSET          ; MOVE(SRC.P=D.CHRSET DST.P=$C00 A=0 CNT=$400)
003481          STA          SRC.P
003482          LDA          #<D.CHRSET
003483          STA          SRC.P+1
003484          LDA          #>$C00
003485          STA          DST.P
003486          LDA          #<$C00
003487          STA          DST.P+1

```

```

003488         LDA      #>$400
003489         STA      CNT
003490         LDA      #<$400
003491         STA      CNT+1
003492         LDA      #0
003493         JSR      MOVE
003494         *
003495         * MOVE KEYBOARD TABLE
003496         *
003497         LDA      #>D.KYBD          ; MOVE(SRC.P=D.KYBD DST.P=$1700 A=0 CNT=$100.IN)
003498         STA      SRC.P
003499         LDA      #<D.KYBD
003500         STA      SRC.P+1
003501         LDA      #>$1700
003502         STA      DST.P
003503         LDA      #<$1700
003504         STA      DST.P+1
003505         LDA      #>$100
003506         STA      CNT
003507         LDA      #<$100
003508         STA      CNT+1
003509         LDA      #0
003510         JSR      MOVE
003511         *
003512         * RE-INITIALIZE SDT TABLE
003513         *
003514         LDY      #>D.DRIVES-D.FILE ; LINK.INIT(A=D.DRIVES DIB1..4.IN, SDT.TBL BLKDLST.IO)
003515         LDA      (RDBUF.P),Y
003516         JSR      LINK.INIT
003517         *
003518         LDA      #0                ; DST.P:=0:I.BASE.P/256*256
003519         STA      CXPAGE+DST.P+1
003520         STA      DST.P
003521         LDA      I.BASE.P+1
003522         STA      DST.P+1
003523         CMP      #$A0                ; IF DST.P>=$A000 THEN DST.P:=$A000
003524         BCC      LDR105
003525         LDA      #$A0
003526         STA      DST.P+1
003527         LDR105         LDA      SYSBANK          ; DSTBANK:=SYSBANK
003528         STA      DSTBANK
003529         JSR      REVERSE          ; REVERSE(D.HDR.CNT.IN, WORK.P.OUT)
003530         *
003531         * RELOCATE AND MOVE DRIVERS
003532         *
003533         NEXTDRIVER     JSR      DADVANCE          ; "NO DRIVERS LEFT" := DADVANCE(WORK.P.IO SRC.P CNT REL.P.OUT)
003534         BCS      LDR140
003535         JSR      FLAGS              ; "INACTIVE" := FLAGS(SRC.P.IN, PG.ALIGN FIRST.ADIB.OUT)
003536         BVS      NEXTDRIVER
003537         JSR      GETMEM             ; GETMEM(PG.ALIGN CNT.IN, DST.P DSTBANK DSEGLIST.IO, PREVBANK.OUT)

```

```

003538          JSR      RELOC          ; RELOC(SRC.P REL.P DST.P.IN)
003539  *
003540          LDA      DSTBANK        ; IF DSTBANK < 0 OR DST.P < SRC.P THEN ERROR
003541          BMI      LDR120
003542          LDA      CXPAGE+SRC.P+1  ; (CONVERT SRC.P TO BANK SWITCHED ADDRESS)
003543          AND      #$7F
003544          STA      TEMP.BANK
003545          LDA      SRC.P+1
003546          BPL      LDR110
003547          INC      TEMP.BANK
003548  LDR110    AND      #$7F
003549          CLC
003550          ADC      #<$2000
003551          STA      TEMP.ADRH
003552          LDA      DST.P          ; (NOW COMPARE)
003553          CMP      SRC.P
003554          LDA      DST.P+1
003555          SBC      TEMP.ADRH
003556          LDA      DSTBANK
003557          SBC      TEMP.BANK
003558          BCS      LDR130
003559  LDR120    LDX      #ERR6X      ; ERROR("DRIVER FILE TOO LARGE")
003560          LDY      #ERR6L
003561          JSR      ERROR
003562  *
003563  LDR130    LDA      DSTBANK        ; MOVE(SRC.P DST.P A=DSTBANK CNT.IN)
003564          JSR      MOVE
003565          JSR      LINK          ; LINK(DST.P DSTBANK PREVBANK FIRST.ADIB.IN, SDT.TBL BLKDLST.IO)
003566          JMP      NEXTDRIVER
003567          REP      100
003568  * SETUP USER ENVIRONMENT
003569          REP      100
003570  *
003571  * RE-INITIALIZE KERNEL/DRIVERS, ALLOCATE SYSTEM SEGMENTS
003572  *
003573  LDR140    JSR      INIT.KRNL      ; INIT.KRNL()
003574          JSR      ALLOC.SEG      ; ALLOC.SEG(K.BASE I.BASE.P SYSBANK.IN)
003575          JSR      ALLOC.DSEG      ; ALLOC.DSEG(DSEGLIST.IN)
003576  *
003577  * SET PREFIX TO THE BOOT VOLUME
003578  *
003579          LDA      #0              ; TURN VIDEO OFF - PREVENTS CHAR "GROWTH" DURING DOWNLOAD
003580          STA      SCRNMODE
003581          BRK          ; SET.PREFIX(PREFIXPATH=".D1")
003582          DFB      SETPREFIX
003583          DW      PREFX.PARMS
003584  *
003585  * LAUNCH CHARACTER SET DOWNLOAD (CONSOLE) AND CLEAR SCREEN
003586  *
003587          CLI              ; BEGIN CHARACTER SET DOWNLOAD (CONSOLE)

```



```

003588 *
003589         LDA        #0                ; CLEAR TEXT SCREENS
003590         STA        CXPAGE+SRC.P+1
003591         STA        CXPAGE+DST.P+1
003592         LDA        #$04
003593         STA        SRC.P+1
003594         STA        DST.P+1
003595         LDA        #$00
003596         STA        SRC.P
003597         LDA        #$80
003598         STA        DST.P
003599         LDA        #$A0
003600         LDX        #8
003601 CLEAR0   LDY        #$77
003602 CLEAR1   STA        (SRC.P),Y
003603         STA        (DST.P),Y
003604         DEY
003605         BPL        CLEAR1
003606         INC        SRC.P+1          ; NEXT PAGE
003607         INC        DST.P+1          ; NEXT PAGE
003608         DEX
003609         BNE        CLEAR0
003610 *
003611 WAIT    INC        SRC.P            ; WAIT FOR DOWNLOAD TO COMPLETE
003612         BNE        WAIT
003613         INX
003614         BNE        WAIT
003615 *
003616         LDA        #$80                ; TURN VIDEO ON
003617         STA        SCRNMODE
003618         RTS
003619         REP        100
003620
003621         CHN        SOSLDR.E.SRC
003622
003623 *****
003624 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.D.SRC
003625 *****
003626
003627
003628

```

```

003629 =====
003630 DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.E.TEXT
003631 =====
003632
003633 *****
003634 * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.E.SRC
003635 *****
003636 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
003637
003638 PAGE
003639 REP      100
003640 *
003641 * SET.DRIVES ( IN:  A=# DRIVES
003642 *              IN:  DIB1..4   )
003643 * (INITIALIZES DIB LINKS IN KERNEL'S FLOPPY DRIVER)
003644 REP      100
003645 *
003646 SET.DRIVES EQU      *
003647 TAY                      ; SAVE # OF DRIVES
003648 LDA      #>DIB2          ; DIB1:=ADR(DIB2)
003649 STA      DIB1
003650 LDA      #<DIB2
003651 STA      DIB1+1
003652 LDA      #>DIB3          ; DIB2:=ADR(DIB3)
003653 STA      DIB2
003654 LDA      #<DIB3
003655 STA      DIB2+1
003656 LDA      #>DIB4          ; DIB3:=ADR(DIB4)
003657 STA      DIB3
003658 LDA      #<DIB4
003659 STA      DIB3+1
003660 *
003661 LDA      #0                ; CASE (Y=# OF DRIVES)
003662 CPY      #2
003663 BCC      STDR010
003664 BEQ      STDR020
003665 CPY      #4
003666 BCC      STDR030
003667 BCS      STDR040
003668 *
003669 STDR010 STA      DIB1          ; 1: DIB1:=0
003670 STA      DIB1+1
003671 RTS
003672 *
003673 STDR020 STA      DIB2          ; 2: DIB2:=0
003674 STA      DIB2+1
003675 RTS
003676 *
003677 STDR030 STA      DIB3          ; 3: DIB3:=0

```

```

003678          STA      DIB3+1
003679          RTS
003680      *
003681  STDR040      STA      DIB4          ; 4: DIB4:=0
003682          STA      DIB4+1
003683          RTS          ; RETURN
003684          PAGE
003685          REP      100
003686      *
003687      * INIT.KRNL ( )
003688      *
003689      * (CALLS KERNEL INITIALIZATION MODULES)
003690          REP      100
003691      *
003692  INIT.KRNL     EQU      *
003693          LDA      E.REG          ; SWITCH IN I/O BANK AND SELECT PRIMARY STACK
003694          ORA      #$44          ; E:=( 0.1.1.X:0.1.0.0 )
003695          STA      E.REG          ; ( 1.I.S.R:W.P.R.R )
003696      *
003697          LDA      #<SZPAGE      ; SWITCH TO SOS ZPAGE
003698          STA      Z.REG
003699      *
003700          JSR      INT.INIT      ; CALL KERNEL INITIALIZATION ROUTINES
003701          JSR      EVQ.INIT
003702          JSR      BFM.INIT2
003703          BCS      INITK.ERR
003704          JSR      DMGR.INIT
003705          JSR      CFMGR.INIT
003706          JSR      MMGR.INIT
003707          JSR      BMGR.INIT
003708          JSR      BFM.INIT
003709          JSR      CLK.INIT
003710      *
003711          LDA      E.REG          ; SWITCH OUT I/O BANK AND RETURN TO ALTERNATE STACK
003712          AND      #$BB          ; E:=( 0.0.1.X:0.0.0.0 )
003713          STA      E.REG          ; ( 1.I.S.R:W.P.R.R )
003714      *
003715          LDA      #<CZPAGE      ; SWITCH BACK TO USER ZPAGE
003716          STA      Z.REG
003717      *
003718          RTS          ; RETURN
003719      *
003720      *
003721  INITK.ERR      LDX      #ERRORX      ; ERROR("I/O ERROR")
003722          LDY      #ERROL
003723          JMP      ERROR
003724          PAGE
003725          REP      100
003726      *
003727      * ADVANCE ( I/O: WORK.P

```

```

003728 *          OUT:  SRC.P
003729 *          OUT:  DST.P
003730 *          OUT:  CNT      )
003731 * (ADVANCES WORK.P TO NEXT INTERP.KERNEL MODULE.  INITS SRC.P, DST.P, CNT FOR MOVE)
003732          REP          100
003733 *
003734 ADVANCE          EQU          *
003735                  CLC
003736                  LDY          #2          ; Y:=0
003737                  LDA          WORK.P      ; WORK.P:=WORK.P+(WORK.P),Y + 4
003738                  ADC          (WORK.P),Y
003739                  TAX
003740                  INY
003741                  LDA          WORK.P+1
003742                  ADC          (WORK.P),Y
003743                  PHA
003744                  TXA
003745                  ADC          #4
003746                  STA          WORK.P
003747                  PLA
003748                  ADC          #0
003749                  STA          WORK.P+1
003750                  CLC          ; SRC.P:=X:WORK.P+4
003751                  LDA          WORK.P
003752                  ADC          #>$0004
003753                  STA          SRC.P
003754                  LDA          WORK.P+1
003755                  ADC          #<$0004
003756                  STA          SRC.P+1
003757                  LDA          CXPAGE+WORK.P+1
003758                  STA          CXPAGE+SRC.P+1
003759                  LDY          #0          ; DST.P:=0:(WORK.P)
003760                  STY          CXPAGE+DST.P+1
003761                  LDA          (WORK.P),Y
003762                  STA          DST.P
003763                  INY
003764                  LDA          (WORK.P),Y
003765                  STA          DST.P+1
003766                  INY          ; Y:=2
003767                  LDA          (WORK.P),Y      ; CNT:=(WORK.P),Y
003768                  STA          CNT
003769                  INY
003770                  LDA          (WORK.P),Y
003771                  STA          CNT+1
003772                  RTS          ; RETURN
003773                  PAGE
003774                  REP          100
003775 *
003776 * REVERSE ( IN:   D.HDR.CNT
003777 *          IN:   SDT.SIZE = CONSTANT

```

```

003778 *           I/O:  DRIVER FILE,
003779 *           OUT:  WORK.P      )      )
003780 *
003781 *           LOCAL:  REV.SAVE, REV.TEMP
003782 * (REVERSES TITLE/CODE/RELOC COUNTS TO ALLOW DRIVER FILE TO BE PROCESSED FROM BACK TO FRONT)
003783             REP           100
003784 REVERSE     EQU           *
003785             LDA           #>D.HDR.CNT      ; WORK.P:=80:D.HDR.CNT
003786             STA           WORK.P
003787             LDA           #<D.HDR.CNT
003788             STA           WORK.P+1
003789             LDA           #$80
003790             STA           CXPAGE+WORK.P+1
003791             CLC           ; WORK.P:=WORK.P+(WORK.P)+2
003792             LDY           #0
003793             LDA           WORK.P
003794             ADC           (WORK.P),Y
003795             TAX
003796             INY
003797             LDA           WORK.P+1
003798             ADC           (WORK.P),Y
003799             PHA
003800             TXA
003801             ADC           #2
003802             STA           WORK.P
003803             PLA
003804             ADC           #0
003805             STA           WORK.P+1
003806             LDA           (WORK.P),Y      ; IF (WORK.P)=$FFFF
003807             DEY
003808             AND           (WORK.P),Y      ; THEN
003809             CMP           #$FF
003810             BNE           REV010
003811             LDX           #ERR10X      ; ERROR("EMPTY DRIVER FILE")
003812             LDY           #ERR10L
003813             JSR           ERROR
003814 REVERSE     LDA           #$FF
003815             STA           REV.SAVE
003816             STA           REV.SAVE+1
003817 *
003818 REV020      LDA           REV.SAVE      ;R1: STACK:=REV.SAVE
003819             PHA
003820             LDA           REV.SAVE+1
003821             PHA
003822             LDY           #0      ; REV.SAVE:=(WORK.P)
003823             LDA           (WORK.P),Y
003824             STA           REV.SAVE
003825             INY
003826             LDA           (WORK.P),Y
003827             STA           REV.SAVE+1

```

```

003828          PLA                ; (WORK.P) := STACK
003829          STA          (WORK.P),Y
003830          DEY
003831          PLA
003832          STA          (WORK.P),Y
003833          LDA          REV.SAVE          ; IF REV.SAVE = $FFFF THEN EXIT
003834          AND          REV.SAVE+1
003835          CMP          #$FF
003836          BEQ          REV.EXIT
003837  REV030          BIT          REV.SAVE+1          ; IF REV.SAVE >= $8000 THEN ERROR
003838          BMI          REV040
003839          CLC                ; WORK.P := WORK.P + REV.SAVE + 2
003840          LDA          WORK.P
003841          ADC          REV.SAVE
003842          TAX
003843          LDA          WORK.P+1
003844          ADC          REV.SAVE+1
003845          PHA
003846          BCS          REV040
003847          TXA
003848          ADC          #2
003849          STA          WORK.P
003850          PLA
003851          ADC          #0
003852          STA          WORK.P+1
003853          BCC          REV020          ; IF C=FALSE THEN R1
003854  REV040          LDX          #ERR5X          ; ELSE ERROR("INVALID DRIVER FILE")
003855          LDY          #ERR5L
003856          JSR          ERROR
003857          *
003858  REV.EXIT          RTS                ; RETURN
003859          PAGE
003860          REP          100
003861          *
003862          * DADVANCE ( I/O: WORK.P
003863          *          OUT: C="NO DRIVERS LEFT"
003864          *          OUT: SRC.P
003865          *          OUT: CNT
003866          *          OUT: REL.P )
003867          * (ADVANCES WORK.P TO NEXT DRIVER MODULE.  INITS SRC.P, CNT, REL.P FOR RELOCATION AND MOVE)
003868          REP          100
003869  DADVANCE          EQU          *
003870          LDY          #0                ; IF (WORK.P)=$FFFF THEN EXIT "NO DRIVERS LEFT IN FILE"
003871          LDA          (WORK.P),Y
003872          INY
003873          AND          (WORK.P),Y
003874          CMP          #$FF
003875          BNE          DADV010
003876          SEC                ; C:="NO DRIVERS LEFT"
003877          RTS                ; RETURN

```

```

003878 *
003879 *
003880 DADV010      LDA      WORK.P          ; REL.P:=X:WORK.P
003881            STA      REL.P
003882            LDA      WORK.P+1
003883            STA      REL.P+1
003884            LDA      CXPAGE+WORK.P+1
003885            STA      CXPAGE+REL.P+1
003886 *
003887            JSR      DADD              ; ADVANCE TO CODE COUNT FIELD
003888 *
003889            LDY      #0                ; CNT:=(WORK.P)
003890            LDA      (WORK.P),Y
003891            STA      CNT
003892            INY
003893            LDA      (WORK.P),Y
003894            STA      CNT+1
003895 *
003896            JSR      DADD              ; ADVANCE TO TITLE CNT FIELD
003897 *
003898            CLC                          ; SRC.P:=X:WORK.P+2
003899            LDA      WORK.P
003900            ADC      #2
003901            STA      SRC.P
003902            LDA      WORK.P+1
003903            ADC      #0
003904            STA      SRC.P+1
003905            LDA      CXPAGE+WORK.P+1
003906            STA      CXPAGE+SRC.P+1
003907 *
003908            JSR      DADD              ; ADVANCE TO RELOC FIELD OF NEXT DRIVER
003909            CLC                          ; C:="DRIVERS LEFT"
003910            RTS                          ; RETURN
003911            PAGE
003912            REP      100
003913 *
003914 * DADD ( I/O:  WORK.P )
003915 *
003916 * (ADVANCES WORK.P TO NEXT FIELD IN DRIVER MODULE)
003917            REP      100
003918 DADD            EQU      *
003919            SEC                          ; WORK.P:=WORK.P-(WORK.P)-2
003920            LDY      #0
003921            LDA      WORK.P
003922            SBC      (WORK.P),Y
003923            TAX
003924            INY
003925            LDA      WORK.P+1
003926            SBC      (WORK.P),Y
003927            PHA

```

```

003928          TXA
003929          SBC          #2
003930          STA          WORK.P
003931          PLA
003932          SBC          #0
003933          STA          WORK.P+1
003934          RTS                          ; RETURN
003935          PAGE
003936          REP          100
003937 *
003938 *  FLAGS ( IN:  SRC.P
003939 *          OUT: PG.ALIGN
003940 *          OUT: FIRST.ADIB
003941 *          OUT: OV="ALL DIBS INACTIVE" )
003942 *
003943 *          LOCAL:  PREV.ADIB.P, DIB.P
003944 * (PROCESSES "INACTIVE" & "PAGE ALIGN" FLAGS IN DRIVER MODULE'S DIBS"
003945          REP          100
003946  FLAGS          EQU          *
003947          SEC                          ; C="FIRST DIB"
003948  FLAG010        JSR          NEXT.DIB      ; NEXT.DIB(SRC.P.IN, DIB.P PG.ALIGN C OV.OUT)
003949          BVC          FLAG015          ; IF OV <> "INACTIVE" THEN ACTIVE DIB FOUND
003950          BCC          FLAG010          ; IF C <> "LAST DIB" THEN CHECK NEXT DIB
003951          RTS                          ; RETURN (OV="ALL DIBS INACTIVE")
003952 *
003953  FLAG015        PHP                          ; PUSH STATUS
003954          SEC                          ; FIRST.ADIB:=DIB.P-SRC.P
003955          LDA          DIB.P
003956          SBC          SRC.P
003957          STA          FIRST.ADIB
003958          LDA          DIB.P+1
003959          SBC          SRC.P+1
003960          STA          FIRST.ADIB+1
003961          LDA          DIB.P              ; PREV.ADIB.P:=X:DIB.P
003962          STA          PREV.ADIB.P
003963          LDA          DIB.P+1
003964          STA          PREV.ADIB.P+1
003965          LDA          CXPAGE+DIB.P+1
003966          STA          CXPAGE+PREV.ADIB.P+1
003967          PLP                          ; PULL STATUS
003968          BCS          FLAG100          ; IF C="LAST DIB" THEN EXIT
003969 *
003970  FLAG020        JSR          NEXT.DIB      ; NEXT.DIB(SRC.P.IN, DIB.P PG.ALIGN C OV.OUT)
003971          PHP                          ; PUSH STATUS
003972          LDY          #0                ; IF OV="INACTIVE DIB"
003973          BVC          FLAG025
003974          SEC                          ; THEN
003975          LDA          PREV.ADIB.P        ; (PREV.ADIB.P):=PREV.ADIB.P-SRC.P
003976          SBC          SRC.P
003977          STA          (PREV.ADIB.P),Y

```



```

003978          INY
003979          LDA          PREV.ADIB.P+1
003980          SBC          SRC.P+1
003981          STA          (PREV.ADIB.P),Y
003982          JMP          FLAG050
003983 *
003984 FLAG025      SEC                      ; ELSE
003985          LDA          DIB.P              ; (PREV.ADIB.P):=DIB.P-SRC.P
003986          SBC          SRC.P
003987          STA          (PREV.ADIB.P),Y
003988          INY
003989          LDA          DIB.P+1
003990          TAX
003991          SBC          SRC.P+1
003992          STA          (PREV.ADIB.P),Y
003993          STX          PREV.ADIB.P+1    ; PREV.ADIB.P:=DIB.P
003994          LDA          DIB.P
003995          STA          PREV.ADIB.P
003996 FLAG050      PLP                      ; PULL STATUS
003997          BCC          FLAG020        ; IF C <> "LAST DIB" THEN PROCESS NEXT DIB
003998 *
003999 FLAG100      CLV                      ; OV:="ACTIVE DIBS"
004000          RTS                      ; RETURN
004001          PAGE
004002          REP          100
004003 *
004004 * NEXT.DIB ( IN:  C="FIRST DIB"
004005 *              IN:  SRC.P
004006 *              OUT: DIB.P
004007 *              OUT: PG.ALIGN
004008 *              OUT: C="LAST DIB"
004009 *              OUT: OV="INACTIVE DIB" )
004010 *
004011 *          LOCAL:  DIB.FLAGS, DIB.DCB = CONSTANT
004012 * (ADVANCES TO NEXT DIB IN DRIVER MODULE)
004013          REP          100
004014 NEXT.DIB     EQU          *
004015          LDY          #0
004016          BCC          NXTD010          ; IF C = "FIRST DIB"
004017          STY          PG.ALIGN        ; THEN
004018          STY          PG.ALIGN+1      ; PG.ALIGN:=0
004019          LDA          SRC.P          ; DIB.P:=X:SRC.P
004020          STA          DIB.P
004021          LDA          SRC.P+1
004022          STA          DIB.P+1
004023          LDA          CXPAGE+SRC.P+1
004024          STA          CXPAGE+DIB.P+1
004025          JMP          NXTD020
004026 NXTD010     LDA          SRC.P          ; ELSE
004027          ADC          (DIB.P),Y        ; DIB.P:=SRC.P+(DIB.P)

```

```

004028          TAX
004029          INY
004030          LDA          SRC.P+1
004031          ADC          (DIB.P),Y
004032          STA          DIB.P+1
004033          STX          DIB.P
004034          *
004035  NXTD020          LDY          #DIB.FLAGS          ; IF (DIB.P),DIB.FLAGS.BIT7 = "INACTIVE"
004036          LDA          (DIB.P),Y
004037          BMI          NXTD030
004038          BIT          NXTD999          ; THEN
004039          BVS          NXTD040          ; OV:="INACTIVE"
004040          *
004041  NXTD030          AND          #$40          ; ELSE
004042          BEQ          NXTD040          ; IF (DIB.P),DIB.FLAGS.BIT6 = "PAGE ALIGN"
004043          CLC
004044          LDA          #DIB.DCB+2          ; THEN
004045          TAY          ; PAGE.ALIGN:=DIB.DCB+2+(SRC.P),DIB.DCB
004046          DEY
004047          DEY
004048          ADC          (SRC.P),Y
004049          STA          PG.ALIGN
004050          INY
004051          LDA          #0
004052          ADC          (SRC.P),Y
004053          STA          PG.ALIGN+1
004054          CLV          ; OV:="ACTIVE"
004055          *
004056  NXTD040          LDY          #0          ; IF (DIB.P) = 0
004057          LDA          (DIB.P),Y
004058          INY
004059          ORA          (DIB.P),Y
004060          BNE          NXTD998
004061          SEC          ; THEN C:="LAST DIB"
004062          BCS          NXTD999
004063  NXTD998          CLC          ; ELSE C:=NOT "LAST DIB"
004064  NXTD999          RTS          ; RETURN
004065          REP          100
004066
004067          CHN          SOSLDR.F.SRC
004068
004069          RTS          ; RETURN
004070
004071          *****
004072          * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.E.SRC
004073          *****
004074
004075

```

```

004076 =====
004077 DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.F.SRC.TEXT
004078 =====
004079
004080 *****
004081 * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.F.SRC
004082 *****
004083 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
004084
004085             PAGE
004086             REP          100
004087 *
004088 * GETMEM ( IN:   PG.ALIGN
004089 *           IN:   CNT
004090 *           I/O:  DST.P
004091 *           I/O:  DSTBANK
004092 *           I/O:  DSEGLIST
004093 *           OUT:  PREVBANK )
004094 *
004095 *           LOCAL:  PREVDST
004096 * (COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT)
004097             REP          100
004098 GETMEM      EQU          *
004099             LDA          DSTBANK           ; PREVBANK:=DSTBANK
004100             STA          PREVBANK
004101             LDA          DST.P             ; PREVDST:=DST.P
004102             STA          PREVDST
004103             LDA          DST.P+1
004104             STA          PREVDST+1
004105             JSR          NEWDST           ; NEWDST(PG.ALIGN.IN, PREVDST.IN, CNT.IN, DST.P.OUT)
004106 *
004107             LDA          DST.P+1           ; IF DST.P >= $2000
004108             CMP          #$20
004109             BCC          GETM010
004110             SEC                               ; THEN
004111             LDA          PREVDST+1         ; A=PAGES:=PREVDST-DST.P
004112             SBC          DST.P+1
004113             CLC
004114             JSR          BUILD.DSEG        ; BUILD.DSEG(C="NEXT BANK".IN, A=PAGES.IN, DSEGLIST.IO)
004115             JMP          GETM.EXIT
004116 *
004117 GETM010     DEC          DSTBANK           ; DSTBANK:=DSTBANK-1
004118             LDA          #>$A000         ; PREVDST:=$A000
004119             STA          PREVDST
004120             LDA          #<$A000
004121             STA          PREVDST+1
004122             JSR          NEWDST           ; NEWDST(PG.ALIGN.IN, PREVDST.IN, CNT.IN, DST.P.OUT)
004123             SEC                               ; A="PAGES" :=PREVDST-DST.P
004124             LDA          PREVDST+1

```

```

004125          SBC      DST.P+1
004126          SEC
004127          JSR      BUILD.DSEG          ;      BUILD.DSEG(C="NEXTBANK".IN, A="PAGES".IN, DSEGLIST.IO)
004128 *
004129 GETM.EXIT   RTS          ; RETURN
004130          PAGE
004131          REP      100
004132 *
004133 * NEWDST ( IN:  PG.ALIGN
004134 *          IN:  PREVDST
004135 *          IN:  CNT
004136 *          I/O: DST.P      )
004137 * (COMPUTES DESTINATION BASE ADDRESS, ALIGNING ON PAGE BOUNDARY IF REQUESTED)
004138          REP      100
004139 NEWDST      EQU      *
004140          SEC          ; IF (PREVDST-$2000) < CNT
004141          LDA      PREVDST
004142          SBC      #>$2000
004143          TAX
004144          LDA      PREVDST+1
004145          SBC      #<$2000
004146          CPX      CNT
004147          SBC      CNT+1
004148          BCS      NEWD010
004149          LDA      #0          ; THEN
004150          STA      DST.P      ;      DST.P:=0
004151          STA      DST.P+1
004152          BEQ      NEWD.EXIT
004153 NEWD010     SEC          ; ELSE
004154          LDA      PREVDST      ;      DST.P:=PREVDST-CNT
004155          SBC      CNT
004156          STA      DST.P
004157          LDA      PREVDST+1
004158          SBC      CNT+1
004159          STA      DST.P+1
004160          LDA      PG.ALIGN      ;      IF PG.ALIGN <> 0
004161          ORA      PG.ALIGN+1    ;      THEN
004162          BEQ      NEWD.EXIT
004163          SEC          ;      DST.P:=(DST.P/256*256)-PG.ALIGN
004164          LDA      #0
004165          SBC      PG.ALIGN
004166          STA      DST.P
004167          LDA      DST.P+1
004168          SBC      PG.ALIGN+1
004169          STA      DST.P+1
004170 NEWD.EXIT   RTS          ; RETURN
004171          PAGE
004172          REP      100
004173 *
004174 * BUILD.DSEG ( IN:  C="NEXTBANK"

```

```

004175 *           IN:   A="PAGES"
004176 *           I/O:  DSEGLIST  )
004177 * (COMPUTES # OF PAGES TO ADD TO DRIVER SEGMENT AND WHETHER TO BEGIN A NEW SEGMENT)
004178           REP           100
004179 BUILD.DSEG   EQU           *
004180           PHA
004181           BCS           BLDS010           ; IF ("NEXTBANK"=TRUE OR DSEGX=$FF)
004182           LDA           DSEGX           ; THEN
004183           BPL           BLDS020
004184 BLDS010       INC           DSEGX           ;           DSEGX:=DSEGX+1
004185 BLDS020       LDX           DSEGX
004186           CLC           ; DSEGLIST(DSEGX):=DSEGLIST(DSEGX)+"PAGES"
004187           PLA
004188           ADC           DSEGLIST,X
004189           STA           DSEGLIST,X
004190           RTS           ; RETURN
004191 *
004192 *
004193 *
004194 DSEGX          DFB          $FF           ; DRIVER SEGMENT LIST TABLE
004195 DSEGLIST       DFB          $0           ; # PAGES FOR 1ST DRIVER SEGMENT (BANK N )
004196           DFB          $0           ;           " 2ND           " (BANK N-1)
004197           DFB          $0           ;           " 3RD           " (BANK N-2)
004198           DFB          $0           ;           " 4TH           " (BANK N-3)
004199           PAGE
004200           REP           100
004201 *
004202 * RELOC ( IN:   SRC.P
004203 *           IN:   REL.P
004204 *           IN:   DST.P
004205 *           OUT:  RELOCATED DRIVER MODULE )
004206 *
004207 *           LOCAL:  REL.END, CODE.P
004208 * (RELOCATES DRIVER MODULE'S CODE FIELD USING RELOCATION FIELD)
004209           REP           100
004210 RELOC          EQU           *
004211           SEC           ; REL.END:=REL.P-(REL.P)
004212           LDY           #0
004213           LDA           REL.P
004214           SBC           (REL.P),Y
004215           STA           REL.END
004216           INY
004217           LDA           REL.P+1
004218           SBC           (REL.P),Y
004219           STA           REL.END+1
004220 REL.LOOP      SEC           ; REL.P:=REL.P-2
004221           LDA           REL.P
004222           SBC           #2
004223           STA           REL.P
004224           LDA           REL.P+1

```

```

004225          SBC          #0
004226          STA          REL.P+1
004227          LDA          REL.P          ; IF REL.P < REL.END THEN EXIT
004228          CMP          REL.END
004229          LDA          REL.P+1
004230          SBC          REL.END+1
004231          BCC          REL.EXIT
004232          LDY          #0          ; CODE.P:=X:SRC.P+(REL.P)
004233          CLC
004234          LDA          SRC.P
004235          ADC          (REL.P),Y
004236          STA          CODE.P
004237          INY
004238          LDA          SRC.P+1
004239          ADC          (REL.P),Y
004240          STA          CODE.P+1
004241          LDA          CXPAGE+SRC.P+1
004242          STA          CXPAGE+CODE.P+1
004243          LDY          #0          ; (CODE.P):=(CODE.P)+DST.P
004244          CLC
004245          LDA          (CODE.P),Y
004246          ADC          DST.P
004247          STA          (CODE.P),Y
004248          INY
004249          LDA          (CODE.P),Y
004250          ADC          DST.P+1
004251          STA          (CODE.P),Y
004252          JMP          REL.LOOP          ; GOTO REL.LOOP
004253          *
004254          REL.EXIT      RTS          ; RETURN
004255          PAGE
004256          REP          100
004257          *
004258          * ALLOC.SEG ( IN:  K.BASE
004259          *                IN:  I.BASE.P
004260          *                IN:  SYSBANK )
004261          *                I.BASE.P
004262          *                D.BASE.PG
004263          * (ALLOCATES SEGMENTS FOR KERNEL, INTERPRETER AND SYSTEM WORK AREA)
004264          REP          100
004265          ALLOC.SEG      EQU          *
004266          BRK          ; REQ.SEG(BASE=(F,0), LIMIT=(F,1D), SEGID=0, SEGNUM)
004267          DFB          REQSEG
004268          DW          SEGMENT
004269          *
004270          LDA          #$10          ; SET BASE/LIMIT BANKS
004271          STA          SEGBASE
004272          STA          SEGLIM
004273          LDA          #0          ; AND INIT BASE PAGE
004274          STA          SEGBASE+1

```

```

004275 *
004276         LDX      K.BASE+1          ; KERNEL SEGMENT, ID=1
004277         JSR      RSEG
004278 *
004279         LDX      I.BASE.P+1        ; INTERPRETER SEGMENT, ID=2
004280         JSR      RSEG
004281         RTS
004282         PAGE
004283         REP      100
004284 *
004285 * RSEG ( IN:  X=BASE.PAGE OF SEGMENT )
004286 *
004287         REP      100
004288 RSEG     EQU      *
004289         INC      SEGID              ; SEGID:=SEGID+1
004290         LDY      SEGBASE+1         ; LIMIT.PAGE:=BASE.PAGE-1
004291         DEY
004292         STY      SEGLIM+1
004293         STX      SEGBASE+1         ; BASE.PAGE:=X
004294 *
004295         CPX      #$A0              ; IF BASE>=$A0 OR LIMIT<$A0 THEN
004296         BCS      RSEG010           ;     THEN
004297         LDA      SEGLIM+1          ;     REQUEST ONLY ONE SEGMENT
004298         CMP      #$A0
004299         BCC      RSEG010
004300 *
004301         TXA                      ;     ELSE
004302         PHA                      ;     REQUEST TWO SEGMENTS
004303         LDX      #$A0
004304         STX      SEGBASE+1
004305 *
004306         BRK                      ;     REQ.SEG(BASE, LIMIT, SEGID, SEGNUM)
004307         DFB      REQSEG
004308         DW       SEGMENT
004309 *
004310         PLA
004311         STA      SEGBASE+1
004312         LDA      #$9F
004313         STA      SEGLIM+1
004314         LDA      SYSBANK
004315         STA      SEGBASE
004316         STA      SEGLIM
004317 *
004318 *
004319 RSEG010   BRK                      ; REQ.SEG(BASE, LIMIT, SEGID, SEGNUM)
004320         DFB      REQSEG
004321         DW       SEGMENT
004322 *
004323         RTS                      ; RETURN
004324         PAGE

```

```

004325          REP          100
004326 *
004327 * ALLOC.DSEG ( IN:  DSEGLIST )
004328 *
004329 * (ALLOCATES SEGMENTS FOR DRIVER MODULES"
004330          REP          100
004331 ALLOC.DSEG          EQU          *
004332          INC          DSEGX          ; DSEGX:=DSEGX+1
004333          BNE          ALDS010        ; IF DSEGX=0
004334          LDX          #ERR5X        ; THEN ERROR("INVALID DRIVER FILE")
004335          LDY          #ERR5L
004336          JSR          ERROR
004337 *
004338 ALDS010          LDY          #$FF          ; Y:=-1
004339 ALDS020          INY          ; WHILE (Y:=Y+1) < DSEGX
004340          CPY          DSEGX          ; DO
004341          BCS          ALDS.EXIT
004342          LDA          DSEGLIST,Y        ; PAGECT:=DSEGLIST(Y)
004343          STA          SEGPGCNT
004344          BRK          ; FINDSEG (SRCHMODE=0.IN, SEGID=3.IN
004345          DFB          FINDSEG          ; PAGECT=DSEGLIST(Y)
004346          DW          SEGMENT1        ; BASE.OUT, LIMIT.OUT)
004347          JMP          ALDS020
004348 *
004349 ALDS.EXIT          RTS          ; RETURN
004350          PAGE
004351          REP          100
004352 *
004353 * ERROR (IN:  X=MESSAGE INDEX
004354 *          IN:  Y=MESSAGE LENGTH
004355 * (DISPLAYS ERROR MESSAGE, SOUNDS BELL AND LOOPS UNTIL CONTROL/RESET PRESSED)
004356          REP          100
004357 ERROR          EQU          *
004358          STY          ETEMP          ; CENTER MSG (Y:=LEN/2+LEN)
004359          SEC
004360          LDA          #40
004361          SBC          ETEMP
004362          LSR          A
004363          CLC
004364          ADC          ETEMP
004365          TAY
004366 *
004367 PRNT010          LDA          ERR,X          ; MOVE MESSAGE TO SCREEN MEMORY
004368          STA          EMSGADR-1,Y
004369          DEX
004370          DEY
004371          DEC          ETEMP
004372          BNE          PRNT010
004373 *
004374          LDA          #$73          ; E:=( 0.1.1.1:0.0.1.1 )

```



```

004375          STA          E.REG          ; ( 1.I.S.R:W.P.R.S )
004376          LDA          $C040         ; SOUND BELL
004377          JMP          *              ; LOOP UNTIL REBOOT (CTRL/RESET)
004378          PAGE
004379          REP          100
004380          *
004381          * ERROR MESSAGES
004382          *
004383          REP          100
004384          MSGADR       EQU          $7A8
004385          *
004386          ERR          EQU          *
004387          ERR0         ASC          "I/O ERROR"
004388          ERR0L        EQU          *-ERR0
004389          ERR0X        EQU          *-ERR-1
004390          ERR1         ASC          "INTERPRETER FILE NOT FOUND"
004391          ERR1L        EQU          *-ERR1
004392          ERR1X        EQU          *-ERR-1
004393          ERR2         ASC          "INVALID INTERPRETER FILE"
004394          ERR2L        EQU          *-ERR2
004395          ERR2X        EQU          *-ERR-1
004396          ERR3         ASC          "INCOMPATIBLE INTERPRETER"
004397          ERR3L        EQU          *-ERR3
004398          ERR3X        EQU          *-ERR-1
004399          ERR4         ASC          "DRIVER FILE NOT FOUND"
004400          ERR4L        EQU          *-ERR4
004401          ERR4X        EQU          *-ERR-1
004402          ERR5         ASC          "INVALID DRIVER FILE"
004403          ERR5L        EQU          *-ERR5
004404          ERR5X        EQU          *-ERR-1
004405          ERR6         ASC          "DRIVER FILE TOO LARGE"
004406          ERR6L        EQU          *-ERR6
004407          ERR6X        EQU          *-ERR-1
004408          ERR7         ASC          "ROM ERROR: PLEASE NOTIFY YOUR DEALER"
004409          ERR7L        EQU          *-ERR7
004410          ERR7X        EQU          *-ERR-1
004411          ERR8         ASC          "TOO MANY DEVICES"
004412          ERR8L        EQU          *-ERR8
004413          ERR8X        EQU          *-ERR-1
004414          ERR9         ASC          "TOO MANY BLOCK DEVICES"
004415          ERR9L        EQU          *-ERR9
004416          ERR9X        EQU          *-ERR-1
004417          ERR10        ASC          "EMPTY DRIVER FILE"
004418          ERR10L       EQU          *-ERR10
004419          ERR10X       EQU          *-ERR-1
004420          PAGE
004421          REP          100
004422          *
004423          * WELCOME ( )
004424          *

```

```

004425 * (PRINTS WELCOME MESSAGE - "APPLE ///", VERSION, DATE/TIME, COPYRIGHT)
004426          REP          100
004427 WELCOME          EQU          *
004428 *
004429 * PRINT "APPLE III" MESSAGE
004430 *
004431          LDY          #AMSGL
004432 WAM010          LDA          MSG-1,Y
004433          STA          MSGADR-1,Y
004434          DEY
004435          BNE          WAM010
004436 *
004437 * PRINT SOS VERSION MESSAGE
004438 *
004439          CLC
004440          LDA          #40
004441          ADC          #>SOSVERL
004442          LSR          A
004443          TAX
004444          LDY          #>SOSVERL
004445 WSM010          LDA          SOSVER-1,Y
004446          ORA          #$80
004447          STA          SMSGADR-1,X
004448          DEX
004449          DEY
004450          BNE          WSM010
004451 *
004452 * PRINT DATE AND TIME MESSAGE
004453 *
004454          BRK                                ; GET.TIME(TIME.OUT)
004455          DFB          GETTIME
004456          DW          DTPARMS
004457 *
004458          LDA          DATETIME+8          ;SET UP WEEKDAY
004459          AND          #$0F
004460          BEQ          WDM040          ;NO CLOCK
004461          STA          WTEMP
004462          ASL          A
004463          ADC          WTEMP
004464          TAX
004465          LDY          #3
004466 WDM010          LDA          DAYNAME-1,X
004467          STA          DMSG-1,Y
004468          DEX
004469          DEY
004470          BNE          WDM010
004471 *
004472          LDA          DATETIME+7          ;SET UP DATE
004473          LDX          DATETIME+6
004474          STA          DMSG+6

```

```

004475          STX          DMSG+5
004476  *
004477          LDA          DATETIME+5          ;SET UP MONTH
004478          AND          #$0F
004479          LDX          DATETIME+4
004480          CPX          #$31
004481          BCC          WDM020
004482          ADC          #9
004483  WDM020          STA          WTEMP
004484          ASL          A
004485          ADC          WTEMP
004486          TAX
004487          LDY          #3
004488  WDM030          LDA          MONNAME-1,X
004489          STA          DMSG+7,Y
004490          DEX
004491          DEY
004492          BNE          WDM030
004493  *
004494          LDA          DATETIME+3          ;SET UP YEAR
004495          LDX          DATETIME+2
004496          STA          DMSG+13
004497          STX          DMSG+12
004498  *
004499          LDA          DATETIME+10         ;SET UP HOUR
004500          LDX          DATETIME+09
004501          STA          DMSG+17
004502          STX          DMSG+16
004503  *
004504          LDA          DATETIME+12         ;SET UP MINUTE
004505          LDX          DATETIME+11
004506          STA          DMSG+20
004507          STX          DMSG+19
004508  *
004509          LDY          #DMSGL              ;PRINT DATE & TIME
004510  WDM050          LDA          DMSG-1,Y
004511          ORA          #$80
004512          STA          DMSGADR-1,Y
004513          DEY
004514          BNE          WDM050
004515  *
004516  *  PRINT COPYRIGHT MESSAGE
004517  *
004518  WDM040          LDY          #CMSGL
004519  WCM010          LDA          CMSG-1,Y
004520          STA          CMSGADR-1,Y
004521          DEY
004522          BNE          WCM010
004523          RTS
004524          PAGE

```

```

004525          REP          100
004526 *
004527 * WELCOME ( ) - DATA DECLARATIONS
004528 *
004529          REP          100
004530          MSB          ON
004531  AMMSG          ASC          "APPLE ///"
004532  AMMSG          EQU          *-AMMSG
004533  AMMSGADR       EQU          40-AMMSG/2+$4A8
004534          MSB          OFF
004535  SMSGADR        EQU          $5A8
004536  DMSG          ASC          "DAY, DD-MON-YY  HH:MM"
004537  DMSG          EQU          *-DMSG
004538  DMSGADR        EQU          40-DMSG/2+$6A8
004539  DAYNAME       ASC          "SUNMONTUEWEDTHUFRISAT"
004540  MONNAME       ASC          "JANFEBMARAPRMAYJUN"
004541          ASC          "JULAUGSEPCTNOVDEC"
004542          MSB          ON
004543  CMSG          ASC          "(C)1980,1981,1982 BY APPLE COMPUTER INC."
004544  CMSG          EQU          *-CMSG
004545  CMSGADR        EQU          40-CMSG/2+$7D0
004546          MSB          OFF
004547          PAGE
004548          REP          100
004549 *
004550 * SOS SYSTEM CALLS (1)
004551 *
004552          REP          100
004553 * OPEN (PATHNAME.IN, REFNUM.OUT, OPENLIST.IN, OPENCNT.IN) ** (ACCESS.IN, PAGES.IN, SYSBUF.IN)
004554          REP          100
004555  OPEN          EQU          $C8
004556 *
004557  OPEN.PARMS     DFB          $4
004558          DW          PATH
004559  OPEN.REF       DFB          $0
004560          DW          OPEN.LIST
004561          DFB          $4
004562  OPEN.LIST     DFB          $0,$4          ; PAGES:=4
004563          DW          SYSBUF.P
004564  PATH          DS          $40          ; PATHNAME BUFFER
004565  I.LABEL        ASC          "SOS NTRP"          ; FILE LABELS
004566  D.LABEL        ASC          "SOS DRVR"
004567          REP          100
004568 * READ (REFNUM.IN, BUFFER.IN, BYTES.IN, BYTESREAD.OUT)
004569          REP          100
004570  READ          EQU          $CA
004571 *
004572  READ.PARMS     DFB          $4
004573  READ.REF       DFB          $0
004574  READ.BUF       DW          RDBUF.P

```

```

004575 READ.BYT      DW      $FFFF-FILE+1
004576 READ.BYTRD   DW      $0
004577              REP      100
004578 * CLOSE (REFNUM.IN)
004579              REP      100
004580 CLOSE        EQU      $CC
004581 *
004582 CLOSE.PARMS   DFB      $1
004583 CLOSE.REF      DFB      $0
004584              REP      100
004585 * FIND.SEG (SRCHMODE.IN, PAGES.IN, SEGID.IN, BASE.OUT, LIMIT.OUT, SEGNUM.OUT)
004586              REP      100
004587 FINDSEG      EQU      $41
004588 *
004589 SEGMENT1       DFB      $6                ; FIND.SEG(SRCHMODE, SEGID, PAGECT, BASE, LIMIT, SEGNUM)
004590 SEGSRCH        DFB      $0,$3
004591 SEGPCNT        DW      $0000
004592              DW      $0
004593              DW      $0
004594              DFB      $0
004595              PAGE
004596              REP      100
004597 *
004598 * SOS SYSTEM CALLS (2)
004599 *
004600              REP      100
004601              REP      100
004602 * REQUEST.SEG (BASE.IN, LIMIT.IN, SEGID.IN, SEGNUM.OUT)
004603              REP      100
004604 REQSEG        EQU      $40
004605 *
004606 SEGMENT        DFB      $4                ; REQUEST SEG PARM LIST
004607 SEGBASE        DFB      $F,$0
004608 SEGLIM         DFB      $F,$1D
004609 SEGID          DFB      $0,$0
004610              REP      100
004611 * SET.PREFIX (PREFIXPATH.IN)
004612              REP      100
004613 SETPREFIX      EQU      $C6
004614 PREFIX.PARMS   DFB      $1
004615              DW      PREFIX.PATH
004616 PREFIX.PATH     DFB      $3
004617              ASC      '.D1'
004618              REP      100
004619 * GETTIME (TIME.OUT)
004620              REP      100
004621 GETTIME        EQU      $63
004622 *
004623 DTPARMS         DFB      1
004624              DW      DATETIME

```

```

004625 DATETIME      ASC      "YYYYMMDDWHHMMSSMMM"
004626              PAGE
004627              REP      100
004628 *
004629 * END OF SOSLDR CODE
004630 *
004631              REP      100
004632 SLOP          EQU      >$F8-*
004633              DS      SLOP          ; +-----+
004634 INITMODULE    DS      $200        ; ! KERNEL'S INIT MODULE RESIDES HERE !
004635 LDREND         EQU      *          ; +-----+
004636 FILE          EQU      *-$200+$400
004637              REP      100
004638 * SOS INTERPRETER FILE
004639              REP      100
004640 I.FILE         EQU      FILE
004641 I.HDR.CNT       EQU      I.FILE+$8
004642              REP      100
004643 * SOS DRIVER FILE
004644              REP      100
004645 D.FILE         EQU      FILE
004646 D.HDR.CNT       EQU      D.FILE+$8
004647 D.DRIVES       EQU      D.HDR.CNT+$2
004648 D.CHRSET       EQU      D.DRIVES+$2+$10
004649 D.KYBD         EQU      D.CHRSET+$10+$400
004650              REP      100
004651
004652              LST      ON
004653 ZZEND         EQU      *
004654 ZZLEN         EQU      ZZEND-ZZORG
004655 *
004656 NE            ZZLEN-LENLODR
004657              FAIL      2, "SOSORG      FILE IS INCORRECT FOR SOS LOADER"
004658              FIN
004659 *
004660
004661 *****
004662 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.F.SRC
004663 *****
004664
004665

```

```

004666 =====
004667 DOCUMENT :SOS1.3.1of5.ONE:SOS.SOSLDR.SRC.TEXT
004668 =====
004669
004670 *****
004671 * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.SRC
004672 *****
004673 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
004674
004675             SBTL             "SOS 1.1 SOS LOADER"
004676             REL
004677             ORG             $1E00
004678 ZZORG             EQU             *
004679             MSB             OFF
004680             REP             100
004681 *             COPYRIGHT (C) APPLE COMPUTER INC. 1980
004682 *             ALL RIGHTS RESERVED
004683             REP             100
004684 *
004685 *             SOS KERNEL LOAD & MEMORY POINTS
004686 *
004687 *  MODULE      START  END    I/O  ROM  SOS BLOAD  SIZE
004688 * -----
004689 *  SOSLDR      1E00 - 28F7                2000      0CF8
004690 *  INIT        28F8 - 2AA9                2AF8      [01B2]
004691 *  SYSGLOB     18FC - 1A03                2CF8
004692 *
004693 *  BFM.INIT2 + BITMAPS
004694 *             B800 - BBFF                2E00      03FF
004695 *  BFM         BC00 - DE62                3200      2263
004696 *  <PATCH>    DE63 - DE6A                5463      0008
004697 *
004698 *  OPRMSG      DE6B - E48A      X          546B      015A
004699 *  IPL         DFC5 - E48F      X      X    55C5      04CB
004700 *  UMGR        E490 - E89D      X      X    5A8B      040E
004701 *
004702 *  DISK3       E899 - EE03      X      X    5E99      056B
004703 *  SYSERR      EE04 - EED8      X          64D9      00D5
004704 *  DEVMGR      EED9 - F05D                64D9      0185
004705 *
004706 *  SCMGR       F05E - F2F3                665E      0296
004707 *  FMGR        F2F4 - F354                68F4      0061
004708 *  CFMGR       F355 - F551                6955      01FD
004709 *
004710 *  BUFMGR      F552 - F86D                6B52      031C
004711 *  MEMMGR      F86E - FFBE                6E6E      0751
004712 *  <END>       FFBE
004713 *
004714             REP             100

```

```

004715 * SOS LOADER (VERSION = 1.10 )
004716 * (DATE = 8/04/81)
004717 *
004718 * SOURCE FILES: SOSLDR.SRC, SOSLDR.A.SRC, SOSLDR.B.SRC, SOSLDR.C.SRC,
004719 * SOSLDR.D.SRC, SOSLDR.E.SRC, SOSLDR.F.SRC
004720 *
004721 * FUNCTION:
004722 * MOVES AND INITIALIZES SOS KERNEL, READS INTERPRETER FROM DISK, READS CHARACTER SET TABLE,
004723 * KEYBOARD TABLE AND DRIVERS FROM DISK, INITIALIZES ALL DRIVERS AND THEN JUMPS TO INTERPRETER
004724 * ENTRY POINT.
004725 *
004726 * CALLED BY:
004727 * SOSBOOT 7.0 WITH KERNEL FILE LOADED AT $I:1E00.9FFF(MAX)
004728 * WHERE: $I=INTERPRETER BANK (HIGHEST BANK IN SYSTEM)
004729 *
004730 * CALLS:
004731 * INTERPRETER ENTRY POINT (FIRST BYTE OF INTERPRETER CODE)
004732 *
004733 * DOCUMENTS:
004734 * SOS ERS APPENDICES - XX/XX/81
004735 * APPLE III I/O SYSTEM PROGRAMMERS GUIDE - DEC-15-80
004736 *
004737 * CONSTRAINTS:
004738 * INTERPRETER FILE: READ INTO BANK 0 BEGINNING AT $80:LDREND+$400(=BUFSIZE).
004739 * INTERPRETER CODE DOES NOT CONTAIN RELOCATION INFORMATION.
004740 * MAX = 38K ($I:2000..B7FF)
004741 * MIN = .25K ($I:B700..B7FF)
004742 *
004743 * DRIVER FILE: READ INTO BANK 0 BEGINNING AT $80:LDREND+$400(=BUFSIZE).
004744 * DRIVER MODULES ARE RELOCATED AND MOVED TO THE HIGHEST AVAILABLE 32K BANK USING
004745 * A "FIRST FIT" ALGORITHM. MODULES ARE REMOVED FROM THE FILE BEGINNING AT THE BACK
004746 * AND WORKING TOWARD THE FRONT. A DRIVER MODULE CANNOT SPAN A BANK BOUNDARY.
004747 *
004748 * DRIVER FILE: MAX = 60K (APPROX) DRIVER MODULE: MAX = 32K-1
004749 * MIN = .25K MIN < .25K
004750 *
004751 *
004752 * DATA STRUCTURES:
004753 * SOS.KERNEL FILE FORMAT
004754 * SOS.INTERP FILE FORMAT
004755 * SOS.DRIVER FILE FORMAT
004756 *
004757 * REP 100
004758 * PAGE
004759 * REP 100
004760 *
004761 * NOTATION:
004762 *
004763 * A, X, Y ::= 6502 REGISTERS
004764 *

```



```

004765 *   C, OV           ::= CARRY, OVERFLOW FLAGS IN 6502 STATUS (P) REGISTER
004766 *   E, Z, B       ::= ENVIRONMENT, ZERO PAGE, BANK REGISTERS (SYSTEM CONTROL REGISTERS)
004767 *
004768 *   (1.I.S.R:W.P.R.R) ::= ENVIRONMENT REGISTER FLAGS. FROM LEFT TO RIGHT BITS 7..0
004769 *                   (1MHZ, I/O ENABLE, SCREEN ENABLE, RESET ENABLE,
004770 *                   WRITE PROTECT, PRIMARY STACK, ROM1, ROM ENABLE)
004771 *
004772 *   "POSITIVE LOGIC" ::= ALL LOGIC USED IS POSITIVE LOGIC. FOR EXAMPLE, C="NO DRIVERS LEFT"
004773 *                   INDICATES THAT NO DRIVERS ARE LEFT WHEN CARRY = SET, AND THAT ONE OR
004774 *                   MORE DRIVERS ARE LEFT WHEN CARRY = CLEAR.
004775 *
004776 *   TRUE,FALSE      ::= TRUE = SET = ON, WHILE FALSE = CLEAR = OFF.
004777 *
004778 *                   REP          100
004779 *
004780 * ABBREVIATIONS:
004781 *
004782 *   DIB             ::= DEVICE INFORMATION BLOCK. DEFINES A UNIQUE DEVICE THAT CAN BE LINKED
004783 *                   INTO THE SYSTEM DEVICE TABLE. EACH DRIVER MODULE CONTAINS ONE OR MORE
004784 *                   DIBS (DEVICES) EACH OF WHICH CAN BE "ACTIVE" OR "INACTIVE".
004785 *
004786 *   ADIB           ::= "ACTIVE DIB"
004787 *
004788 *   <VARNAME>.P     ::= POINTER. A 3 BYTE ZERO PAGE POINTER. DON'T FORGET THE X BYTE!
004789 *
004790 *   SDT            ::= SYSTEM DEVICE TABLE. CONTAINS THE ENTRY POINT AND DIB ADDRESS OF EACH
004791 *                   DEVICE CONFIGURED INTO THE SYSTEM, (USED BY THE DEVICE MANAGER).
004792 *                   REP          100
004793 *
004794 *                   CHN          SOSLDR.A.SRC
004795 *
004796 * *****
004797 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSLDR.SRC
004798 * *****
004799 *
004800 *
004801 *

```

```

004802 =====
004803 DOCUMENT :SOS1.3.1of5.ONE:SOS.SYSGLOB.SRC.TEXT
004804 =====
004805
004806 *****
004807 * APPLE /// SOS 1.3 SOURCE CODE FILE: SYSGLOB.SRC
004808 *****
004809 * ASSEMBLER: APPLE |[ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
004810
004811         SBTL          "SOS 1.1  GLOBAL EQUATES"
004812         REL
004813         ORG          $18FC
004814         MSB          OFF
004815         REP          60
004816 *           COPYRIGHT (C) APPLE COMPUTER INC. 1980
004817 *           ALL RIGHTS RESERVED
004818         REP          60
004819 *
004820 *   SOS SYSTEM GLOBAL DATA & EQUATES
004821 *
004822 *   THIS MODULE CONTAINS THE SOS JUMP TABLE, AND ALL GLOBAL
004823 *   DATA AND EQUATES.  THE JUMP TABLE, AND ALL DATA THAT IS
004824 *   TO BE REFERENCED BY DEVICE HANDLERS, ARE ASSIGNED FIXED
004825 *   ADDRESSES AT THE BEGINNING OF MEMORY PAGE $19.  DATA
004826 *   THAT IS ONLY REFERENCED BY SOS BEGINS $1980, BUT MAY BE
004827 *   MOVED WHENEVER SOS IS RELINKED.
004828 *
004829         REP          60
004830 *
004831         EXTRN        ALLOCSIR
004832         EXTRN        DEALCSIR
004833         EXTRN        NMIDSBL
004834         EXTRN        NMIENBL
004835         EXTRN        QUEEVENT
004836         EXTRN        SELC800
004837         EXTRN        SYSDEATH
004838         EXTRN        SYSERR
004839         EXTRN        REQBUF
004840         EXTRN        GETBUFADR
004841         EXTRN        RELBUF
004842         EXTRN        NMIDBUG
004843         EXTRN        NMICONT
004844         EXTRN        COLDSTRT
004845 *
004846 *
004847         ENTRY        MEMSIZE
004848         ENTRY        SYSBANK
004849         ENTRY        SUSPFLSH
004850         ENTRY        NMIFLAG

```

004851	ENTRY	SCRNMODE
004852	ENTRY	GRSIZE
004853	*	
004854	ENTRY	SERR
004855	ENTRY	DEBUGBRK
004856	ENTRY	KYBDNMI
004857	ENTRY	NMISPSV
004858	ENTRY	SDEATH.REGS
004859	*	
004860	ENTRY	SOSVER
004861	ENTRY	SOSVERL
004862	*	
004863	ENTRY	SZPAGE
004864	ENTRY	SXPAGE
004865	ENTRY	SSPAGE
004866	*	
004867	ENTRY	CZPAGE
004868	ENTRY	CXPAGE
004869	ENTRY	CSPAGE
004870	ENTRY	CEVPRI
004871	*	
004872	ENTRY	SIRTEMP
004873	ENTRY	SIRARGSIZ
004874	ENTRY	IRQCNTN
004875	ENTRY	NMICNTR
004876	ENTRY	QEVTEMP
004877	ENTRY	QEV.THIS
004878	ENTRY	QEV.LAST
004879	*	
004880	ENTRY	BADBRK
004881	ENTRY	BADINT1
004882	ENTRY	BADINT2
004883	ENTRY	NMIHANG
004884	ENTRY	EVQOVFL
004885	ENTRY	STKOVFL
004886	ENTRY	BADSYSCALL
004887	ENTRY	DEV.OVFLOW
004888	ENTRY	MEM2SML
004889	ENTRY	VCBERR
004890	ENTRY	FCBERR
004891	ENTRY	ALCERR
004892	ENTRY	DIRERR
004893	ENTRY	TOOLONG
004894	ENTRY	BADBUFNUM
004895	ENTRY	BADBUFSIZ
004896	ENTRY	BITMAPADR
004897	*	
004898	ENTRY	BADSCNUM
004899	ENTRY	BADCZPAGE
004900	ENTRY	BADXBYTE

004901	ENTRY	BADSCPCNT
004902	ENTRY	BADSCBND
004903	*	
004904	ENTRY	NODNAME
004905	ENTRY	BADDDNUM
004906	*	
004907	ENTRY	BADPATH
004908	ENTRY	CFCBFULL
004909	ENTRY	FCCBFULL
004910	ENTRY	BADREFNUM
004911	ENTRY	PATHNOTFND
004912	ENTRY	VNFERR
004913	ENTRY	FNFERR
004914	ENTRY	DUPERR
004915	ENTRY	OVRERR
004916	ENTRY	DIRFULL
004917	ENTRY	CPTERR
004918	ENTRY	TYPERR
004919	ENTRY	EOFERR
004920	ENTRY	POSNERR
004921	ENTRY	ACCSEERR
004922	ENTRY	BTSERR
004923	ENTRY	FILBUSY
004924	ENTRY	NOTSOS
004925	ENTRY	BADLSTCNT
004926	ENTRY	OUTOFMEM
004927	ENTRY	BUFTBLFULL
004928	ENTRY	BADSYSBUF
004929	ENTRY	DUPVOL
004930	ENTRY	NOTBLKDEV
004931	ENTRY	LVLERR
004932	*	
004933	ENTRY	BADJMODE
004934	*	
004935	ENTRY	BADBKPG
004936	ENTRY	SEGRQDN
004937	ENTRY	SEGTBLFULL
004938	ENTRY	BADSEGNUM
004939	ENTRY	SEGNOTFND
004940	ENTRY	BADSRCHMODE
004941	ENTRY	BADCHGMODE
004942	ENTRY	BADPGCNT
004943	*	
004944	ENTRY	XREQCODE
004945	ENTRY	XCTLCODE
004946	ENTRY	XCTLPARM
004947	ENTRY	XNOTOPEN
004948	ENTRY	XNOTAVAIL
004949	ENTRY	XNORESRC
004950	ENTRY	XBADOP

```

004951          ENTRY      XIOERROR
004952          ENTRY      XNODRIVE
004953          ENTRY      XNOWRITE
004954          ENTRY      XBYTECNT
004955          ENTRY      XBLKNUM
004956          ENTRY      XDISKSW
004957          ENTRY      BACKMASK          ; MASK BYTE FOR BACKUP BIT.
004958          *
004959          ENTRY      E1908          ; DISK DRIVER IS READING/WRITING (SET) ELSE NOT (RESET)
004960          *
004961          PAGE
004962          DW          SYSGLOB          ;SYSGLOB TARGET ADDRESS
004963          DW          $0100          ; AND LENGTH
004964          *
004965          * SYSTEM GLOBAL DATA
004966          * (ACCESSIBLE TO SOS AND DEVICE HANDLERS)
004967          *
004968          SYSGLOB      EQU          *
004969          *
004970          MEMSIZE     DFB          $08          ;MEMORY SIZE = 128K
004971          SYSBANK     DFB          $02          ;SYSTEM BANK = 2
004972          SUSPFLSH    DFB          $00          ;SYSOUT SUSPEND/FLUSH FLAG
004973          NMIFLAG     DFB          $00          ;NMI PENDING FLAG
004974          DW          NMIEXIT          ;DEFAULT NMI VECTOR
004975          SCRNMODE     DFB          $80          ;CURRENT SCREEN MODE
004976          GRSIZE      DFB          $00
004977          *
004978          *
004979          * SOS JUMP TABLE
004980          *
004981          DS          SYSGLOB+$10-*, $00          ; USED BY THE MOUSE DRIVER
004982          USERNMI      JMP          NMIEXIT          ;KEYBOARD NMI VECTOR
004983          JMP          ALLOCSIR          ;ALLOCATE A SIR
004984          JMP          DEALCSIR          ;DEALLOCATE A SIR
004985          JMP          NMIDSBL          ;DISABLE NMI
004986          JMP          NMIEENBL          ;ENABLE NMI
004987          JMP          QUEEVENT          ;QUEUE AN EVENT
004988          JMP          SELC800          ;SELECT I/O EXPANSION ROM
004989          JMP          SYSDEATH          ;SYSTEM DEATH
004990          JMP          SYSERR          ;SOS ERROR
004991          JMP          REQBUF          ;REQUEST BUFFER
004992          JMP          GETBUFADR          ;GET BUFFER'S ADDRESS
004993          JMP          RELBUF          ;RELEASE BUFFER
004994          JMP          CLRBMASK          ;VECTOR TO CLRBMASK
004995          PAGE
004996          *
004997          * SOS DATA AND EQUATES
004998          * (ACCESSIBLE ONLY TO SOS)
004999          *
005000          DS          SYSGLOB+$80-*, $00

```

```

005001 SERR          DFB          $00          ;SYSTEM ERROR CODE
005002 *
005003 DEBUGBRK     NOP
005004              PLA          ;TO ENABLE DEBUG BREAK POINTS,
005005              PLA          ; PATCH THESE BYTES TO
005006              RTS          ; JMP TO THE DEBUGGER
005007 *
005008 KYBDNMI        JMP          USERNMI
005009              JMP          NMIDBUG
005010 NMISPSV        DFB          0
005011              JMP          NMICONT
005012 NMIEEXIT       RTS
005013 *
005014 *
005015 SOSVER         ASC          "SOS 1.3   01-DEC-82"
005016 SOSVERL        EQU          *-SOSVER
005017 *
005018              ASC          "(C) 1980, 1981 BY APPLE COMPUTER INC."
005019 *
005020 E1908            EQU          $1908          ; ALLOCATED TO STEPHEN SMITH (MOUSE DRIVER)
005021 * ABOVE SET AND RESET IN DISK DRIVER
005022 SZPAGE          EQU          $1800          ;SYSTEM ZERO PAGE
005023 SXPAGE          EQU          $1400          ;SYSTEM EXTEND PAGE
005024 SSPAGE          EQU          $0100          ;SYSTEM STACK PAGE
005025 *
005026 CZPAGE          EQU          $1A00          ;CALLER'S ZERO PAGE
005027 CXPAGE          EQU          $1600          ;CALLER'S EXTEND PAGE
005028 CSPAGE          EQU          $1B00          ;CALLER'S STACK PAGE
005029 CEVPRI          DFB          $00          ;CALLER'S EVENT PRIORITY
005030 *
005031 SIRTEMP          DFB          $00          ;TEMP FOR ALLOCSIR & DEALCSIR
005032 SIRARGSIZ        DFB          $00          ;ARGUMENT COUNT FOR ALLOCSIR & DEALCSIR
005033 IRQCNTR         DW           $0000          ;FALSE IRQ COUNTER
005034 NMICNTR         DW           $0000          ;COUNTER FOR NMILOCK
005035 QEVTEMP          DFB          $00          ;TEMP FOR QUEEVENT
005036 QEV.THIS         DFB          $00          ;POINTER FOR QUEEVENT
005037 QEV.LAST         DFB          $00          ;POINTER FOR QUEEVENT
005038 *
005039 SOSQUIT          DS           COLDSTRT
005040 BACKMASK         DFB          BACKBIT          ; MASK USED BY BFM TO UPDATE BACKUP BIT
005041 *
005042 * TO CLEAR THE BACKUP BIT, A PROGRAM MUST JSR TO CLRBMASK THRU 1934 THEN DO A
005043 * SET-FILE-INFO WITH NO INTERVENING SOS CALLS. ANY SOS CALL WILL
005044 * SET BACKMASK AGAIN. THIS FEATURE IS INTENTIONALLY LEFT UNDOCUMENTED.
005045 *
005046 CLRBMASK         AND          #BACKBIT          ; PURIFY
005047              STA          BACKMASK          ; SET THE MASK
005048              RTS          ; AND BACK TO THE CALLER
005049              PAGE
005050 *

```

```

005051 * SYSTEM DEATH REGISTER SAVE AREA
005052 * (SYSTEM STACK MOVED TO $1700-$17FF)
005053 *
005054          DS          SYSGLOB+$F6-*, $00
005055 SDEATH.REGS    EQU          *
005056          DFB          $00          ;BANK
005057          DFB          $00          ;ZERO PAGE
005058          DFB          $00          ;ENVIRONMENT
005059          DFB          $00          ;Y
005060          DFB          $00          ;X
005061          DFB          $00          ;A
005062          DFB          $00          ;STATUS
005063          DW           $00          ;PROGRAM COUNTER
005064          DFB          $00          ;STACK POINTER
005065 *
005066 * SYSTEM DEATH ERROR NUMBERS
005067 *
005068 BADBRK          EQU          $01          ;BRK FROM SOS
005069 BADINT1         EQU          $02          ;INTERRUPT NOT FOUND
005070 BADINT2         EQU          $03          ;BAD ZERO PAGE ALLOCATION
005071 NMIHANG         EQU          $04          ;UNABLE TO LOCK NMI
005072 EVQOVFL         EQU          $05          ;EVENT QUEUE OVERFLOW
005073 STKOVFL         EQU          $06          ;STACK OVERFLOW
005074 *
005075 BADSYSCALL        EQU          $07          ;DMGR DETECTED INVALID REQUEST CODE
005076 DEV.OVFLOW      EQU          $08          ;DMGR - TOO MANY DEVICE HANDLERS
005077 MEM2SML          EQU          $09          ;MEMORY SIZE < 64K
005078 VCBERR          EQU          $0A          ;VOLUME CONTROL BLOCK NOT USABLE (BFMGR)
005079 FCBERR          EQU          $0B          ;FILE CONTROL BLOCK CRASHED
005080 ALCERR          EQU          $0C          ;ALLOCATION BLOCKS INVALID
005081 TOOLONG         EQU          $0E          ;PATHNAME BUFFER OVERFLOW
005082 BADBUFNUM        EQU          $0F          ;INVALID BUFFER NUMBER
005083 BADBUFSIZ        EQU          $10          ;INVALID BUFFER SIZE (=0 OR >16K)
005084          PAGE
005085 *
005086 * SYSTEM ERROR NUMBERS
005087 *
005088 * - SYSTEM CALL MANAGER
005089 *
005090 BADSCNUM         EQU          $01          ;BAD SYSTEM CALL NUMBER
005091 BADCZPAGE        EQU          $02          ;BAD CALLER'S ZPAGE (MUST=$1A)
005092 BADXBYTE         EQU          $03          ;BITS 6..4 <> 0
005093 BADSCPCNT       EQU          $04          ;BAD SYSTEM CALL PARM COUNT
005094 BADSCBND5        EQU          $05          ;SYS CALL PARM ADR
005095 *
005096 * - DEVICE MANAGER
005097 *
005098 NODNAME          EQU          $10          ;DEVICE NAME NOT FOUND
005099 BADDNUM          EQU          $11          ;INVALID DEV.NUM PARM
005100 *

```

```

005101 * - DEVICE HANDLERS (STANDARD ERRORS)
005102 *
005103 XREQCODE      EQU      $20      ;INVALID REQUEST CODE
005104 XCTLCODE      EQU      $21      ;INVALID CONTROL/STATUS CODE
005105 XCTLPARM      EQU      $22      ;INVALID CONTROL/STATUS PARM
005106 XNOTOPEN      EQU      $23      ;DEVICE NOT OPEN
005107 XNOTAVAIL     EQU      $24      ;DEVICE NOT AVAILABLE
005108 XNORESRC      EQU      $25      ;UNABLE TO OBTAIN RESOURCE
005109 XBADOP        EQU      $26      ;INVALID OPERATION
005110 XIOERROR      EQU      $27      ;I/O ERROR
005111 *
005112 XNODRIVE       EQU      $28      ;NO DRIVE CONNECTED
005113 XNOWRITE      EQU      $2B      ;DEVICE WRITE PROTECTED
005114 XBYTECNT      EQU      $2C      ;BYTE COUNT <> A MULTIPLE OF 512
005115 XBLKNUM       EQU      $2D      ;BLOCK NUMBER TOO LARGE
005116 XDISKSW      EQU      $2E      ;DISK MEDIA HAS BEEN SWITCHED
005117 *
005118 * - NOTE: ERROR CODES $30-$3F HAVE BEEN RESERVED FOR DEVICE
005119 *   HANDLER SPECIFIC ERRORS
005120 *
005121 *
005122 * - FILE MANAGER
005123 *
005124 BADPATH        EQU      $40      ;PATHNAME, INVALID SYNTAX
005125 FCBCFULL       EQU      $41      ;CHAR FILE CTRL BLOCK TABLE FULL
005126 FCBFULL        EQU      $42      ;BLOCK FILE CTRL BLOCK TABLE FULL
005127 BADREFNUM      EQU      $43      ;INVALID REF.NUM PARM
005128 PATHNOTFND    EQU      $44      ;PATHNAME NOT FOUND
005129 VNFERR         EQU      $45      ;VOLUME NOT FOUND
005130 FNFERR         EQU      $46      ;FILE NOT FOUND
005131 DUPERR         EQU      $47      ;DUPLICATE FILE NAME ERROR
005132 OVRERR         EQU      $48      ;NOT ENOUGH DISK SPACE FOR PREALLOCATION
005133 DIRFULL        EQU      $49      ;DIRECTORY FULL ERROR
005134 CPTERR         EQU      $4A      ;FILE INCOMPATIBLE SOS VERSION
005135 TYPERR         EQU      $4B      ;NOT CURRENTLY SUPPORTED FILE TYPE
005136 EOFERR        EQU      $4C      ;POSITION ATTEMPTED BEYOND END OF FILE
005137 POSNERR       EQU      $4D      ;ILLEGAL POSITION (L.T. 0 OR G.T. $FFFFFF)
005138 ACCSERR        EQU      $4E      ;FILE ACCESS R/W REQUEST CONFLICTS WITH ATTRIBUTES
005139 BTSERR         EQU      $4F      ;USER SUPPLIED BUFFER TOO SMALL
005140 FILBUSY        EQU      $50      ;EITHER WRITE WAS REQUESTED OR WRITE ACCESS ALREADY ALLOCATED
005141 DIRERR         EQU      $51      ;DIRECTORY ERROR
005142 NOTSOS         EQU      $52      ;NOT A SOS DISKETTE
005143 BADLSTCNT      EQU      $53      ;INVALID VALUE IN LIST PARAMETER
005144 OUTOFMEM       EQU      $54      ;OUT OF FREE MEMORY FOR BUFFER
005145 BUFTBLFULL     EQU      $55      ;BUFFER TABLE FULL
005146 BADSYSBUF      EQU      $56      ;INVALID SYSBUF PARAMETER
005147 DUPVOL         EQU      $57      ;SON A BITCH GOT TWO VOLUMES OF SAME ROOT NAME!!!
005148 NOTBLKDEV     EQU      $58
005149 LVLERR         EQU      $59      ;INVALID FILE LEVEL
005150 BITMAPADR      EQU      $5A

```



```

005151 BACKBIT      EQU      $20      ; MASK FOR BACKUP BIT
005152 *
005153 * - UTILITY MANAGER
005154 *
005155 BADJMODE      EQU      $70      ; INVALID JOYSTICK REQUEST
005156 *
005157 * - MEMORY MANAGER
005158 *
005159 BADBKPG       EQU      $E0      ; INVALID BANK/PAGE PAIR
005160 SEGRQDN       EQU      $E1      ; SEGMENT REQUEST DENIED
005161 SEGTBLFULL    EQU      $E2      ; SEGMENT TABLE FULL
005162 BADSEGNUM     EQU      $E3      ; INVALID SEGMENT NUMBER
005163 SEGNOTFND    EQU      $E4      ; SEGMENT NOT FOUND
005164 BADSRCHMODE   EQU      $E5      ; INVALID SEARCH MODE PARM
005165 BADCHGMODE   EQU      $E6      ; INVALID CHANGE MODE PARM
005166 BADPGCNT     EQU      $E7      ; INVALID PAGE COUNT PARM
005167             ORG      SYSGLOB+$100
005168             DW      $B800      ; KERNEL TARGET ADDRESS
005169             DW      $47C0      ; AND LENGTH
005170
005171 *****
005172 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SYSGLOB.SRC
005173 *****
005174
005175
005176

```

```

005177 =====
005178 DOCUMENT :SOS1.3.2of5.TWO:SOS.BUGMGR.TEXT
005179 =====
005180
005181 *****
005182 * APPLE /// SOS 1.3 SOURCE CODE FILE: BUFMGR.SRC
005183 *****
005184 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
005185
005186             SBTL          "SOS 1.1  BUFFER MANAGER"
005187             REL
005188             INCLUDE      SOSORG,6,1,254
005189 *ORGBUFMG EQU $F552
005190 *LENBUFMG EQU $31C
005191             ORG           ORGBUFMG
005192 ZZORG       EQU         *
005193             MSB          OFF
005194             REP          60
005195 *           COPYRIGHT (C) APPLE COMPUTER INC. 1980
005196 *           ALL RIGHTS RESERVED
005197             REP          60
005198 *
005199 * BUFFER MANAGER (VERSION = 1.10  )
005200 *           (DATE      = 8/04/81)
005201 *
005202 * THIS MODULE IS RESPONSIBLE FOR CREATING AND RELEASING BUFFERS
005203 * FOR BOTH THE BLOCK FILE MANAGER AND, LATER, DEVICE HANDLERS
005204 * THE BUFFER MANAGER CREATES BUFFERS BY REQUESTING MEMORY
005205 * SEGMENTS FROM THE MEMORY MANAGER, AND RELEASES THEM VIA SAME.
005206 * THE PRIMARY DATA STRUCTURE IN THIS MODULE IS THE BUFFER TABLE.
005207 *
005208             REP          60
005209 *
005210             ENTRY       REQBUF
005211             ENTRY       REQFXBUF
005212             ENTRY       GETBUFADR
005213             ENTRY       CHKBUF
005214             ENTRY       RELBUF
005215 *
005216             EXTRN       MMGR
005217             EXTRN       SXPAGE
005218             EXTRN       CZPAGE
005219             EXTRN       CXPAGE
005220 *
005221             EXTRN       SYSERR
005222             EXTRN       SERR
005223             EXTRN       OUTFMEM
005224             EXTRN       BUFTBLFULL
005225             EXTRN       BADSYSBUF

```

```

005226 *
005227         EXTRN      SYSDEATH
005228         EXTRN      BADBUFNUM
005229         EXTRN      BADBUFSIZ
005230 *
005231         ENTRY      BUF.CNT
005232         ENTRY      PGCT.T
005233         ENTRY      XBYTE.T
005234         ENTRY      BUFREF
005235         PAGE
005236         REP        60
005237 *
005238 * DATA DECLARATIONS
005239 *
005240         REP        60
005241 *
005242 Z.REG         EQU      $FFD0
005243 *
005244 * MEMORY MGMT CALL PARM LOCATIONS ON SOS ZPAGE
005245 *
005246 M.TPARMX      EQU      $60                ; FIRST ADR OF MEM SYS CALL PARMS ON SOS ZPAGE
005247 REQCODE      EQU      M.TPARMX+$0
005248 *
005249 FINDSEG      EQU      $1
005250 SRCHMODE     EQU      M.TPARMX+$1
005251 F.ID         EQU      M.TPARMX+$2
005252 F.PGCT      EQU      M.TPARMX+$3
005253 F.PGCTX     DS        2                ; TEMP LOC FOR F.PGCT PARM
005254 F.BASE      EQU      M.TPARMX+$5
005255 F.BASEX     DS        2                ; TEMP LOC FOR F.BASE PARM
005256 F.LIM      EQU      M.TPARMX+$7
005257 F.LIMX     DS        2                ; TEMP LOC FOR F.LIM PARM
005258 F.NUM      EQU      M.TPARMX+$9
005259 F.NUMX     DS        1                ; TEMP LOC FOR F.NUM PARM
005260 *
005261 RELSEG      EQU      $5
005262 RLS.NUM     EQU      M.TPARMX+$1
005263 *
005264 * REQBUF DATA DECLARATIONS
005265 *
005266 RQB.PGCT    DS        1                ; REQUESTED PAGE COUNT
005267 RQB.BNUM    DS        1                ; BUFFER NUMBER (FM GETFREE CALL)
005268 *
005269 * REQFXBUF DATA DECLARATIONS
005270 *
005271 RQFB.PGCT  DS        1                ; REQUESTED PAGE COUNT
005272 RQFB.BNUM  DS        1                ; BUFFER NUMBER (FM GETFREE CALL)
005273 MAXPGCT    EQU      64                ; MAX BUFSIZE=16K
005274 F.TPARMX   EQU      $A0                ; FIRST ADR OF FILE SYS CALL PARMS ON SOS ZPAGE
005275 OPEN.LIST  EQU      F.TPARMX+$5        ; LOC OF OPEN.LIST PARM (OPEN SYS CALL)

```

```

005276 *
005277 * BUFCompact DATA DECLARATIONS (SOURCE ALSO USED BY CHKBUF)
005278 *
005279 BUFC.BNUM      DS          1          ; BUF# OF LOWEST BUFFER IN BUF.TBL
005280 SOURCE        EQU        M.TPARAMX+$10      ; & $11
005281 DEST         EQU        M.TPARAMX+$12      ; & $13
005282                PAGE
005283                REP          60
005284 *
005285 * BUFFER TABLE
005286 *
005287 * THE BUFFER TABLE CONSISTS OF "CNT"-1 ENTRIES (1 TO "CNT"-1).
005288 * EACH ENTRY IS "SIZ" BYTES IN LENGTH. THE "PGCT" FIELD
005289 * CONTAINS 3 SUBFIELDS. BIT 7 IS THE "FREE" FLAG (0=ACTIVE,1=FREE)
005290 * BIT 6 IS THE "FIXED" FLAG (0=FLOATING BUFFER,1=FIXED BUFFER)
005291 * BITS 5 THRU 0 CONTAIN THE PAGE COUNT OF AN "ACTIVE" ENTRY
005292 * (0=>1 PAGE,63=>64 PAGES DECIMAL). THE "XBYTE" FIELD CONTAINS
005293 * THE PROPER XBYTE OF AN "ACTIVE" ENTRY. THE "ADRH" FIELD
005294 * CONTAINS THE HIGH BYTE OF THE BUFFER ADDRESS. IF THE
005295 * BUFFER ENTRY IS "FLOATING", THEN THE "SEG" FIELD CONTAINS THE
005296 * SEGMENT NUMBER AND THE LOW BYTE OF THE BUFFER ADDRESS IS
005297 * ASSUMED TO BE ZERO.
005298 *
005299 * THUS, THE FOLLOWING RESTRICTIONS APPLY TO BUFFERS:
005300 *
005301 * (1) MAXIMUM BUFFER LENGTH IS 64 PAGES (16K)
005302 * (2) "FLOATING" BUFFERS ALWAYS BEGIN ON A PAGE BOUNDARY
005303 *      "FIXED" BUFFERS DO NOT.
005304 * (3) BUFFERS ARE ALWAYS AN INTEGRAL NUMBER OF PAGES IN LENGTH
005305 * (4) BUFFERS ALWAYS RESIDE IN THE 32K BANK MEMORY REGION,
005306 *      A LIMITATION OF FIND.SEG (MEMORY MANAGER)
005307 * (5) MAXIMUM NUMBER OF BUFFERS = 16; ENTRY 0 IS NOT USED.
005308 *
005309                REP          60
005310 *
005311 * BUFFER TABLE
005312 *
005313 BUF.SIZ         EQU        5
005314 BUF.CNT        EQU        17
005315 BUF.TBL        DS          BUF.SIZ*BUF.CNT
005316 PGCT.T        EQU        BUF.TBL
005317 XBYTE.T        EQU        PGCT.T+BUF.CNT
005318 ADRH.T         EQU        XBYTE.T+BUF.CNT
005319 SEG.T          EQU        ADRH.T+BUF.CNT
005320 ADRL.T         EQU        SEG.T
005321 CHK.T          EQU        ADRL.T+BUF.CNT
005322 ISFIXED        EQU        $40
005323 ISFREE         EQU        $80
005324 *
005325 * BUFFER REFERENCE TABLE

```

```

005326 *
005327 * FIRST BYTE IS COUNT, FOLLOWED BY "COUNT" BUFFER #S.
005328 * THIS TABLE IS A LIST OF ALL BUFFERS REFERENCED DURING ONE
005329 * SOS SYSTEM CALL. BUFFER #S ARE ADDED TO THIS LIST BY
005330 * GETBUFADR AND REMOVED BY CHKSUM.
005331 *
005332 BUFREF.CNT      EQU          17
005333 BUFREF          DS          BUFREF.CNT
005334 ZPAGEX        DS          1
005335                PAGE
005336                REP          60
005337 *
005338 * REQBUF
005339 *
005340 * INPUT:  PAGE.CNT (A)
005341 * OUTPUT: BUFNUM  (A)
005342 * ERROR:  "BUFFER TABLE FULL" - SYSERR
005343 *         "OUT OF MEMORY"      - SYSERR
005344 *         "BAD BUFFER SIZE"    - SYSDEATH
005345 *
005346 * THIS ROUTINE FINDS A FREE ENTRY IN THE BUFFER TABLE
005347 * AND THEN CALLS FIND.SEG (MMGR) TO OBTAIN MEMORY FOR IT.
005348 * IF MEMORY IS FOUND THEN THE BUFFER ENTRY IS MARKED "ACTIVE"
005349 * AND THE BUFFER INFO IS INSERTED INTO THE ENTRY
005350 *
005351                REP          60
005352 *
005353 REQBUF          EQU          *
005354 *
005355 * IF REQUESTED PGCT OUT OF BOUNDS THEN FATAL ERR
005356 *
005357                TAY
005358                BEQ          RQB.ERR2          ; FATAL ERR, INVALID BUFFER SIZE
005359                CPY          #MAXPGCT+1
005360                BCS          RQB.ERR2          ; FATAL ERR, INVALID BUFFER SIZE
005361                STY          RQB.PGCT         ; SAVE PAGE COUNT
005362 *
005363 * FIND FREE ENTRY IN BUF.TBL
005364 *
005365                JSR          GETFREE
005366                BCS          RQB.ERR          ; ERR, BUFFER TABLE FULL
005367                STX          RQB.BNUM
005368 *
005369 * FIND PGCT*256 BYTES OF FREE MEMORY
005370 *
005371                LDA          RQB.PGCT
005372                JSR          FSEG
005373                BCS          RQB.ERR1          ; ERR, OUT OF MEMORY
005374 *
005375 * INSERT PGCT, XBYTE, ADRH, SEG#, CHK BYTE IN BUF.TBL(BUF#)

```

```

005376 *
005377         LDX      RQB.BNUM
005378         DEC      RQB.PGCT          ; PAGE COUNT FIELD
005379         LDA      RQB.PGCT
005380         STA      PGCT.T,X
005381 *
005382         LDX      F.BASEX          ; XBYTE & ADRH FIELDS
005383         LDY      F.BASEX+1
005384         JSR      CNVRT.ADR
005385         CPX      #$8F
005386         BNE      RQB010
005387         LDX      #$7F          ; IF XBYTE=$8F THEN XBYTE:=$7F
005388 RQB010     TXA
005389         LDX      RQB.BNUM
005390         STA      XBYTE.T,X
005391         TYA
005392         STA      ADRH.T,X
005393 *
005394         LDA      F.NUMX          ; SEG# FIELD
005395         STA      SEG.T,X
005396 *
005397         LDA      #0          ; INIT CHECK BYTE TO NULL
005398         STA      CHK.T,X
005399 *
005400         TXA          ; RETURN BUF#
005401         CLC
005402         RTS          ; NORMAL EXIT
005403 *
005404 *
005405 RQB.ERR     LDA      #BUFTBLFULL
005406         JSR      SYSERR
005407 *
005408 RQB.ERR1    LDA      #OUTOFMEM
005409         JSR      SYSERR
005410 *
005411 RQB.ERR2    LDA      #BADBUFSIZ
005412         JSR      SYSDEATH
005413         PAGE
005414         REP      60
005415 *
005416 * REQFXBUF
005417 *
005418 * INPUT:  PAGE.CNT (A)
005419 * OUTPUT: BUFNUM  (A)
005420 * ERROR:  "BUFFER TABLE FULL"      - SYSERR
005421 *         "BAD SYSTEM.BUF PARM ADDRESS" - SYSERR
005422 *         "BAD BUFFER SIZE"         - SYSDEATH
005423 *
005424 * THIS ROUTINE COMPUTES THE ACTUAL BUFFER ADDRESS IN THE OPEN
005425 * CALL (PARM "OPEN.LIST"), AND ALLOCATES A BUFFER ENTRY FOR IT.

```

```

005426 * NOTE: THE SYSBUF PARAMETER MUST BE AN EXTENDED INDIRECT PTR!!
005427 *
005428             REP             60
005429 *
005430 REQFXBUF     EQU             *
005431 *
005432 * IF REQUESTED PGCT OUT OF BOUNDS THEN FATAL ERR
005433 *
005434             TAY
005435             BEQ             RQFB.ERR2             ; FATAL ERR, BAD BUFFER SIZE
005436             CPY             #MAXPGCT+1
005437             BCS             RQFB.ERR2             ; FATAL ERR, BAD BUFFER SIZE
005438 *
005439             STY             RQFB.PGCT             ; SAVE PAGE COUNT
005440 *
005441 * GET A FREE BUFFER ENTRY
005442 *
005443             JSR             GETFREE
005444             BCS             RQFB.ERR             ; ERR, BUFFER TABLE FULL
005445             STX             RQFB.BNUM             ; SAVE BUF#
005446 *
005447 * FETCH SYSTEM.BUF PARAMETER IN OPEN SYSTEM CALL
005448 *
005449             LDY             #3
005450             LDA             (OPEN.LIST),Y
005451             BNE             RQFB.ERR1             ; ERR, SYSBUF ADR
005452             DEY
005453             LDA             (OPEN.LIST),Y
005454             TAY
005455             LDA             CXPAGE+1,Y
005456             BPL             RQFB.ERR1             ; ERR, SYSBUF ADR
005457             CMP             #$8F
005458             BCS             RQFB.ERR1             ; ERR, SYSBUF ADR
005459 *
005460 * INSERT XBYTE, ADRH, ADRL, PGCT, CHK BYTE INTO BUF.TBL(BUF#)
005461 *
005462             LDX             RQFB.BNUM
005463             STA             XBYTE.T,X
005464 *
005465             LDA             CZPAGE+1,Y
005466             BEQ             RQFB.ERR1             ; ERR SYSBUF ADR
005467             CMP             #$81             ; CHECK FOR ADDRESS COMPENSATION
005468             BCC             RQFB010
005469             INC             XBYTE.T,X
005470             AND             #$7F
005471 RQFB010      STA             ADRH.T,X
005472 *
005473             LDA             CZPAGE,Y
005474             STA             ADRL.T,X
005475 *

```

```

005476          DEC          RQFB.PGCT
005477          LDA          RQFB.PGCT
005478          ORA          #ISFIXED
005479          STA          PGCT.T,X          ; BUFFER ENTRY NOW "ACTIVE"
005480 *
005481          LDA          #0          ; INIT CHECK BYTE TO NULL
005482          STA          CHK.T,X
005483 *
005484          TXA          ; RETURN BUF#
005485          CLC
005486          RTS          ; NORMAL EXIT
005487 *
005488 RQFB.ERR      LDA          #BUFTBLFULL
005489          JSR          SYSERR
005490 *
005491 RQFB.ERR1     LDA          #BADSYSBUF
005492          JSR          SYSERR
005493 *
005494 RQFB.ERR2     LDA          #BADBUFSIZ
005495          JSR          SYSDEATH
005496          PAGE
005497          REP          60
005498 *
005499 * GETBUFADR
005500 *
005501 * INPUT:  BUFNUM   (A)
005502 *        ZPAGELOC (X)
005503 * OUTPUT: BUF ADR AT: X,X+1 & SXPAGE+1,X
005504 *        PAGE.CNT (A)
005505 *        BUFNUM   (Y)
005506 *
005507 * ERROR:  "BADBUFNUM" SYSDEATH
005508 *
005509          REP          60
005510 *
005511 GETBUFADR     EQU          *
005512 *
005513 * IF BUF# OUT OF RANGE OR BUF.TBL(BUF#)=FREE
005514 * THEN FATAL ERR
005515 *
005516          TAY
005517          BEQ          GTBF.ERR          ; BUF#=0, FATAL ERR
005518          CPY          #BUF.CNT
005519          BCS          GTBF.ERR          ; BUF# > MAX BUF TABLE ENTRY, FATAL ERR
005520          LDA          PGCT.T,Y
005521          BMI          GTBF.ERR          ; BUF ENTRY MARKED "FREE", FATAL ERR
005522 *
005523 * OTHERWISE, CONSTRUCT BUFFER PTR ON SOS ZPAGE
005524 *
005525          JSR          GETBUFADR1

```



```

005526 *
005527 * IF BUFFER NOT PREVIOUSLY REFERENCED ON THIS SOS CALL AND CHECK BYTE <> 0
005528 *     THEN COMPARE FIRST BYTE OF BUFFER WITH CHECK BYTE IN BUFFER TABLE.
005529 *     IF NO MATCH THEN KILL SYSTEM.
005530 *
005531         STX         ZPAGEX
005532         TYA
005533         LDX         BUFREF
005534         BEQ         GTBF020             ; BUFREF EMPTY
005535 *
005536 GTBF010     CMP         BUFREF,X             ; SEARCH FOR PREVIOUS REFERENCE
005537         BEQ         GTBF030             ; MATCH FOUND
005538         DEX
005539         BNE         GTBF010
005540 *
005541 GTBF020     INC         BUFREF             ; LOG BUF # IN BUFREF TABLE
005542         LDX         BUFREF
005543         CPX         #BUFREF.CNT
005544         BCS         GTBF.ERR             ; BUFREF TABLE OVFLOW, KILL SYSTEM
005545         STA         BUFREF,X
005546 *
005547         LDA         CHK.T,Y
005548         BEQ         GTBF030             ; NO CHECK BYTE, SKIP CHECK
005549         LDX         ZPAGEX
005550         LDA         ($0,X)             ; COMPARE FIRST BYTE OF BUFFER
005551         CMP         CHK.T,Y             ; WITH CHECK BYTE IN BUF TABLE
005552         BNE         GTBF.ERR             ; NO MATCH, PULL THE PLUG
005553 *
005554 * RETURN PAGE.CNT TO CALLER
005555 *
005556 GTBF030     LDA         PGCT.T,Y
005557         AND         #$3F             ; STRIP OFF FREE, FIXED FLAGS
005558         CLC
005559         ADC         #1
005560 *
005561         CLC
005562         RTS
005563 *
005564 *
005565 GTBF.ERR     LDA         #BADBUFNUM
005566         JSR         SYSDEATH
005567 *
005568 *
005569         REP         60
005570 *
005571 * GETBUFADR1
005572 *
005573 * INPUT: PGCT.T(BUF#) (A)
005574 *         ZPAGELOC (X)
005575 *         BUF# (Y)

```

```

005576 * ERROR: NONE.
005577 *
005578 * EXTRACTS THE BUFFER POINTER FROM THE BUFFER TABLE AND
005579 * PLACES IT ON ZERO PAGE AT X, X+1 & SXPAGE+1,X
005580 *
005581         REP         60
005582 *
005583 GETBUFADR1     EQU         *
005584             AND         #$40
005585             BNE         GTB1010
005586             LDA         #0             ; "FIXED" BUFFER
005587             BEQ         GTB1020     ; ALWAYS TAKEN
005588 GTB1010       LDA         ADRL.T,Y   ; "FLOATING" BUFFER
005589 GTB1020       STA         0,X
005590             LDA         ADRH.T,Y
005591             STA         1,X
005592             LDA         XBYTE.T,Y
005593             ORA         #$80         ; ENSURE $7F->$8F
005594             STA         SXPAGE+1,X
005595             RTS
005596             PAGE
005597         REP         60
005598 *
005599 * CHKBUF
005600 *
005601 * CHECK BUFFER.  FETCHES THE FIRST BYTE OF EACH BUFFER
005602 * REFERENCED DURING THE CURRENT SYSTEM CALL AND PLACES IT
005603 * IN CHK.T(BUF#).
005604 *
005605 * INPUT:  BUFREF TABLE
005606 *        BUFFER TABLE
005607 * OUTPUT: EMPTY BUFREF TABLE
005608 *        BUFFER TABLE'S CHECK BYTES UPDATED
005609 *        Z REG:=$18
005610 * ERROR: NONE.
005611 *
005612         REP         60
005613 *
005614 CHKBUF         EQU         *
005615             LDY         BUFREF       ; PICK UP COUNT
005616             BEQ         CHKB.EXIT   ; EXIT IF BUFREF EMPTY
005617 *
005618             LDA         #$18         ; ENSURE SOS ZPAGE SWITCHED IN
005619             STA         Z.REG
005620 *
005621 * UPDATE THE CHECK BYTE OF EACH BUF# IN THE BUFREF TABLE
005622 *
005623 CHKB010       LDX         #>SOURCE
005624             LDA         BUFREF,Y
005625             TAY

```

```

005626          LDA      PGCT.T,Y
005627          JSR      GETBUFADR1          ; PUT BUF#S ADR ON ZPAGE
005628          LDA      ($0,X)
005629          STA      CHK.T,Y
005630          DEC      BUFREF
005631          LDY      BUFREF
005632          BNE      CHKB010          ; IF COUNT<>0 THEN PROCESS NEXT BUF# IN BUFREF TABLE
005633          *
005634          CHKB.EXIT          RTS          ; BUFREF TABLE IS EMPTY (COUNT=0)
005635          PAGE
005636          REP      60
005637          *
005638          * RELBUF
005639          *
005640          * INPUT:  BUFNUM   (A)
005641          * OUTPUT: NONE.
005642          * ERROR:  "BADBUFNUM" SYSDEATH
005643          *
005644          * THIS ROUTINE RELEASES THE BUFFER ENTRY, CALLS FIND.SEG TO
005645          * RELEASE THE CORRESPONDING MEMORY SEGMENT, AND CALLS
005646          * BUFCOMPACT TO PERFORM BUFFER COMPACTION.
005647          *
005648          REP      60
005649          *
005650          RELBUF      EQU      *
005651          *
005652          * IF BUF# OUT OF RANGE OR BUF.TBL(BUF#)=FREE
005653          * THEN FATAL ERR
005654          *
005655          TAY
005656          BEQ      RLBF.ERR
005657          CPY      #BUF.CNT
005658          BCS      RLBF.ERR
005659          LDA      PGCT.T,Y
005660          BMI      RLBF.ERR
005661          *
005662          * MARK BUF.TBL(BUF#)=FREE
005663          *
005664          ORA      #ISFREE
005665          STA      PGCT.T,Y
005666          *
005667          * IF BUF.TBL(BUF#)=FIXED THEN EXIT
005668          *
005669          AND      #ISFIXED
005670          BNE      RLBF.EXIT
005671          *
005672          * OTHERWISE CALL MEMORY MGR TO RELEASE BUFFER'S MEMORY SEG
005673          *
005674          LDA      #RELSEG
005675          STA      REQCODE

```

```

005676 *
005677         LDA     SEG.T,Y
005678         STA     RLS.NUM
005679 *
005680         JSR     MMGR
005681         BCS     RLBF.ERR           ; ANY ERR IS FATAL
005682 *
005683 * AND COMPACT BUFFERS
005684 *
005685         JSR     BUFCompact
005686 *
005687 RLBF.EXIT   CLC
005688         RTS
005689 *
005690 RLBF.ERR     LDA     #BADBUFNUM
005691         JSR     SYSDEATH
005692         PAGE
005693         REP     60
005694 *
005695 * BUFCompact
005696 *
005697 * THIS ROUTINE IS RESPONSIBLE FOR PACKING ALL SOS BUFFERS UP
005698 * AGAINST THE HIGHEST AVAILABLE FREE MEMORY.  COULD IMPROVE THE
005699 * EFFICIENCY OF THIS COMPACTION CYCLE BY NOT RELEASING THE "RELEASED" BUFFER
005700 * UNTIL IT IS KNOWN THAT ANOTHER BUFFER WILL NOT BE MOVED INTO ITS LOC.
005701 *
005702         REP     60
005703 *
005704 BUFCompact   EQU     *
005705 *
005706 * FIND THE FLOATING BUFFER IN BUF.TBL WITH THE LOWEST ADDRESS.
005707 *
005708 BUF010      LDY     #0
005709         LDX     #BUF.CNT-1
005710 *
005711 BUF020      LDA     PGCT.T,X
005712         AND     #$C0           ; STRIP OUT PAGE COUNT BITS
005713         BNE     BUF030
005714 *
005715         LDA     ADRH.T,X
005716         CMP     ADRH.T,Y
005717         LDA     XBYTE.T,X
005718         SBC     XBYTE.T,Y
005719         BCS     BUF030
005720 *
005721         TXA           ; SMALLER BUFFER FOUND, SAVE IN Y
005722         TAY
005723 *
005724 BUF030      DEX
005725         BNE     BUF020

```

```

005726 *
005727 * IF NO BUFFER FOUND THEN DONE
005728 *
005729         TYA
005730         BNE         BUFC040
005731         JMP         BUFC.EXIT
005732 BUFC040     STY         BUFC.BNUM           ; OTHERWISE SAVE BUF# IN Y REG.
005733 *
005734 * CALL FIND.SEG:  FINDS HIGHEST AVAILABLE FREE MEMORY
005735 *
005736         LDA         PGCT.T,Y
005737         AND         #$3F                   ; STRIP OUT "FREE","FIXED" FLAGS
005738         CLC
005739         ADC         #1
005740         JSR         FSEG
005741         BCS         BUFC.EXIT           ; DONE IF NO FREE SEG FOUND
005742 *
005743 * CONVERT BASE.BKPG TO BUFFER ADR
005744 *
005745         LDX         F.BASEX               ; BASE BANK
005746         LDY         F.BASEX+1           ; BASE PAGE
005747         JSR         CNVRT.ADR
005748         STX         F.BASEX             ; XBYTE
005749         STY         F.BASEX+1         ; ADRH
005750 *
005751 * IF NEW SEG'S BASE < CURRENT BUFFER'S BASE ADR THEN DONE
005752 *
005753         LDY         BUFC.BNUM
005754         LDA         ADRH.T,Y
005755         STA         SOURCE+1
005756         CMP         F.BASEX+1
005757         LDA         XBYTE.T,Y
005758         STA         SXPAGE+SOURCE+1
005759         SBC         F.BASEX
005760         BCS         BUFC.EXIT1
005761 *
005762 * MOVE DATA FROM CURRENT BUFFER TO NEW BUFFER
005763 *
005764         LDX         F.BASEX
005765         STX         SXPAGE+DEST+1
005766         LDY         F.BASEX+1
005767         STY         DEST+1
005768         LDA         #0
005769         STA         SOURCE
005770         STA         DEST
005771 *
005772         TAY
005773         LDX         F.PGCTX
005774 BUFC200     LDA         (SOURCE),Y       ; MOVE LOOP
005775         STA         (DEST),Y

```

```

005776          DEY
005777          BNE          BUFC200
005778          INC          SOURCE+1
005779          INC          DEST+1
005780          DEX
005781          BNE          BUFC200
005782          *
005783          * UPDATE BUF.TBL(BUF#)
005784          *
005785          LDY          BUFC.BNUM
005786          LDA          F.BASEX
005787          STA          XBYTE.T,Y
005788          LDA          F.BASEX+1
005789          STA          ADRH.T,Y
005790          *
005791          LDX          SEG.T,Y
005792          LDA          F.NUMX
005793          STA          SEG.T,Y
005794          *
005795          * AND RELEASE OLD MEMORY SEGMENT
005796          *
005797          STX          RLS.NUM
005798          LDA          #RELSEG
005799          STA          REQCODE
005800          JSR          MMGR
005801          BCS          BUFC.ERR
005802          *
005803          JMP          BUFC010          ; REPEAT COMPACTION CYCLE
005804          *
005805          *
005806          BUFC.EXIT1  LDX          F.NUMX          ; DONE,
005807          STX          RLS.NUM          ; RELEASE SEG BEFORE EXIT
005808          LDA          #RELSEG
005809          STA          REQCODE
005810          JSR          MMGR
005811          BCS          BUFC.ERR
005812          *
005813          BUFC.EXIT  LDA          #0
005814          STA          SERR          ; MASK OUT ANY ERROR FROM MEMORY MGR
005815          CLC
005816          RTS          ; NORMAL EXIT
005817          *
005818          *
005819          BUFC.ERR  LDA          #BADBUFNUM
005820          JSR          SYSDEATH
005821          PAGE
005822          REP          60
005823          *
005824          * FSEG
005825          *

```

```

005826 * INPUT: PAGE.CNT (A)
005827 * OUTPUT: PAGE.CNT (A) UNCHANGED IF FIND.SEG SUCCESSFUL
005828 * ERROR: CARRY SET "UNABLE TO FIND MEMORY SEG OF PAGE.CNT*256 BYTES"
005829 *
005830 * THIS ROUTINE BUILDS THE PARAMETERS FOR A FIND.SEG SYSTEM CALL
005831 * AND THEN CALLS THE MEMORY MANAGER.
005832 *
005833             REP             60
005834 *
005835 FSEG             EQU             *
005836 *
005837 * SETUP INPUT PARAMETERS FOR FIND.SEG CALL
005838 *
005839             STA             F.PGCTX
005840             LDA             #FINDSEG
005841             STA             REQCODE
005842             LDA             #2
005843             STA             SRCHMODE
005844             LDA             #4
005845             STA             F.ID
005846 *
005847 * SETUP OUTPUT PARAMETER ADRESSES
005848 *
005849             LDA             #>F.PGCTX
005850             STA             F.PGCT
005851             LDA             #<F.PGCTX
005852             STA             F.PGCT+1
005853             LDA             #>F.BASEX
005854             STA             F.BASE
005855             LDA             #<F.BASEX
005856             STA             F.BASE+1
005857             LDA             #>F.LIMX
005858             STA             F.LIM
005859             LDA             #<F.LIMX
005860             STA             F.LIM+1
005861             LDA             #>F.NUMX
005862             STA             F.NUM
005863             LDA             #<F.NUMX
005864             STA             F.NUM+1
005865 *
005866             LDA             #0
005867             STA             F.PGCTX+1
005868             STA             SXPAGE+F.PGCT+1
005869             STA             SXPAGE+F.BASE+1
005870             STA             SXPAGE+F.LIM+1
005871             STA             SXPAGE+F.NUM+1
005872 *
005873             JSR             MMGR
005874             LDA             F.PGCTX
005875 *

```

```

005876             RTS             ; EXIT.  CARRY SET->ERR
005877             PAGE
005878             REP             60
005879 *
005880 * GETFREE
005881 *
005882 * INPUT:  NONE
005883 * OUTPUT: BUF# (X)
005884 * ERROR:  "BUFTBLFULL" SYSERR
005885 *
005886 * THIS ROUTINE SEARCHES THE BUFFER TABLE, LOOKING FOR A FREE
005887 * ENTRY.  IF FOUND, IT RETURNS THE BUFFER NUMBER, ELSE ERROR.
005888 *
005889             REP             60
005890 *
005891 GETFREE             EQU             *
005892             LDX             #BUF.CNT-1
005893 GFR010             LDA             PGCT.T,X
005894             BMI             GFR.EXIT             ; FREE ENTRY FOUND
005895             DEX
005896             BNE             GFR010
005897 *
005898             LDA             #BUFTBLFULL
005899             JSR             SYSERR             ; ERR EXIT
005900 *
005901 GFR.EXIT           CLC
005902             RTS             ; NORMAL EXIT
005903             PAGE
005904             REP             60
005905 *
005906 * CNVRT.ADR
005907 *
005908 * INPUT:  BANK VALUE (X)
005909 *         PAGE VALUE (Y)
005910 * OUTPUT: XBYTE (X)
005911 *         ADRH (Y)
005912 * ERROR:  NONE.
005913 *
005914 * THIS ROUTINE CONVERTS A BASE.BKPG PARM (MMGR) INTO A
005915 * VIRTUAL POINTER
005916 *
005917             REP             60
005918 *
005919 CNVRT.ADR           EQU             *
005920 *
005921 * IF PAGE <> $20 THEN GOTO L2
005922 *
005923             CPY             #$20
005924             BNE             CNVA020
005925 *

```



```

005926 * IF BANK <> 0 THEN GOTO L1
005927 *
005928             TXA
005929             BNE          CNVA010
005930 *
005931 * XBYTE=$8F
005932 * ADRH:=PAGE
005933 *
005934             LDX          #$8F
005935             BMI          CNVA.EXIT
005936 *
005937 * L1: XBYTE:=(BANK-1) ORA #$80
005938 *       ADRH:=#$80
005939 *
005940 CNVA010     ORA          #$80
005941             TAX
005942             DEX
005943             LDY          #$80
005944             BMI          CNVA.EXIT
005945 *
005946 * L2: XBYTE:=BANK ORA #$80
005947 *       ADRH:=ADRH-#$20
005948 *
005949 CNVA020     TXA
005950             ORA          #$80
005951             TAX
005952             SEC
005953             TYA
005954             SBC          #$20
005955             TAY
005956 *
005957 CNVA.EXIT   RTS
005958 *
005959             LST          ON
005960 ZZEND       EQU          *
005961 ZZLEN       EQU          ZZEND-ZZORG
005962             IFNE        ZZLEN-LENBUFMG
005963             FAIL       2,"SOSORG          FILE IS INCORRECT FOR BUFMGR"
005964             FIN
005965
005966 *****
005967 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: BUFMGR.SRC
005968 *****
005969
005970

```

```

005971 =====
005972 DOCUMENT :SOS1.3.2of5.TWO:SOS.CRMGR.TEXT
005973 =====
005974
005975 *****
005976 * APPLE /// SOS 1.3 SOURCE CODE FILE: CFMGR.SRC
005977 *****
005978 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
005979
005980          SBTL          "SOS 1.1 CHARACTER FILE MANAGER"
005981          REL
005982          INCLUDE      SOSORG,6,1,254
005983          ORG          ORGCFM
005984 ZZORG          EQU          *
005985          MSB          OFF
005986          REP          60
005987 *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
005988 *          ALL RIGHTS RESERVED
005989          REP          60
005990 *
005991 * CHARACTER FILE MANAGER (VERSION = 1.10 )
005992 *          (DATE      = 8/04/81)
005993 *
005994 * THIS MODULE TRANSFORMS CHARACTER FILE SYSTEM CALLS INTO
005995 * DEVICE CALLS TO THE APPROPRIATE DEVICE HANDLER. ONLY
005996 * OPEN, NEWLINE, READ, WRITE AND CLOSE CALLS ARE PERMITTED
005997 * ON CHARACTER FILES.
005998 *
005999          REP          60
006000 *
006001          ENTRY        CFMGR
006002 *
006003          ENTRY        CFCB.MAX
006004          ENTRY        CFCB.DEV
006005 *
006006          EXTRN        DMGR
006007          EXTRN        LEVEL
006008          EXTRN        MAX.DNUM
006009          EXTRN        SXPAGE
006010 *
006011          EXTRN        SYSERR
006012          EXTRN        SERR
006013          EXTRN        BADSCNUM
006014          EXTRN        CFCBFULL
006015          EXTRN        BADREFNUM
006016          EXTRN        FNFERR
006017          PAGE
006018          REP          60
006019 *

```

```

006020 * DATA DECLARATIONS
006021 *
006022             REP             60
006023 *
006024 * FILE CALL PARM LOCATIONS ON SOS ZPAGE
006025 *
006026 F.TPARMX      EQU           $A0
006027 REQCODE       EQU           F.TPARMX
006028 O.PATH         EQU           F.TPARMX+1      ; OPEN'S PATHNAME LOC
006029 O.REFNUM      EQU           F.TPARMX+3      ; OPEN'S REFNUM LOC
006030 REFNUM        EQU           F.TPARMX+1      ; REFNUM'S LOC IN OTHER CALLS
006031 NL.ISNL      EQU           F.TPARMX+2      ; NEWLINE'S ISNEWLINE LOC
006032 NL.NLCHR     EQU           F.TPARMX+3      ; NEWLINE'S NEWLINECHAR LOC
006033 RW.BUF       EQU           F.TPARMX+2      ; READ/WRITE'S BUF LOC
006034 RW.BYTES    EQU           F.TPARMX+4      ; READ/WRITE'S BYTES LOC
006035 RD.BYTESRD   EQU           F.TPARMX+6      ; READ'S BYTESREAD LOC
006036 *
006037 * FILE REQUEST CODE VALUES
006038 *
006039 OPEN          EQU           8
006040 NEWLINE      EQU           9
006041 READ         EQU           $A
006042 WRITE        EQU           $B
006043 CLOSE       EQU           $C
006044             PAGE
006045 * DEVICE CALL PARM LOCATIONS ON SOS ZPAGE
006046 *
006047 D.TPARMX      EQU           $C0
006048 D.SCNUM       EQU           D.TPARMX          ; DEVICE SYS CALL # LOC
006049 GDN.DNAME     EQU           D.TPARMX+1      ; GETDEVNUM DNAME LOC
006050 GDN.DNUM      EQU           D.TPARMX+3      ; GETDEVNUM DNUM LOC
006051 D.DNUM        EQU           D.TPARMX+1      ; OPN/CLOSE/RD/WR/CTRL'S DNUM LOC
006052 DRW.BUF      EQU           D.TPARMX+2      ; RD/WR'S BUF LOC
006053 DRW.BYTES    EQU           D.TPARMX+4      ; RD/WR'S BYTES LOC
006054 DRD.BYTESRD  EQU           D.TPARMX+8      ; RD/WR'S BYTESREAD LOC
006055 DC.CCODE     EQU           D.TPARMX+2      ; DCTRL'S CTRLCODE LOC
006056 DC.CLIST    EQU           D.TPARMX+3      ; DCTRL'S CTRLLIST LOC
006057 *
006058 * DEVICE REQUEST CODE VALUES
006059 *
006060 DREAD         EQU           $0
006061 DWRITE       EQU           $1
006062 DCTRL        EQU           $3
006063 GETDEVNUM    EQU           $4
006064 DOPEN        EQU           $6
006065 DCLOSE      EQU           $7
006066 *
006067 CTRL.LIST     DS             2              ; CONTAINER FOR NEWLINE DCTRL CALL
006068 NEWLINECC    EQU           2              ; NEWLINE CTRL CODE
006069 *

```

```

006070 * GETDNUM VARS
006071 *
006072 DNUM.TEMP      DS      1
006073 *
006074 * CLOSEALL VARS
006075 *
006076 DCLOSE.ERR      EQU      F.TPARMX+$F
006077 DCLOSE.TBL      EQU      $200
006078 TRUE           EQU      $80
006079 FALSE        EQU      $0
006080 *
006081 *
006082             REP      60
006083 *
006084 * CHARACTER FILE CONTROL BLOCK TABLE
006085 * (ENTRY 0 IS NOT USED)
006086 *
006087             REP      60
006088 CFCB.MAX         EQU      17
006089 CFCB.DEV        DS      CFCB.MAX
006090 CFCB.LVL        DS      CFCB.MAX
006091             PAGE
006092             REP      60
006093 *
006094 * CHARACTER FILE MANAGER - MAIN ENTRY POINT
006095 *
006096             REP      60
006097 CFMGR           EQU      *
006098 *
006099 * SWITCH, BASED ON REQUEST CODE
006100 *
006101             LDA      REQCODE
006102             CMP      #OPEN
006103             BEQ      CFOPEN           ; "OPEN"
006104             CMP      #NEWLINE
006105             BEQ      CFNEWLINE       ; "NEWLINE"
006106             CMP      #READ
006107             BEQ      CFREAD          ; "READ"
006108             CMP      #WRITE
006109             BNE      CFM010
006110             JMP      CFWRITE         ; "WRITE"
006111 CFM010          CMP      #CLOSE
006112             BNE      CFM020
006113             JMP      CFCLOSE        ; "CLOSE"
006114 CFM020          LDA      #BADSCNUM
006115             JSR      SYSERR          ; ERR EXIT
006116             PAGE
006117             REP      60
006118 * OPEN(IN.PATHNAME; OUT.REFNUM; IN.OPENLIST,LENGTH) SYSTEM CALL
006119             REP      60

```

```

006120 CFOPEN      EQU      *          ; BUILD "D.OPEN" CALL
006121           JSR      GETDNUM      ; MAP PATH TO DEV#
006122           BCS      CFOP.ERR1    ; ERR - FILE NOT FOUND
006123           STA      D.DNUM
006124 *
006125           JSR      REQ.CFCB      ; BUILD NEW CFCB ENTRY
006126           BCS      CFOP.ERR1    ; ERR - CFCB FULL
006127           LDX      #0
006128           STA      (O.REFNUM,X)   ; RETURN REFNUM TO CALLER
006129           CPY      #1
006130           BNE      CFOP.EXIT     ; DEVICE ALREADY OPEN
006131 *
006132           LDA      #DOPEN
006133           STA      D.SCNM
006134           JSR      DMGR          ; DOPEN CALL
006135           BCS      CFOP.ERR
006136 CFOP.EXIT    RTS
006137 *
006138 CFOP.ERR      LDA      SERR        ;KLUDGE - 1.0 DRIVERS DON'T SUPPORT CARRY ERR PROTOCOL
006139           BEQ      CFOP.EXIT     ;NO ERROR
006140           LDX      #0             ; RELEASE CFCB ENTRY
006141           LDA      (O.REFNUM,X)
006142           JSR      REL.CFCB
006143 CFOP.ERR1    RTS                ; ERR EXIT
006144           PAGE
006145           REP      60
006146 * NEWLINE ( IN.REFNUM, IS      .NEWLINE,NEWLINE.CHAR) SYSTEM CALL
006147           REP      60
006148 CFNEWLINE    EQU      *          ; BUILD "D.CONTROL" CALL
006149           LDA      #DCTRL
006150           STA      D.SCNM
006151           LDA      REFNUM
006152           JSR      GET.CFCB      ; MAP REFNUM TO DEV #
006153           BCS      CFNL.ERR     ; ERR - BAD REFNUM
006154 *
006155           STA      D.DNUM
006156           LDA      #NEWLINECC
006157           STA      DC.CCODE
006158 *
006159           LDA      #>CTRL.LIST
006160           STA      DC.CLIST
006161           LDA      #<CTRL.LIST
006162           STA      DC.CLIST+1
006163           LDA      #0
006164           STA      SXPAGE+DC.CLIST+1
006165 *
006166           LDA      NL.ISNL
006167           STA      CTRL.LIST
006168           LDA      NL.NLCHR
006169           STA      CTRL.LIST+1

```

```

006170 *
006171         JSR         DMGR             ; DCONTROL CALL
006172         RTS             ; NORMAL EXIT
006173 *
006174 CFNL.ERR     RTS             ; ERR EXIT
006175         PAGE
006176         REP         60
006177 * READ(IN.REFNUM,BUF,BYTES,BYTESREAD) SYSTEM CALL
006178         REP         60
006179 CFREAD        EQU         *             ; BUILD "D.READ" CALL
006180         LDA         #DREAD
006181         STA         D.SCNUM
006182         LDA         REFNUM
006183         JSR         GET.CFCB          ; MAP REFNUM TO DEV #
006184         BCS         CFRD.ERR          ; ERR - BAD REFNUM
006185 *
006186         STA         D.DNUM
006187         LDX         #3
006188 CFRD010       LDA         RW.BUF,X
006189         STA         DRW.BUF,X
006190         DEX
006191         BPL         CFRD010
006192 *
006193         LDA         RD.BYTESRD
006194         STA         DRD.BYTESRD
006195         LDA         RD.BYTESRD+1
006196         STA         DRD.BYTESRD+1
006197 *
006198         LDA         SXPAGE+RW.BUF+1
006199         STA         SXPAGE+DRW.BUF+1
006200         LDA         SXPAGE+RW.BYTES+1
006201         STA         SXPAGE+DRW.BYTES+1
006202         LDA         SXPAGE+RD.BYTESRD+1
006203         STA         SXPAGE+DRD.BYTESRD+1
006204 *
006205         JSR         DMGR             ; DREAD CALL
006206         RTS             ; NORMAL EXIT
006207 *
006208 CFRD.ERR     RTS             ; ERR EXIT
006209         PAGE
006210         REP         60
006211 * WRITE(IN.REFNUM,BUF,BYTES) SYSTEM CALL
006212         REP         60
006213 CFWRITE       EQU         *             ; BUILD "D.WRITE" CALL
006214         LDA         #DWRITE
006215         STA         D.SCNUM
006216         LDA         REFNUM
006217         JSR         GET.CFCB          ; MAP REFNUM TO DEV #
006218         BCS         CFWR.ERR          ; ERR - BAD REFNUM
006219         STA         D.DNUM

```

```

006220          LDX      #3
006221 CFWR010    LDA      RW.BUF,X
006222          STA      DRW.BUF,X
006223          DEX
006224          BPL      CFWR010
006225          LDA      SXPAGE+RW.BUF+1
006226          STA      SXPAGE+DRW.BUF+1
006227          LDA      SXPAGE+RW.BYTES+1
006228          STA      SXPAGE+DRW.BYTES+1
006229 *
006230          JSR      DMGR              ; DWRITE CALL
006231          RTS              ; NORMAL EXIT
006232 *
006233 CFWR.ERR   RTS              ; ERR EXIT
006234          PAGE
006235          REP      60
006236 * CLOSE(IN.REFNUM) SYSTEM CALL
006237          REP      60
006238 CFCLOSE     EQU      *              ; BUILD "D.CLOSE" CALL
006239          LDA      #DCLOSE
006240          STA      D.SCNUM
006241          LDA      REFNUM
006242          BEQ      CLOSEALL
006243 *
006244          JSR      REL.CFCB          ; RELEASE CFCB ENTRY
006245          BCS      CFCL010
006246          STA      D.DNUM
006247          TYA
006248          BNE      CFCL010
006249          JSR      DMGR              ; DCLOSE CALL
006250 CFCL010     RTS              ; NORMAL EXIT
006251 *
006252          PAGE
006253          REP      60
006254 *
006255 * CLOSE ALL CHARACTER FILES W/LEVELS >= TO CURRENT SYSTEM FILE LEVEL.
006256 *
006257          REP      60
006258 *
006259 CLOSEALL     EQU      *              ; SET ENTRIES IN DEV CLOSE TBL TO FALSE
006260          LDA      #FALSE
006261          LDX      MAX.DNUM
006262 CFCL020     STA      DCLOSE.TBL,X
006263          DEX
006264          BPL      CFCL020
006265 *
006266          LDX      #CFCB.MAX-1      ; CLOSE ALL DEVICES >= TO CURRENT LEVEL
006267 CFCL030     LDA      CFCB.DEV,X    ; AND MARK TRUE IN DEV CLOSE TBL
006268          TAY
006269          BMI      CFCL050

```

```

006270          LDA      CFCB.LVL,X
006271          CMP      LEVEL
006272          BCC      CFCL050
006273          LDA      #TRUE
006274          STA      DCLOSE.TBL,Y
006275          SEC
006276          ROR      CFCB.DEV,X
006277 CFCL050     DEX
006278          BNE      CFCL030
006279          *
006280          LDX      #CFCB.MAX-1          ; DON'T CLOSE DEVICES < CURRENT LEVEL
006281 CFCL060     LDA      CFCB.DEV,X
006282          TAY
006283          BMI      CFCL070
006284          LDA      #FALSE
006285          STA      DCLOSE.TBL,Y
006286 CFCL070     DEX
006287          BNE      CFCL060
006288          *
006289          LDA      #0
006290          STA      DCLOSE.ERR
006291          LDX      MAX.DNUM          ; ISSUE D'CLOSE CALLS TO ALL DEVICES MARKED AS TRUE
006292 CFCL080     LDA      DCLOSE.TBL,X          ; IN DEV CLOSE TABLE
006293          BPL      CFCL090
006294          TXA
006295          PHA
006296          STX      D.DNUM
006297          JSR      DMGR
006298          PLA
006299          TAX
006300          LDA      SERR
006301          BEQ      CFCL090          ; IF ERROR,
006302          STA      DCLOSE.ERR          ; THEN SAVE IT
006303 CFCL090     DEX
006304          BNE      CFCL080
006305          *
006306          LDA      DCLOSE.ERR          ; IF $0 THEN NO ERRORS FROM D.CLOSE CALLS
006307          BNE      CFCL.ERR
006308          RTS          ; NORMAL EXIT
006309 CFCL.ERR     JSR      SYSERR          ; RETURN LAST D.CLOSE ERROR REPORTED
006310          PAGE
006311          REP      60
006312          *
006313          * GET DEVICE NUMBER
006314          *
006315          * INPUT:  CPATH
006316          * OUTPUT: DEVICE NUMBER (A)
006317          * ERROR:  CARRY SET ("FILE NOT FOUND")
006318          *
006319          * GETDNUM FIRST CALLS THE DMGR (GETDEVNUM) MAP THE PATHNAME

```



```

006320 * TO A DEVICE #.  GETDNUM THEN ENSURES THAT THE PATHNAME
006321 * IS NOT A BLOCK DEVICE BY CHECKING THE DBLKLST TABLE.
006322 *
006323         REP         60
006324 *
006325 GETDNUM         EQU         *
006326         LDA         #GETDEVNUM
006327         STA         D.SCNUM
006328 *
006329         LDA         O.PATH
006330         STA         GDN.DNAME
006331         LDA         O.PATH+1
006332         STA         GDN.DNAME+1
006333 *
006334         LDA         #>DNUM.TEMP
006335         STA         GDN.DNUM
006336         LDA         #<DNUM.TEMP
006337         STA         GDN.DNUM+1
006338 *
006339         LDA         SXPAGE+O.PATH+1
006340         STA         SXPAGE+GDN.DNAME+1
006341         LDA         #0
006342         STA         SXPAGE+GDN.DNUM+1
006343 *
006344         JSR         DMGR
006345         BCS         GETD.ERR             ; D.NAME NOT FOUND
006346         BMI         GETD.ERR             ; BLOCK DEVICE FOUND
006347         LDA         DNUM.TEMP
006348         RTS
006349 *
006350 GETD.ERR         LDA         #FNFERR
006351         JSR         SYSERR
006352         PAGE
006353         REP         60
006354 * REQUEST FCB ENTRY
006355 *
006356 * INPUT:  DNUM (A)
006357 * OUTPUT: REFNUM (A), OPENCT (Y)
006358 * ERROR:  CARRY SET ("CFCB FULL")
006359 *
006360 * REQ.CFCB FIRST SEARCHES THE CFCB TABLE USING THE DEV#
006361 * AS A KEY.  IF FOUND THE OPENCT IS INCREMENTED, OTHERWISE,
006362 * REQ.CFCB FINDS A FREE ENTRY AND STORES THE DEV# AND LEVEL #.
006363 *
006364         REP         60
006365 *
006366 REQ.CFCB         EQU         *
006367         LDX         #CFCB.MAX-1
006368         TAY
006369 REQ010         LDA         CFCB.DEV,X

```

```

006370          BMI          REQ020
006371          DEX
006372          BNE          REQ010
006373          LDA          #CFCBFULL
006374          JSR          SYSERR
006375  REQ020    TYA
006376          STA          CFCB.DEV,X
006377          LDA          LEVEL
006378          STA          CFCB.LVL,X
006379          TXA
006380          PHA
006381          TYA
006382          JSR          OPENCOUNT
006383          PLA
006384          ORA          #$80
006385          CLC
006386          RTS          ; NORMAL EXIT
006387          PAGE
006388          REP          60
006389  *
006390  * RELEASE FCB ENTRY
006391  *
006392  * INPUT:  REFNUM (A)
006393  * OUTPUT: DNUM (A), OPENCT (Y)
006394  * ERROR:  CARRY SET ("INVALID REFNUM")
006395  *
006396  * USES REFNUM AS AN CFCB TABLE INDEX TO RELEASE A CFCB ENTRY.
006397  *
006398          REP          60
006399  REL.CFCB  EQU          *
006400          AND          #$7F
006401          CMP          #CFCB.MAX
006402          BCS          REL.ERR
006403          TAX
006404          LDA          CFCB.DEV,X
006405          BMI          REL.ERR
006406          SEC          ; MARK ENTRY FREE
006407          ROR          CFCB.DEV,X
006408          JSR          OPENCOUNT
006409          CLC
006410          RTS          ; NORMAL EXIT
006411  *
006412  REL.ERR   LDA          #BADREFNUM
006413          JSR          SYSERR
006414          REP          60
006415  *
006416  * OPENCOUNT SUBROUTINE
006417  *
006418  * INPUT:  DEVNUM (A)
006419  * OUTPUT: DEVNUM (A), OPENCTR (Y)

```

```

006420 *
006421 * OPENCTR:=COUNT OF ALL CFCB ENTRIES W/CFCB.DEV=DEVNUM
006422 *
006423             REP      60
006424 OPENCOUNT EQU      *
006425             LDY      #0
006426             LDX      #CFCB.MAX-1
006427 OPNCT010  CMP      CFCB.DEV,X
006428             BNE      OPNCT020
006429             INY
006430 OPNCT020  DEX
006431             BNE      OPNCT010
006432             RTS
006433             PAGE
006434             REP      60
006435 *
006436 * GET FCB ENTRY
006437 *
006438 * INPUT:  REFNUM (A)
006439 * OUTPUT: DNUM (A)
006440 * ERROR:  CARRY SET ("INVALID REFNUM")
006441 *
006442 * USES REFNUM AS AN INDEX TO RETURN THE CORRESPONDING DEVICE #.
006443 * IF THE ENTRY INDICATED BY REFNUM IS A FREE ENTRY, THEN AN
006444 * ERROR, "INVALID REF NUM" IS RETURNED.
006445 *
006446             REP      60
006447 GET.CFCB    EQU      *
006448             AND      #$7F
006449             CMP      #CFCB.MAX
006450             BCS      GET.ERR
006451             TAX
006452             LDA      CFCB.DEV,X
006453             BMI      GET.ERR
006454             CLC
006455             RTS                ; NORMAL EXIT
006456 *
006457 GET.ERR     LDA      #BADREFNUM
006458             JSR      SYSERR                ; ERR EXIT
006459 *
006460             LST      ON
006461 ZZEND      EQU      *
006462 ZZLEN      EQU      ZZEND-ZZORG
006463             IFNE     ZZLEN-LENCFM
006464             FAIL     2,"SOSORG          FILE IS INCORRECT FOR CFMGR"
006465             FIN
006466
006467 *****
006468 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: CFMGR.SRC
006469 *****

```

006470  
006471

```

006472 =====
006473 DOCUMENT :SOS1.3.2of5.TWO:SOS.D3MAIN.TEXT
006474 =====
006475
006476 *****
006477 * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.MAIN.SRC
006478 *****
006479 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
006480
006481 PAGE
006482 * MAIN ENTRY POINT:
006483 *
006484 * DISABLE NMI/RESET AND ENABLE ROM/IO SPACE
006485 *
006486 MAIN EQU *
006487 LDA E.REG ;SAVE CALLER'S
006488 AND #$FF-$20 ;DROP SCREEN BIT
006489 STA ESAVE ; ENVIRONMENT
006490 DO 1-TEST ;NO RESETLOCK FOR TESTING
006491 LDA E.REG ;GET EREG AGAIN
006492 AND #$FF-$10 ;DISABLE NMI/RESET
006493 FIN
006494 ORA #$03 ;ENABLE ROM/IO SPACE
006495 STA E.REG
006496 *
006497 LDA NOSCROLL ;DISABLE SMOOTHSCROLL
006498 *
006499 PHP ;IF ALREADY SEI'D, THEN WE
006500 PLA ; STAY THAT WAY...
006501 ROR A
006502 ROR A
006503 ROR A
006504 ROR A
006505 STA IRQMASK ;'I' BIT INTO BIT7
006506 *
006507 * MAKE SURE WE HAVE A VALID COMMAND:
006508 *
006509 LDA D.COMMAND ;GET IT
006510 BMI BADCMD ;=>WOW!
006511 BEQ IOSETUP ;=>ZERO IS A READ
006512 CMP #10 ;OFF THE END?
006513 BCS BADCMD ;=>YES
006514 CMP #9 ;REPEAT?
006515 BNE CMD1 ;=>NOPE
006516 *
006517 * REPEAT. SIMPLY GET PRIOR COMMAND:
006518 *
006519 LDA PREVUNIT ;IS THIS REPEAT FOR
006520 CMP D.UNITNUM ; SAME UNIT?

```

```

006521          BNE      BADOP          ;=>NO? ILLEGAL!
006522          LDA      PREVCMO        ;YES, SET COMMAND
006523          BEQ      RPTOK          ;=>REPEAT'ED READ IS OK
006524          CMP      #1              ;IF NOT, IS IT REPEAT'ED WRITE?
006525          BNE      BADOP          ;=>CAN'T REPEAT OTHER COMMANDS
006526 RPTOK      EQU      *
006527          STA      D.COMMAND        ;SAME AS BEFORE
006528          CMP      #0              ;READ?
006529          BEQ      IOSETUP         ;=>YES
006530 * NOW REPEAT GOES LIKE OTHERS:
006531 *
006532 *
006533 CMD1      EQU      *
006534          CMP      #1              ;WRITE?
006535          BNE      CMD2              ;=>NOPE
006536          JMP      IOSETUP         ;=>YES
006537 CMD2      EQU      *
006538          CMP      #2              ;STATUS?
006539          BNE      CMD3              ;=>NOT STATUS
006540          LDA      D.STATCODE        ;IS IT 'SENSE'?
006541          BEQ      GOSTAT          ;=>YES
006542          LDA      #XCTLCODE        ;ILLEGAL CODE
006543          JMP      EXIT
006544 GOSTAT     EQU      *
006545          JMP      DRVSETUP         ;=>YES
006546 *
006547 CMD3      EQU      *
006548          CMP      #8              ;INIT?
006549          BNE      BADOP          ;=>NOPE
006550          JMP      INIT              ;=>YES, DO INIT
006551 *
006552 BADOP      EQU      *
006553          LDA      #XBADOP          ;ILLEGAL COMMAND
006554          JMP      EXIT              ;BACK TO YOU
006555 *
006556 BADCMD     EQU      *
006557          LDA      #XREQCODE        ;INVALID COMMAND
006558          JMP      EXIT              ;BACK TO YOU
006559          PAGE
006560 * SETUP WHAT WE HAVE TO BEFORE
006561 * PERFORMING THE I/O OPERATION:
006562 *
006563 IOSETUP     EQU      *
006564          LDA      D.BLOCK+1        ;VALIDATE BLOCKNUM
006565          BEQ      CHKBYTE          ;=> IF <256, IT'S OK
006566          CMP      #2              ;IS IT <512?
006567          BCS      BADBLOCK        ;=>BAD BOY!
006568          LDA      D.BLOCK          ;YES, CHECK LO HALF
006569          CMP      #280-256        ; FOR RANGE
006570          BCC      CHKBYTE          ;=>IT'S OK

```

```

006571  BADBLOCK      EQU      *
006572                LDA      #XBLKNUM      ;BAD BLOCK NUMBER
006573                JMP      EXIT          ;RETURN BAD NEWS
006574  *
006575  CHKBYTE        EQU      *
006576                LDA      D.BYTES      ;GET LO COUNT
006577                BNE      BADCOUNT    ;=>ERR, NOT INTEGRAL BLOCK(S)
006578                LDA      D.BYTES+1    ;GET HI COUNT
006579                LSR      A            ;MAKE BLOCK COUNT
006580                BCS      BADCOUNT    ;=>BAD IF HALF-BLOCK COUNT
006581                STA      BLKCOUNT    ;SAVE COUNT OF BLOCKS
006582  *
006583  * DOES REQUESTED BYTECOUNT CAUSE US
006584  *   TO RUN OFF END OF DISK?
006585  *
006586                LDA      BLKCOUNT    ;NO. ADD STARTBLOCK
006587                CLC                    ; AND BLKCOUNT AND SEE
006588                ADC      D.BLOCK      ; IF WE'RE TOO BIG
006589                LDX      D.BLOCK+1    ;DID IT START OUT > 255?
006590                BNE      BLKG255     ;=>YES
006591                BCC      DRVSETUP     ;=>DEFINITELY < 256
006592                BCS      CHKLO       ;=>IF CARRY, THEN >256
006593  BLKG255        EQU      *
006594                BCS      BADCOUNT    ;>255+CARRY IS NOW >511
006595  CHKLO          EQU      *
006596                CMP      #280-256+1  ;281..511 ?
006597                BCC      DRVSETUP     ;=>NO, WE ARE OK
006598  BADCOUNT     EQU      *
006599                LDA      #XBYTECNT    ;ILLEGAL BYTECOUNT
006600                JMP      EXIT          ;SORRY...
006601                PAGE
006602  *
006603  * SELECT THE APPROPRIATE DRIVE:
006604  *
006605  DRVSETUP        EQU      *
006606                LDA      D.COMMAND    ;SAVE THIS COMMAND
006607                STA      PREVCMND     ; AND DEVICE FOR
006608                LDA      D.UNITNUM    ; SUBSEQUENT
006609                STA      PREVUNIT     ; 'REPEAT' CALL
006610                LDA      E.REG       ;DOWNSHIFT TO
006611                ORA      #$80        ; 1MHZ FOR REMAINDER
006612                STA      E.REG       ; OF DRIVER EXECUTION
006613                JSR      UNITSEL     ;SELECT & START IT
006614  *
006615  * SEE IF THE MOTOR STARTED. IF NOT,
006616  * THEN IT'S EITHER DISKSWITCH OR NODRIVE.
006617  *
006618                JSR      CHKDRV       ;MOTOR RUNNING?
006619                BNE      DOIO         ;=>YES, GREAT.
006620  *

```

```

006621 * IF WE GET A MOTOR WHEN WE MOVE
006622 * THE HEAD, THEN IT'S DISKSWITCH.
006623 *
006624         LDX      D.UNITNUM           ;FORCE HEAD MOTION
006625         INC      DRVTRACK,X          ; EVEN IF ALREADY ON ZERO
006626         INC      DRVTRACK,X          ;GIVE HIM A FIRM KNOCKER
006627         LDA      #0                  ;SEEK TO TRACK ZERO
006628         JSR      MYSEEK              ; FOR BFM DIR READ
006629         JSR      CHKDRV              ;RUNNING NOW?
006630         BNE      DSWITCH            ;=>YES, A SWITCHEROO
006631         LDA      #0
006632         LDY      D.UNITNUM           ;FORGET THAT THIS
006633         STA      DRIVESEL,Y          ; DRIVE WAS 'SELECTED'
006634         LDA      #XNODRIVE          ;NO, A MISSING DRIVE!
006635         JMP      EXIT
006636 *
006637 DSWITCH      EQU      *
006638         LDA      #XDISKSW            ;USER PULLED A FAST ONE
006639         JMP      EXIT                ; BUT HE CAN'T FOOL US.
006640         PAGE
006641 * PREPARE TO DO THE OPERATION:
006642 *
006643 DOIO         EQU      *
006644         LDA      D.BUFL              ;COPY USER BUFFER
006645         STA      BUFTEMP              ; AND BLOCK NUMBER
006646         LDA      D.BUFH              ; TO OUR WORKSPACE
006647         STA      BUFTEMP+1
006648         LDA      $1400+D.BUFH
006649         STA      $1400+BUFTEMP+1
006650         LDA      D.BLOCK
006651         STA      BLKTEMP
006652         LDA      D.BLOCK+1
006653         STA      BLKTEMP+1
006654 *
006655 * IF CALLER GAVE US A COUNT OF ZERO BYTES,
006656 * THEN WE'RE ALL DONE!
006657 *
006658         LDA      D.COMMAND           ;IS IT STATUS?
006659         CMP      #2                  ;IF SO, THEN BYTECOUNT
006660         BNE      DOIO2              ; IS MEANINGLESS
006661         JMP      STATUS
006662 DOIO2      EQU      *
006663         LDY      BLKCOUNT           ;BLKS=0?
006664         BEQ      READOK              ;=>YES, YOU GET GOOD RETURN
006665         CMP      #0                  ;READ COMMAND?
006666         BEQ      READREQ            ;=>YES
006667         JMP      WRITEREQ
006668         PAGE
006669         REP      40
006670 * -- READ --

```



```

006671          REP          40
006672 READREQ          EQU          *
006673          LDA          #0          ;CLEAR COUNT OF
006674          LDY          #0
006675          STA          (D.BYTRD),Y          ; BYTES READ
006676          INY
006677          STA          (D.BYTRD),Y
006678 READREQ2         EQU          *
006679          JSR          BLK2SECT          ;COMPUTE TRK/SECTOR THIS BLOCK
006680          *
006681          JSR          SECTORIO          ;READ IT PLEASE
006682          BCS          READERR          ;=>WE LOSE.
006683          INC          SECTOR          ;BUMP TO NEXT
006684          INC          SECTOR          ; LOGICAL SECTOR
006685          INC          BUF+1          ;BUMP SECTOR BUFFER
006686          JSR          SECTORIO          ;READ IT TOO
006687          BCS          READERR          ;=>WE LOSE.
006688          LDY          #1
006689          LDA          (D.BYTRD),Y          ;BUMP COUNT OF
006690          CLC
006691          ADC          #2
006692          STA          (D.BYTRD),Y          ; BYTES READ
006693          *
006694          * MORE BLOCKS TO GO?
006695          *
006696          JSR          MOREBLKS          ;SETUP FOR NEXT BLOCK
006697          BNE          READREQ2          ;=>MORE TO READ...
006698 READOK           EQU          *
006699          LDA          #0          ;GOOD RETURN
006700          JMP          EXIT          ;TELL HAPPY USER
006701          *
006702 READERR           EQU          *
006703          JMP          EXIT          ;RETURN ERROR CODE
006704          CHN          DISK3.WRT.SRC
006705
006706          *****
006707          * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.MAIN.SRC
006708          *****
006709
006710

```

```

006711 =====
006712 DOCUMENT :SOS1.3.2of5.TWO:SOS.D3SIO.TEXT
006713 =====
006714
006715 *****
006716 * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SIO.SRC
006717 *****
006718 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
006719
006720 PAGE
006721 REP 40
006722 * NAME : SECTORIO
006723 * FUNCTION: READ OR WRITE A SECTOR
006724 * INPUT : IBSTRK, IBSECT, MONTIME,
006725 * RETURNS : CARRY CLEAR IF OK (AC=00)
006726 * : CARRY SET IF ERROR (AC=ERRCODE)
006727 * : SEEKWAIT ALL SETUP
006728 * DESTROYS: ALL REGISTERS
006729 REP 40
006730 *
006731 SECTORIO EQU *
006732 LDA #R.RECAL ;SETUP THE
006733 * R.RECAL MUST BE NON-ZERO!! (SEE BELOW)
006734 STA RECALCNT ; RECAL TRIES
006735 NOP ; PAD ONE BYTE
006736 STA E1908 ; A-REG MUST BE NON-ZERO !!!
006737 * E1908 = NON-ZERO LOCKOUT MOUSE
006738 *
006739 LDY D.UNITNUM ;ARE WE ON-TRACK?
006740 LDA TRACK
006741 CMP DRVTRACK,Y
006742 BEQ SOUGHT ;=>IF SO, FORGET SEEK & DELAY!
006743 *
006744 * WAIT BEFORE STEPPING:
006745 *
006746 LDA SEEKWAIT ;SEEK DELAY NEEDED?
006747 BEQ GOSEEK ;=>NAW...
006748 LDA #0
006749 STA SEEKWAIT ;CLEAR THE FLAG
006750 LDA #4 ;ADD SEEKDELAY TO
006751 JSR ADDTIME ; THE TOTAL UPTIME(S)
006752 TAY ;4*25 MS DELAY
006753 SEEKDEL EQU *
006754 LDA #0
006755 JSR MSWAIT
006756 DEY
006757 BNE SEEKDEL
006758 *
006759 * ISSUE THE SEEK:

```

```

006760 *
006761 GOSEEK      EQU      *
006762             LDA      TRACK      ;GET DESTINATION TRACK
006763             JSR      MYSEEK     ;=>..AND YOU SHALL FIND...
006764 *
006765 SOUGHT      EQU      *
006766             LDA      IRQMASK     ;SET IRQ MASK FOR
006767             STA      IMASK       ; CORE ROUTINES
006768             LDA      #R.IRQ     ;SETUP IRQ RETRIES
006769             STA      INTRTRY
006770             LDA      #R.IOERR    ; AND ERROR RETRIES
006771             STA      RETRYCNT
006772 *
006773 * DELAY FOR ANY REMAINING MOTOR-UP TIME:
006774 *
006775 MDELAY      EQU      *
006776             LDA      MONTIMEH    ;ANY TIME REMAINING?
006777             BPL      FINDIT     ;=>NO, WE'RE UP TO SPEED.
006778             LDA      #1         ;YES, SO BUMP A SLICE OF
006779             JSR      ADDTIME    ; UPTIME WHILE WE WAIT
006780             LDA      #0
006781             JSR      MSWAIT
006782             JMP      MDELAY     ;=>GO TILL ENOUGH
006783 *
006784 * FIND THE DESIRED SECTOR:
006785 *
006786 * NOTE: FINDSECT RETURNS WITH
006787 *       IRQ INHIBITED!
006788 *
006789 FINDIT      EQU      *
006790             PHP              ;INHIBIT IRQ WHILE
006791             SEI              ; MESSING WITH VBL FLAGS
006792             LDA      E.IER     ;DISABLE VBL IRQ
006793             AND      #$18     ; DURING SECTOR I/O
006794             STA      E.IER
006795             ORA      #$80     ;FOR 'SET' LATER
006796             STA      VBLSAVE
006797             PLP              ;RESTORE IRQ STATUS
006798             JSR      FINDSECT  ;FIND ME PLEASE
006799             BCS      TRYRECAL  ;=>NO? RECAL OR GIVE UP!
006800             LDX      #$60     ;SET UP SLOT FOR CORE RTNS
006801             LDA      D.COMMAND ;WHAT'S YOUR PLEASURE?
006802             BNE      SIOWRITE  ;=>WRITE
006803 *
006804             REP      40
006805 * READ A SECTOR:
006806 *
006807             JSR      READ      ;READ THAT SECTOR
006808             JSR      FIXIRQ    ;ENABLE IRQ IF OK
006809             LDA      VBLSAVE   ;ALLOW VBL DURING

```

```

006810          STA      E.IER          ; POSTNIB
006811          BCS      BADIO          ;=>I/O ERR OR IRQ
006812          LDA      E.REG          ;SET 2MHZ FOR POSTNIB
006813          AND      #$7F
006814          STA      E.REG
006815          JSR      POSTNIB         ;POSTNIB/CHECKSUM
006816          BCS      IORETRY        ;=>I/O ERR:BAD CHKSUM
006817          JMP      SIOGOOD        ;=>GOOD READ
006818          *
006819          REP      40
006820          * WRITE A SECTOR:
006821          *
006822          SIOWRITE EQU      *
006823          JSR      WRITE           ;WRITE THE DATA
006824          JSR      FIXIRQ         ;RE-ENABLE IRQ IF OK
006825          LDA      VBLSAVE        ;RESTORE
006826          STA      E.IER          ; VBL IRQ
006827          BCC      SIOGOOD        ;=>GOOD WRITE
006828          BVC      SIOWPROT       ;=>WRITE PROTECTED
006829          *
006830          REP      40
006831          * IT DIDN'T GO WELL FOR US:
006832          *
006833          BADIO    EQU      *
006834          DO      1-REV0ROM        ;FOR REV1
006835          BVS      FINDIT          ;=>IRQ. JUST RETRY IT.
006836          ELSE
006837          *
006838          * THE REV1 ROM TAKES CARE OF THE
006839          * IRQ RETRY COUNT, BUT REV0 DOESN'T:
006840          *
006841          BVC      IORETRY          ;=>I/O ERROR. RETRY IT
006842          LDA      ROMREV          ;WHICH ROM?
006843          BNE      FINDIT          ;=>REV1. HE DOES IT.
006844          LDA      INTRTRY        ;REV0. OUT OF RETRIES?
006845          BPL      BADIO2         ;=>NO.
006846          STA      IMASK          ;SET HI BIT FOR IRQ MASK
006847          BADIO2  EQU      *
006848          DEC      INTRTRY         ;ONE LESS RETRY
006849          JMP      FINDIT         ;=>RETRY AFTER IRQ
006850          FIN
006851          *
006852          * RETRY AFTER AN I/O ERROR:
006853          *
006854          IORETRY  EQU      *
006855          DEC      RETRYCNT        ;ANY RETRIES LEFT?
006856          BNE      FINDIT        ;=>YEAH, RETRY AFTER ERROR
006857          *
006858          * RETRIES EXHAUSTED. RECALIBRATE:
006859          *

```

```

006860 TRYRECAL      EQU      *
006861              LDA      VBLSAVE      ;ALLOW VBL IF RECAL
006862              STA      E.IER        ; OR UNRECOVERABLE ERROR
006863              DEC      RECALCNT     ;HAVE WE RECALIBRATED YET?
006864              BMI      SIOERR      ;=>YUP. WE'RE DEAD.
006865              JSR      RECAL        ;NO, TRY OUR LUCK
006866              LDY      D.UNITNUM    ;ARE WE ON-TRACK?
006867              LDA      TRACK
006868              CMP      DRVTRACK,Y
006869              BNE      NOTSAME
006870              JMP      SOUGHT      ;=>IF SO, FORGET RESEEK
006871 NOTSAME      EQU      *
006872              JMP      GOSEEK      ;TRY AGAIN ON TARGET TRACK
006873 *
006874              REP      40
006875 SIOERR        EQU      *
006876              LDA      #XIOERROR    ;RETURN CODE
006877              SEC                ;INDICATE HARD ERROR
006878              BCS      SIORET
006879 SIOWPROT     EQU      *
006880              LDA      #XNOWRITE    ;RETURN CODE
006881              SEC                ;INDICATE HARD ERROR
006882              BCS      SIORET
006883 SIOGOOD      EQU      *
006884              LDA      #0
006885              CLC                ;INDICATE GOOD COMPLETION
006886 SIORET       LDX      #0          ; SAY OK TO MOUSE
006887              STX      E1908       ; WITH THIS GLOBAL $1908
006888              RTS
006889              PAGE
006890              REP      40
006891 * NAME      : FINDSECT
006892 * FUNCTION: LOCATE A DESIRED SECTOR
006893 * INPUT    : IBTRK, IBSECT SETUP
006894 * RETURNS  : CARRY CLEAR IF OK,
006895 *          : CARRY SET IF ERROR.
006896 * DESTROYS: ALL REGISTERS & 'TEMP'
006897 * NOTE    : RETURNS WITH IRQ DISABLED IF NO ERROR!
006898              REP      40
006899 *
006900 FINDSECT      EQU      *
006901              LDA      #R.FIND*16    ;SETUP NUMBER OF REVS
006902              STA      RETRYADR      ; ALLOWED TO FIND SECTOR
006903              LSR      TEMP        ;COMPUTE LATENCY FIRST TIME THRU
006904 FINDSEC2     EQU      *
006905              LDX      #$60          ;FAKE SLOT FOR CORE ROUTINES
006906              JSR      RDADR        ;GET NEXT ADDRESS FIELD
006907              BCS      RDADERR      ;=>UGH! AN ERROR!
006908 *
006909 * MAKE SURE WE'RE ON THE CORRECT TRACK:

```

```

006910 *
006911     LDA     TRACK           ;IS IT
006912     CMP     CSSTV+2       ; CORRECT TRACK?
006913     BNE     FINDERR      ;=>NO?!? IT'S USELESS!
006914     LDA     SECTOR       ;IS IT
006915     CMP     CSSTV+1       ; DESIRED SECTOR?
006916     BEQ     FINDGOOD     ;=>YEAH. GOT IT!
006917 *
006918 * COMPUTE LATENCY. EACH TWO-SECTOR
006919 * DISTANCE IS 25 MS OF UPTIME.
006920 *
006921     LDA     TEMP           ;LATENCY ALREADY COMPUTED?
006922     BMI     RDADERR       ;=>YES.
006923     LDA     SECTOR       ;HOW FAR AWAY IS OUR
006924     SEC     SEC           ; DESIRED SECTOR?
006925     ROR     TEMP         ;PREVENT RECOMPUTATION
006926     SBC     CSSTV+1
006927     AND     #$0F
006928     LSR     A            ;EACH 2-SECTORS IS 25 MS
006929     JSR     ADDTIME
006930 *
006931 * KEEP LOOKING TILL WE FIND IT:
006932 *
006933 RDADERR     EQU     *
006934     JSR     FIXIRQ       ;ENABLE IRQ IF APPROPRIATE
006935     DEC     RETRYADR     ;ANY RETRIES LEFT?
006936     BEQ     FINDERR     ;=>NO, WE CAN'T FIND IT.
006937 *
006938 * COMPENSATE FOR A BUG IN RDADR: IF WE TRY
006939 * TO CALL RDADR AGAIN BEFORE THE DATA MARK
006940 * GOES BY, THEN RDADR WILL ACCIDENTALLY CALL
006941 * THAT AN ERROR. WE CAN AVOID THIS 'FAKE'
006942 * ERROR BY DELAYING PAST THE DATA MARK.
006943     LDY     #200         ;1 MS IS PLENTY
006944 ADRDELAY     EQU     *
006945     DEY
006946     BNE     ADRDELAY
006947     JMP     FINDSEC2     ;=>NOW TRY LOOKING AGAIN
006948 *
006949     REP     40
006950 FINDGOOD     EQU     *
006951     LDA     #0           ;CLEAR VOLNUM OUT OF
006952     STA     MONTIMEH     ; MOTORTIME!
006953     CLC
006954     RTS
006955 *
006956 FINDERR     EQU     *
006957     JSR     FIXIRQ       ;ENABLE IRQ IF APPROPRIATE
006958     LDA     #0           ;CLEAR VOLNUM OUT OF
006959     STA     MONTIMEH     ; MOTORTIME!

```

```
006960          SEC          ;INDICATE THE ERROR
006961          RTS
006962
006963          CHN          DISK3.USEL.SRC
006964
006965 *****
006966 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SIO.SRC
006967 *****
006968
006969
006970
```

```

006971 =====
006972 DOCUMENT :SOS1.3.2of5.TWO:SOS.D3SRC.TEXT
006973 =====
006974
006975 *****
006976 * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SRC
006977 *****
006978 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
006979
006980          SBTL          'SOS 1.1  DISK ///  DRIVER'
006981 TEST          EQU          0                      ;FOR FUNNY-MODE TESTING
006982          INCLUDE     SOSORG,6,1,254
006983          DO          TEST
006984          ORG          $2000
006985          ELSE
006986          REL
006987          ORG          ORGDISK3
006988          FIN
006989 ZZORG       EQU          *
006990          CHR          '- '
006991          MSB         OFF
006992 *
006993          REP          40
006994 *      COPYRIGHT (C) APPLE COMPUTER INC.
006995 *      ALL RIGHTS RESERVED
006996          REP          40
006997 *
006998 REVOROM      EQU          0                      ;1=SUPPORT REV0 ROM
006999 *
007000          DO          1-TEST
007001          ENTRY      DIB1                      ;DIB1
007002          ENTRY      DIB2                      ;DIB2
007003          ENTRY      DIB3                      ;DIB3
007004          ENTRY      DIB4                      ;DIB4
007005          ENTRY      SEEKDSK3                  ;SEEK CURRENT DRIVE
007006 *
007007          EXTRN      SYSERR
007008 *
007009          EXTRN      XREQCODE
007010          EXTRN      XBADOP
007011          EXTRN      XNODRIVE
007012          EXTRN      XIOERROR
007013          EXTRN      XNOWRITE
007014          EXTRN      XBYTECNT
007015          EXTRN      XBLKNUM
007016          EXTRN      XDISKSW
007017          EXTRN      XCTLCODE
007018 *
007019          EXTRN      E1908                      ; GLOBAL FLAG FOR MOUSE DRIVER

```



```

007020 * TO SAY WE CANNOT BE INTERRUPTED
007021 *
007022 ELSE
007023 XREQCODE EQU $20
007024 XBADOP EQU $26
007025 XNODRIVE EQU $28
007026 XIOERROR EQU $27
007027 XNOWRITE EQU $2B
007028 XBYTECNT EQU $2C
007029 XBLKNUM EQU $2D
007030 XDISKSW EQU $2E
007031 XCTLCODE EQU $21
007032 FIN
007033 PAGE
007034 * DISK /// CONTROLLER EQUATES:
007035 *
007036 * MOTOR SELECT BITS:
007037 *
007038 * DRIVE INT EXT1 EXT2
007039 * -----
007040 * .D1 1 X X
007041 * .D2 X 0 1
007042 * .D3 X 1 0
007043 * .D4 X 1 1
007044 *
007045 MS.INT EQU $C0D4 ;MOTOR SELECT:INTERNAL DRIVE
007046 MD.INT EQU $C0D5 ;MOTOR DESELECT:INTERNAL DRIVE
007047 *
007048 MS.EXT1 EQU $C0D3 ;MOTOR SELECT:EXTERNAL DRIVE
007049 MS.EXT2 EQU $C0D1 ;MOTOR SELECT:EXTERNAL DRIVE
007050 MD.EXT1 EQU $C0D2 ;MOTOR DESELECT:EXTERNAL DRIVE
007051 MD.EXT2 EQU $C0D0 ;MOTOR DESELECT:EXTERNAL DRIVE
007052 *
007053 IS.INT EQU $C0EA ;I/O SELECT:INTERNAL DRIVE
007054 IS.EXT EQU $C0EB ;I/O SELECT:EXTERNAL DRIVE
007055 *
007056 NOSCROLL EQU $C0D8 ;SMOOTHSCROLL OFF
007057 *
007058 MOTOROFF EQU $C0E8 ;MOTOR(S) START POWEROFF T/O
007059 MOTORON EQU $C0E9 ;MOTOR(S) POWER ON
007060 Q6L EQU $C08C ;Q7L,Q6L=READ
007061 Q6H EQU $C08D ;Q7L,Q6H=SENSE WPROT
007062 Q7L EQU $C08E ;Q7H,Q6L=WRITE
007063 Q7H EQU $C08F ;Q7H,Q6H=WRITE STORE
007064 *
007065 * OTHER EQUATES:
007066 *
007067 E.REG EQU $FFDF ;ENVIRONMENT REGISTER
007068 E.IER EQU $FFEE ;INTERRUPT ENABLE REGISTER
007069 *

```

```

007070 * RETRY COUNTERS:
007071 *
007072 R.RECAL      EQU      1          ;MAX RECALIBRATES
007073 * R.RECAL MUST NOT BECOME ZERO! (MOUSE WILL BE LOCKED OUT)
007074 * SEE DISK3.SIO.SRC LINE 14 FOR DETAIL
007075 R.FIND       EQU      3          ;MAX REVS TO FIND A SECTOR
007076 R.IOERR      EQU      4          ;MAX RETRIES ON READ ERROR
007077 R.IRQ       EQU      6          ;MAX IRQ'S TOLERATED BEFORE SEI
007078 PAGE
007079 * ZPAGE EQUATES FOR CORE ROUTINES:
007080 *
007081           DSECT
007082           ORG      $81
007083 IBSLOT       DS      1          ;SLOT=$60 FOR RTNS
007084           DS      7          ;N/A
007085           DS      1          ;RDADR:CHECKSUM
007086           DS      1          ;N/A
007087 IMASK       DS      1          ;BIT7 SET IF IRQ ALLOWED
007088 CURTRK      DS      1          ;SEEK:CURRENT TRACK
007089           DS      2          ;N/A
007090 INTRTRY     DS      1          ;READ:  IRQ RETRY COUNT
007091           DS      5          ;N/A
007092           DS      1          ;RDADR:'MUST FIND' COUNT
007093           DS      1          ;READ,WRITE: CHECKSUM
007094 CSSTV       DS      4          ;RDADR:CKSUM, SEC, TRK, VOL
007095 MONTIMEL    EQU      CSSTV+2    ;MSWAIT:MOTOR-ON TIME
007096 MONTIMEH    EQU      MONTIMEL+1
007097 BUF        DS      2          ;PRENIB,POSTNIB:USER BUFFER
007098           DS      1          ;SEEK:PRIOR PHASE
007099 TRKN       DS      1          ;SEEK:TARGET TRACK
007100 *
007101 * LOCAL TEMPS:
007102 *
007103           ORG      $D0          ;WE'RE ALLOWED TO $FF
007104 BLKTEMP     DS      2          ;LOCAL TEMP FOR BLKNUMBER
007105 BUFTEMP     DS      2          ;LOCAL TEMP FOR BUFFER ADDRESS
007106 TRACK      DS      1          ;LOCAL TEMP FOR TRACK
007107 SECTOR     DS      1          ;LOCAL TEMP FOR SECTOR
007108 RETRYADR    DS      1          ;LOCAL TEMP FOR SECTOR-FIND RETRIES
007109 RETRYCNT    DS      1          ;LOCAL TEMP FOR I/O RETRIES
007110 RECALCNT    DS      1          ;LOCAL TEMP FOR RECAL COUNT
007111 BLKCOUNT   DS      1          ;BLKS REQD TO SATISFY BYTECOUNT
007112 SEEKWAIT    DS      1          ;<>0 IF SEEK DELAY NEEDED
007113 IRQMASK     DS      1          ;ENTRY 'I' BIT
007114 TEMP       DS      1          ;JUST A TEMP
007115           DEND
007116 PAGE
007117 * DRIVER INTERFACE AREA:
007118 *
007119           DSECT

```

```

007120          ORG          $C0
007121 D.COMMAND      DS          1          ;COMMAND CODE
007122 D.UNITNUM      DS          1          ;UNIT NUMBER
007123 D.BUFL         DS          2          ;BUFFER ADDRESS
007124 D.BUFH         EQU        D.BUFL+1
007125 D.STATCODE     EQU        D.BUFL          ;DSTATUS CODE
007126 D.STATBUF      EQU        D.BUFH          ;^DSTATUS LIST
007127 D.BYTES        DS          2          ;BYTECOUNT
007128 D.BLOCK        DS          2          ;REQUESTED BLOCKNUM
007129 D.BYTRD        DS          2          ;BYTES READ (READ)
007130          DS          6          ;SPARES (OK AS TEMPS)
007131          DEND
007132          PAGE
007133 DIB1           EQU        *          ;DIB FOR .D1
007134          DW          DIB2          ;FLINK
007135          DW          MAIN          ;ENTRY POINT
007136          DFB         3          ;NAME LENGTH
007137          ASC         '.D1          '
007138          DFB         $80          ;DEVNUM: ACTIVE
007139          DFB         0          ;SLOT
007140          DFB         0          ;UNIT NUMBER
007141          DFB         $E1,1,0      ;TYPE,SUB,FILLER
007142          DW          280          ;BLOCKCOUNT
007143          DW          1          ;MANUFACTURER=APPLE
007144          DW          $1100        ;VERSION=1.1
007145 *
007146 DIB2           EQU        *          ;DIB FOR .D2
007147          DW          DIB3          ;FLINK
007148          DW          MAIN          ;ENTRY POINT
007149          DFB         3          ;NAME LENGTH
007150          ASC         '.D2          '
007151          DFB         $80          ;DEVNUM: ACTIVE
007152          DFB         0          ;SLOT
007153          DFB         1          ;UNIT NUMBER
007154          DFB         $E1,1,0      ;TYPE,SUB,FILLER
007155          DW          280          ;BLOCKCOUNT
007156          DW          1          ;MANUFACTURER=APPLE
007157          DW          $1100        ;VERSION=1.1
007158 *
007159 DIB3           EQU        *          ;DIB FOR .D3
007160          DW          DIB4          ;FLINK
007161          DW          MAIN          ;ENTRY POINT
007162          DFB         3          ;NAME LENGTH
007163          ASC         '.D3          '
007164          DFB         $80          ;DEVNUM: ACTIVE
007165          DFB         0          ;SLOT
007166          DFB         2          ;UNIT NUMBER
007167          DFB         $E1,1,0      ;TYPE,SUB,FILLER
007168          DW          280          ;BLOCKCOUNT
007169          DW          1          ;MANUFACTURER=APPLE

```

```

007170          DW          $1100          ;VERSION=1.1
007171          *
007172  DIB4      EQU          *           ;DIB FOR .D4
007173          DW          0              ;NO FLINK
007174          DW          MAIN          ;ENTRY POINT
007175          DFB          3              ;NAME LENGTH
007176          ASC          '.D4          '
007177          DFB          $80            ;DEVNUM: ACTIVE
007178          DFB          0              ;SLOT
007179          DFB          3              ;UNIT NUMBER
007180          DFB          $E1,1,0        ;TYPE,SUB,FILLER
007181          DW          280            ;BLOCKCOUNT
007182          DW          1              ;MANUFACTURER=APPLE
007183          DW          $1100          ;VERSION=1.1
007184          DW          1              ;MANUFACTURER=APPLE
007185          DW          $1100          ;VERSION=1.1
007186
007187          CHN          DISK3.MAIN.SRC
007188          INCLUDE     SOSORG,6,1,254
007189
007190          *****
007191          * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SRC
007192          *****
007193
007194
007195

```

```

007196 =====
007197 DOCUMENT :SOS1.3.2of5.TWO:SOS.D3SUBS.TEXT
007198 =====
007199
007200 *****
007201 * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SUBS.SRC
007202 *****
007203 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
007204
007205             PAGE
007206             REP          40
007207 * NAME      : CHKDRV
007208 * FUNCTION: CHECK IF MOTOR(S) RUNNING
007209 * INPUT     : NONE
007210 * RETURNS  : 'BNE' IF RUNNING
007211 *          : 'BEQ' IF NOT
007212 * DESTROYS: AC,X
007213             REP          40
007214 * NOTES: DUE TO A FLOATING PIN, THERE
007215 *   COULD BE A GLITCH WHICH CAUSES THE
007216 *   SHIFTER TO 'FLASH' ONTO THE BUS
007217 *   INSTEAD OF ALWAYS BEING TRISTATED.
007218 *   THIS COULD CAUSE CHKDRV TO THINK
007219 *   THAT THE MOTOR IS SPINNING WHEN IT
007220 *   IS NOT. THUS WE WILL SAMPLE THE SHIFTER
007221 *   FOR 40 US AT 6-US INTERVALS. IF, AFTER
007222 *   THREE (3) CONSECUTIVE PASSES, ANY OF
007223 *   THE PASSES SEES A 'LOCKED' SHIFTER,
007224 *   THEN WE SAY THE DRIVE IS STOPPED.
007225 *
007226 *
007227 CHKDRV      EQU          *
007228             LDX          #3              ;CHECK SHIFTER SEVERAL TIMES
007229 CHKD1       EQU          *
007230             LDA          Q6L+$60        ;GET DATA
007231             CMP          Q6L+$60        ;HAS IT CHANGED?
007232             BNE          CHANGED        ;=>YES
007233             CMP          Q6L+$60        ;HAS IT CHANGED?
007234             BNE          CHANGED        ;=>YES
007235             CMP          Q6L+$60        ;HAS IT CHANGED?
007236             BNE          CHANGED        ;=>YES
007237             CMP          Q6L+$60        ;HAS IT CHANGED?
007238             BNE          CHANGED        ;=>YES
007239             CMP          Q6L+$60        ;HAS IT CHANGED?
007240             BNE          CHANGED        ;=>YES
007241             CMP          Q6L+$60        ;HAS IT CHANGED?
007242             BNE          CHANGED        ;=>YES
007243             CMP          Q6L+$60        ;HAS IT CHANGED?
007244             BNE          CHANGED        ;=>YES

```

```

007245          RTS          ;IF EVER LOCKED, IT'S STOPPED
007246 *
007247 CHANGED      EQU      *
007248          DEX
007249          BNE          CHKD1          ;TRY SEVERAL TIMES
007250          DEX          ;SET CC=BNE
007251          RTS          ;RETURN ZFLAG APPROPRIATELY
007252          PAGE
007253          REP          40
007254 * NAME      : ADDTIME
007255 * FUNCTION: ADD TO MOTOR UPTIME(S)
007256 * INPUT   : AC=NO. OF 25 MS INCREMENTS
007257 * DESTROYS: Y
007258          REP          40
007259 *
007260 ADDTIME      EQU      *
007261          PHA          ;PRESERVE AC
007262          LDY          #4          ;TABLE INDEX/COUNT
007263 ADD2        EQU      *
007264          LDA          DRIVESEL-1,Y  ;IS IT SELECTED?
007265          BEQ          ADD3          ;=>NOPE
007266          PLA
007267          PHA          ;RECOVER DELTA-T
007268          CLC
007269          ADC          UPTIME-1,Y    ;ADD TO MOTOR UPTIME
007270          CMP          #T1SEC+2    ;IS IT AT MAX TIME?
007271          BCC          ADD2A        ;=>NO, STORE NEW TIME
007272          LDA          #T1SEC+1    ;YES, SET TO >1 SEC
007273 ADD2A      EQU      *
007274          STA          UPTIME-1,Y
007275 ADD3        EQU      *
007276          DEY
007277          BNE          ADD2          ;=>DO ALL 4 DRIVES
007278 *
007279          PLA          ;RESTORE AC
007280          RTS
007281          PAGE
007282          REP          40
007283 * NAME      : RECAL
007284 * FUNCTION: RECALIBRATE DRIVE HEAD
007285 * INPUT   : NONE
007286 * DESTROYS: ALL REGISTERS
007287 * NOTE    : A 'QUIET' RECALIBRATE IS DONE
007288 *          : USING TWO ITERATIONS. IF WE ARE
007289 *          : LOST, THEN SEEK 48-TRACKS
007290 *          : TOWARD TRACK ZERO. IF WE KNOW
007291 *          : WHAT TRACK WE'RE CURRENTLY
007292 *          : ON (+- 1/2 TRACK), THEN JUST
007293 *          : ADD A LITTLE EXTRA AND SEEK
007294 *          : TO TRACK ZERO. A 48-TRACK

```

```

007295 *      : SEEK WILL ALWAYS GET US BACK
007296 *      : ONTO THE MEDIA, EVEN IF WE
007297 *      : WERE "OFF THE CAM". FROM THAT
007298 *      : POINT, THE 2ND SEEK GETS US
007299 *      : BACK TO TRACK ZERO QUIETLY.
007300      REP      40
007301 *
007302 RECAL      EQU      *
007303      LDA      #2      ;TWO ITERATIONS, PLEASE
007304 RECAL1     EQU      *
007305      PHA      ;SAVE LOOPCOUNT
007306      LDX      #$60    ;SETUP SLOT FOR CORE RTNS
007307      JSR      RDADR    ;WHERE ARE WE?
007308      BCC      RECAL2   ;=>NOW WE KNOW
007309      JSR      RDADR    ;GIVE SECOND SHOT
007310      BCC      RECAL2   ;=>THAT GOT IT
007311      LDA      #48     ;LOST? TRY 48-TRACK SEEK
007312      JMP      RECAL3
007313 RECAL2     EQU      *
007314      LDA      CSSTV+2  ;HERE'S WHERE WE ARE
007315      CLC      ;ADD SOME SO WE GET A
007316      ADC      #3      ; HARDER SEEK TO ZERO
007317 RECAL3     EQU      *
007318      LDY      D.UNITNUM ;THIS IS NOW WHERE
007319      STA      DRVTRACK,Y ; WE ARE
007320      JSR      FIXIRQ   ;ENABLE IRQ IF OK
007321 *
007322      LDA      #0      ;DESTINATION TRACK IS 00
007323      STA      MONTIMEH  ;CLEAR MOTOR-UP TIME SO
007324      STA      MONTIMEL  ; SEEK KNOWS HOW LONG RECAL TAKES
007325      JSR      MYSEEK   ;=>SLAM IT BACK!
007326      PLA      ;HAVE WE DONE IT TWICE?
007327      TAY
007328      DEY
007329      TYA
007330      BNE      RECAL1   ;=>DO TWO ITERATIONS
007331      RTS
007332      PAGE
007333      REP      40
007334 * NAME      : SEEKDSK3
007335 * FUNCTION:  SEEK CURRENT DRIVE
007336 * INPUT      : AC=DESTINATION TRACK
007337 * OUTPUT     : NONE
007338 * DESTROYS:  ALL REGISTERS
007339 * NOTE       : MUST BE CALLED WHILE
007340 *           : MOTOR IS RUNNING, IN
007341 *           : 1MHZ+ROM+IO MODE
007342      REP      40
007343 SEEKDSK3     EQU      *
007344      LDY      PREVUNIT ;GET DRIVENUM

```

```

007345          STY          D.UNITNUM          ;SET IT UP
007346          JSR          MYSEEK            ;MOVE IT!
007347          RTS
007348          REP          40
007349 * NAME      : MYSEEK
007350 * FUNCTION: SEEK TO DESIRED TRACK
007351 * INPUT    : AC=DESTINATION TRACK
007352 * DESTROYS: ALL REGISTERS
007353          REP          40
007354 MYSEEK     EQU          *
007355          STA          TRKN                ;TEMP HOLD OF AC
007356          LDY          D.UNITNUM          ;GET DRIVENUM
007357          LDA          DRVTRACK,Y         ;SETUP CURRENT TRACK
007358          ASL          A                  ;SET IN HALFTRACKS FOR SEEK
007359          STA          CURTRK            ; FOR SEEK ROUTINE
007360          LDX          #$60                ;SET UP SLOT FOR CORE RTNS
007361          LDA          MONTIMEH           ;GET STARTING MOTOR TIME
007362          STA          TEMP
007363 *
007364 * NOTE: IRQ'S WHICH SUSPEND SEEK MAY CAUSE A
007365 * SEEK FAILURE. WE WILL HAVE TO RECALIBRATE
007366 * SINCE WE WON'T BE ON-TRACK. WE CAN NOT GET
007367 * ON A HALFTRACK SINCE SEEK ALLOWS SETTTLING
007368 * TIME OF THE PHASE. BECAUSE VBL IS A SERIOUS
007369 * OFFENDER, WE INHIBIT HIM.
007370 *
007371          PHP                                ;INHIBIT IRQ WHILE
007372          SEI                                ; MESSING WITH VBL FLAGS
007373          LDA          E.IER
007374          AND          #$18
007375          STA          VBLSAVE
007376          STA          E.IER
007377          PLP                                ;RESTORE IRQ STATUS
007378          LDA          TRKN                ;RESTORE DESTINATION TRACK
007379          STA          DRVTRACK,Y         ;DEST IS NOW CURRENT
007380          ASL          A                    ;MAKE IT IN HALFTRACKS
007381          JSR          SEEK                ;GO MOVE THE HEAD...
007382          LDA          VBLSAVE            ;NOW ALLOW THAT
007383          ORA          #$80                ; NASTY
007384          STA          E.IER              ; VBL INTERRUPT
007385 *
007386 * COMPUTE THE TIME USED BY SEEK:
007387 *
007388          LDA          MONTIMEH            ;INCLUDE SEEKTIME IN
007389          SEC
007390          SBC          TEMP
007391          JSR          ADDTIME              ; TOTAL MOTOR UPTIME(S)
007392          RTS
007393          PAGE
007394          REP          40

```



```

007395 * NAME      : BLK2SECT
007396 * FUNCTION: COMPUTE TRACK/SECTOR FOR A BLOCK
007397 *          AND ADJUST BUFFER ADDRESS
007398 * INPUT   : D.BLOCK, D.BUF
007399 * OUTPUT  : TRACK, SECTOR, D.BUF
007400 * DESTROYS: AC,Y
007401          REP          40
007402 *
007403 BLK2SECT      EQU          *
007404          LDA          BLKTEMP+1          ;GET HI BLK HALF
007405          ROR          A                  ;MOVE LO BIT TO CARRY
007406          LDA          BLKTEMP          ;GET LO HALF
007407          ROR          A                  ;COMBINE WITH HI BIT
007408          LSR          A
007409          LSR          A                  ;FINISH OFF DIVIDE-BY-8
007410          STA          TRACK            ;THAT'S THE TRACK
007411          LDA          BLKTEMP          ;GET LO HALF AGAIN
007412          AND          #7
007413          TAY
007414          LDA          SECTABLE,Y        ;GET START SECTOR
007415          STA          SECTOR
007416 *
007417 * ADJUST BUFFER ADDRESS SO THAT I/O
007418 * WON'T WRAPAROUND IN THE BANK:
007419 * (THIS ALGORITHM RIPPED OFF FROM 1.0)
007420 *
007421          LDA          BUFTEMP+1          ;GET BUFFER HI ADDRESS
007422          LDY          $1400+BUFTEMP+1    ; AND XTND BYTE
007423          CMP          #$82              ;IF RAM ADDR >=8200 THEN BUMP TO
007424          BCC          NOADJ             ; NEXT BANK PAIR
007425          CPY          #$80
007426          BCC          NOADJ             ;=>NOT USING BANKPAIR
007427          CPY          #$8F              ;SPECIAL BANK 0?
007428          BEQ          NOADJ             ;=>YES
007429          AND          #$7F              ;DROP HI ADDRESS AND
007430          STA          BUFTEMP+1        ; BUMP BANK NUMBER
007431          INC          $1400+BUFTEMP+1
007432 *
007433 NOADJ          EQU          *
007434          LDA          BUFTEMP+1          ;COPY BUFFER ADDRESS
007435          STA          BUF+1              ; FOR PRE & POSTNIB
007436          LDA          BUFTEMP
007437          STA          BUF
007438          LDA          $1400+BUFTEMP+1
007439          STA          $1400+BUF+1
007440          RTS
007441 *
007442 SECTABLE       DFB          $00,$04,$08,$0C,$01,$05,$09,$0D
007443          PAGE
007444          REP          40

```

```

007445 * NAME      : MOREBLKS
007446 * FUNCTION:  SETUP TO DO NEXT BLOCK
007447 * INPUT      : NONE
007448 * RETURNS   : 'BNE' IF MORE TO DO
007449 *           : 'BEQ' IF NO MORE TO DO
007450 * DESTROYS:  NOTHING
007451          REP          40
007452 *
007453 MOREBLKS      EQU          *
007454          INC          BUFTEMP+1          ;BUMP BUFFER ADDRESS
007455          INC          BUFTEMP+1
007456          INC          BLKTEMP          ;BUMP BLOCK NUMBER
007457          BNE          MORE2
007458          INC          BLKTEMP+1
007459 MORE2        EQU          *
007460          DEC          BLKCOUNT          ;MORE BLOCKS TO GO?
007461          RTS          ;RETURN RESULT OF DEC
007462          SKP          4
007463          REP          40
007464 * NAME      : FIXIRQ
007465 * FUNCTION:  ENABLE IRQ IF APPROPRIATE
007466 * INPUT      : NONE
007467 * DESTROYS:  NOTHING
007468          REP          40
007469 *
007470 FIXIRQ        EQU          *
007471          PHA
007472          LDA          IRQMASK          ;SHOULD IRQ BE ENABLED?
007473          BMI          FIXRET          ;=>NO, LEAVE IT ALONE
007474          CLI          ;ENABLE IRQ
007475 FIXRET        EQU          *
007476          PLA
007477          RTS
007478
007479          CHN          DISK3.DATA.SRC
007480
007481 *****
007482 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.SUBS.SRC
007483 *****
007484
007485
007486

```

```

007487 =====
007488 DOCUMENT :SOS1.3.2of5.TWO:SOS.D3USEL.TEXT
007489 =====
007490
007491 *****
007492 * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.USEL.SRC
007493 *****
007494 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
007495
007496             PAGE
007497             REP             40
007498 * NAME      : UNITSEL
007499 * FUNCTION: SELECT & START A DRIVE,
007500 *             SET UP MOTOR & SEEK DELAYS
007501 * INPUT    : NONE
007502 * OUTPUT   : MONTIME,SEEKTIME
007503 * DESTROYS: ALL REGISTERS
007504             REP             40
007505 *
007506 UNITSEL      EQU            *
007507             LDY            D.UNITNUM           ;GET DRIVENUM
007508             LDA            #0                 ;ASSUME NO SEEKWAIT
007509             STA            SEEKWAIT           ; WILL BE NEEDED
007510             STA            MONTIMEL          ;CLEAR MONTIME
007511             STA            MONTIMEH
007512 *
007513 * SEE IF MOTOR(S) STILL SPINNING:
007514 *
007515             JSR            CHKDRV             ;MOTOR(S) POWERED UP?
007516             BNE            SPINNING          ;=>YES. WHO IS IT?
007517 *
007518 * NO MOTOR(S) SPINNING. DESELECT
007519 * ALL MOTORS AND START AFRESH:
007520 *
007521             LDX            MD.INT            ;DESELECT ALL
007522             LDA            #0                 ;SHOW INTERNAL AS
007523             STA            DRIVESEL+0        ; NOT SELECTED
007524             STA            UPTIME+0          ;INDICATE DRIVE IS FULLY STOPPED
007525             JSR            EXTDESEL          ;DESELECT ALL EXTERNALS TOO
007526             JMP            SETTIME          ;GO SETUP MOTOR DELAY
007527             REP             40
007528 * MOTOR(S) SPINNING: OURS?
007529 *
007530 SPINNING     EQU            *
007531             LDA            DRIVESEL,Y        ;HAD WE BEEN SELECTED?
007532             BNE            GOFORIT          ;=>YES, GO FOR IT RIGHT AWAY.
007533 *
007534 * WE AREN'T SPINNING. SHUTDOWN ANOTHER
007535 * DRIVE, IF NECESSARY, TO GET GOING:

```

```

007536 *
007537         CPY         #0             ;ARE WE THE INTERNAL DRIVE?
007538         BEQ         SETTIME        ;=>YES, LEAVE EXT MOTOR ALONE
007539 *
007540 * WE'RE AN EXTERNAL DRIVE. STOP ALL EXTERNAL MOTORS
007541 * UNCONDITIONALLY, BUT LEAVE THE INTERNAL MOTOR ALONE.
007542 * IF WE *DID* HAVE TO STOP ANOTHER EXTERNAL, THEN
007543 * MAKE SURE WE SET THE CORRECT PRE-SEEK DELAY!
007544 *
007545         LDA         #0             ;SEE IF ANOTHER EXTERNAL
007546         ORA         DRIVESEL+3      ; HAD BEEN
007547         ORA         DRIVESEL+2      ;  SELECTED
007548         ORA         DRIVESEL+1      ;  BEFORE...
007549         BEQ         SETTIME        ;=>NO, SEEK DELAY IS UNNECESSARY
007550         INC         SEEKWAIT        ;YES, DELAY BEFORE STEPPING
007551         JSR         EXTDESEL        ;DESELECT ALL EXTERNALS
007552         JMP         SETTIME        ;=>GO SETUP MOTOR DELAY
007553         PAGE
007554         REP         40
007555 * OUR DRIVE IS SPINNING. GO FOR IT!
007556 * DEPENDING OF HOW LONG THE MOTOR'S BEEN ON,
007557 * THIS COMMAND MAY REQUIRE A MOTOR DELAY.
007558 *
007559 GOFORIT     EQU         *
007560         LDX         D.COMMAND        ;GET CURRENT COMMAND
007561         LDA         MTIME,X         ;GET REQUIRED UPTIME FOR IT
007562         SEC
007563         SBC         UPTIME,Y        ;DRIVE RUNNING LONG ENOUGH?
007564         BCS         SELECT        ;=>NO, AC NOW HAS DELTA-T
007565         LDA         #0             ;OTHERWISE, WAIT=0
007566         JMP         SELECT        ;SET MONTIME & SELECT DRIVE
007567         REP         40
007568 *
007569 * ALL MOTORS WERE OFF. CHOOSE THE
007570 * APPROPRIATE MOTOR-ON TIME:
007571 *
007572 SETTIME     EQU         *
007573         LDA         #0             ;INDICATE THAT
007574         STA         UPTIME,Y        ; THE DRIVE WAS OFF
007575         LDX         D.COMMAND        ;GET CURRENT COMMAND
007576         LDA         MTIME,X         ;GET CORRECT DELAY TIME
007577         REP         40
007578 *
007579 * SELECT THE DRIVE & START IT:
007580 *
007581 SELECT     EQU         *
007582         STA         MONTIMEH        ;NEGATE IT BECAUSE
007583         LDA         #0             ; IT GETS INCREMENTED
007584         SEC         ;  INSTEAD OF
007585         SBC         MONTIMEH        ;  DECREMENTED

```

```

007586          STA      MONTIMEH          ;STUFF MOTOR DELAY
007587          CPY      #1                ;ARE WE THE INTERNAL DRIVE?
007588          BCS      SELEXT            ;=>NO, AN EXTERNAL
007589          LDA      IS.INT            ;I/O SELECT INTERNAL
007590          LDA      MS.INT            ;MOTOR SELECT INTERNAL
007591          JMP      UNITRET           ;=>ALL DONE!
007592          *
007593 SELEXT      EQU      *
007594          LDA      IS.EXT            ;I/O SELECT EXTERNAL
007595          CPY      #2                ;ARE WE 2, 3, OR 4 ?
007596          BCS      NOTD2            ;=>DEFINITELY 3 OR 4
007597          LDA      MD.EXT1           ;MOTOR SELECT
007598          LDA      MS.EXT2           ; ONLY .D2
007599          JMP      UNITRET           ;=>ALL DONE!
007600          *
007601 NOTD2      EQU      *
007602          BNE      ISD4                ;=>DEFINITELY NOT 3
007603          LDA      MS.EXT1           ;MOTOR SELECT
007604          LDA      MD.EXT2           ; ONLY .D3
007605          JMP      UNITRET           ;=>ALL DONE!
007606          *
007607 ISD4       EQU      *
007608          LDA      MS.EXT1           ;MOTOR SELECT
007609          LDA      MS.EXT2           ; ONLY .D4
007610          *
007611          *
007612 UNITRET     EQU      *
007613          LDA      MOTORON           ;PROVIDE MOTOR POWER
007614          LDA      #1                ;SAY WE'VE SELECTED
007615          STA      DRIVESEL,Y        ; THIS DRIVE
007616          *
007617          * IF WE HAVE MOTORTIME TO BURN,
007618          * THEN DELAY 50 MS. THIS ENSURES
007619          * A GOOD SOLID CHKDRV AFTER
007620          * TURNING ON THE MOTOR.
007621          *
007622          LDA      MONTIMEH           ;ANY MOTORTIME?
007623          BPL      UNITRTS            ;=>NO, WE GO FOR IT.
007624          LDY      #5                ;5*10 MS
007625 UNITDEL     EQU      *
007626          LDA      #100               ;100*100US IS 10MS
007627          JSR      MSWAIT
007628          DEY
007629          BNE      UNITDEL
007630          LDA      #2                ;INCLUDE THE 50MS
007631          JSR      ADDTIME            ; IN MOTOR UPTIME(S)
007632 UNITRTS     EQU      *
007633          RTS
007634          SKP      5
007635          REP      40

```

```
007636 * NAME      : EXTDESEL
007637 * FUNCTION: DESELECT ALL EXTERNAL DRIVE MOTORS
007638 * INPUT      : NONE
007639 * DESTROYS: AC,X
007640          REP          40
007641 *
007642 EXTDESEL      EQU          *
007643          LDA          MD.EXT1      ;DESELECT ALL EXTERNAL
007644          LDA          MD.EXT2      ; DRIVE MOTORS
007645          LDX          #3           ;SHOW THAT THEY ARE
007646          LDA          #0           ; ARE ALL DEAD DUCKS
007647 EDS1          STA          DRIVESEL,X
007648          STA          UPTIME,X      ;DRIVE MOTORS ARE OFF
007649          DEX
007650          BNE          EDS1
007651          RTS
007652
007653          CHN          DISK3.SUBS.SRC
007654
007655 *****
007656 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.USEL.SRC
007657 *****
007658
007659
```

```

007660 =====
007661 DOCUMENT :SOS1.3.2of5.TWO:SOS.D3WRT.TEXT
007662 =====
007663
007664 *****
007665 * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.WRT.SRC
007666 *****
007667 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
007668
007669             PAGE
007670             REP          40
007671 * --- WRITE ---
007672             REP          40
007673 *
007674 WRITEREQ    EQU          *
007675             JSR          BLK2SECT          ;COMPUTE TRK/SECTOR THIS BLOCK
007676             LDA          E.REG            ;SET 2 MHZ
007677             AND          #$7F
007678             STA          E.REG
007679             JSR          PRENIB           ;PRENIBBLIZE FOR WRITE
007680             JSR          SECTORIO        ;WRITE IT OUT...
007681             BCS          WRITERR        ;=>SOMETHING'S WRONG
007682 *
007683             INC          SECTOR          ;BUMP TO NEXT
007684             INC          SECTOR          ; LOGICAL SECTOR
007685             INC          BUF+1           ;BUMP SECTOR BUFFER ADDRESS
007686             LDA          E.REG            ;SET 2 MHZ
007687             AND          #$7F
007688             STA          E.REG
007689             JSR          PRENIB           ;PRENIBBLIZE FOR WRITE
007690             JSR          SECTORIO        ;WRITE IT OUT
007691             BCS          WRITERR        ;=>SOMETHING'S WRONG
007692 *
007693 * MORE BYTES TO DO?
007694 *
007695             JSR          MOREBLKS         ;SETUP FOR NEXT
007696             BNE          WRITEREQ        ;=>MORE TO DO
007697             LDA          #0              ;GOOD RETURN
007698             JMP          EXIT
007699 *
007700 WRITERR     EQU          *
007701             JMP          EXIT            ;RETURN ERROR CODE
007702             PAGE
007703             REP          40
007704 * --- STATUS ---
007705             REP          40
007706 *
007707 STATUS     EQU          *
007708             LDX          #$60           ;DUMMY SLOT

```

```

007709          LDA      Q6H,X          ;SENSE WRITE PROTECT
007710          LDA      Q7L,X
007711          ASL      A              ;PRESERVE IT IN CARRY
007712          LDA      Q6L,X          ;BACK TO READ MODE
007713          LDA      #0             ;NOW MOVE BIT TO
007714          ROL      A              ; PROPER POSITION
007715          ROL      A              ; ($02)
007716          LDY      #0
007717          STA      (D.STATBUF),Y   ;RETURN IT
007718          LDA      #0             ;GOOD RETURN
007719          JMP      EXIT           ;DONE
007720          PAGE
007721          REP      40
007722          * --- INIT ---
007723          REP      40
007724          *
007725          INIT      EQU      *
007726          LDA      INITFLAG        ;INIT'ED YET?
007727          BMI      GOODINIT       ;=>YES, DONE
007728          *
007729          LDA      #$60            ;SETUP SLOT FOR
007730          STA      IBSLOT          ; CORE ROUTINES
007731          LDA      #$FF            ;PREVENT SECOND
007732          STA      INITFLAG        ; INIT
007733          LDA      #0              ;CLEAR STUFF OUT
007734          STA      PREVUNIT       ;SOSBOOT JUST USED .D1
007735          LDY      #4
007736          CLRDRVS  EQU      *
007737          LDA      #0
007738          STA      DRIVESL-1,Y      ;NOBODY SELECTED
007739          STA      UPTIME-1,Y      ;ALL OFF
007740          STA      DRVTRACK-1,Y
007741          DEY
007742          BNE      CLRDRVS
007743          DO      1-TEST           ;ONLY IF NOT TESTING
007744          *
007745          * SET UP .D1 SINCE LOADER'S USING IT:
007746          *
007747          LDA      E.REG            ;SET 1MHZ FOR THE
007748          ORA      #$80            ; STATEMACHINE I/O
007749          STA      E.REG
007750          JSR      CHKDRV          ;IS .D1 MOTOR SPINNING?
007751          BEQ      INIT2          ;=>NO, MOTOR'S OFF
007752          LDA      #T200MS        ;UPTIME GOOD FOR READS
007753          STA      UPTIME+0
007754          INIT2      EQU      *
007755          LDA      #1
007756          STA      DRIVESL+0       ;.D1 IS THE CURRENT DRIVE
007757          LDA      $0300+CURTRK    ;RETRIEVE CURRENT TRACK
007758          STA      DRVTRACK+0     ;REMEMBER IT

```



```

007759             FIN
007760 *
007761 * SET UP JMP TABLE FOR CORRECT ROM:
007762 *
007763             DO             REV0ROM             ;ONLY IF SUPPORTING IT!
007764             LDA             $F1B9             ;LOOK FOR START OF RDADR
007765             CMP             #$A0             ;IS IT RDADR (REV1)?
007766             BEQ             INITREV1         ;=>YES
007767             CMP             #$60             ;IS IT END OF READ (REV0)?
007768             BNE             INITERR         ;=>NEITHER!
007769             LDY             #0               ;REV=0
007770             BEQ             INITVECT         ;(ALWAYS TAKEN)
007771 INITREV1     EQU             *
007772             LDY             #VSIZE
007773 INITVECT     EQU             *
007774             STY             ROMREV             ;SET ROM REVISION INDICATOR
007775             LDX             #VSIZE
007776 MOVEVECT    EQU             *
007777             LDA             REV0,Y           ;GET A BYTE
007778             STA             JMPTAB,Y         ;MOVE IT
007779             INY
007780             DEX
007781             BNE             MOVEVECT
007782             FIN
007783 GOODINIT     EQU             *
007784             LDA             #0               ;RETCODE=GOOD, IF YOU CARE
007785             CLC
007786             BCC             EXIT             ;(ALWAYS TAKEN)
007787             DO             REV0ROM
007788 INITERR      EQU             *
007789             SEC
007790 * FALL THRU TO EXIT
007791             FIN
007792             PAGE
007793             REP             40
007794 * -- EXIT PATH --
007795             REP             40
007796 *
007797 EXIT         EQU             *
007798             PHA
007799 *
007800 * UPDATE UPTIME BY 50 MS (3 SECTOR-TIMES)
007801 * TO ACCOUNT FOR READ/WRITE TIME:
007802 *
007803             LDA             D.COMMAND         ;GET COMMAND
007804             CMP             #2               ;SENSE OR INIT?
007805             BCS             EXIT2           ;=>YES, NO TIME USED UP
007806             LDA             #2               ;TIME=50 MS (2 UNITS)
007807             JSR             ADDTIME          ;BUMP UPTIME(S)
007808 *

```

```
007809 * RESTORE CALLER ENVIRONMENT:
007810 *
007811 EXIT2      EQU      *
007812          LDA      E.REG      ;GET CURRENT STATE
007813          AND      #$20      ; OF THE SCREEN
007814          ORA      ESAVE      ;MERGE WITH CALLER STATE
007815          STA      E.REG
007816          JSR      FIXIRQ      ;RE-ENABLE IRQ IF OK
007817          LDA      MOTOROFF    ;START MOTOR-OFF TIMEOUT
007818          PLA
007819          DO      TEST      ;IF TEST, NO SYSERR
007820          RTS
007821          ELSE
007822          BNE      GOERR      ;=>ERROR RETURN VIA SYSERR
007823          CLC
007824          RTS      ;GOOD RETURN W/CARRY CLEAR
007825 GOERR      EQU      *
007826          JSR      SYSERR      ;RETURN VIA SYSERR
007827          FIN
007828
007829          CHN      DISK3.SIO.SRC
007830
007831 *****
007832 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.WRT.SRC
007833 *****
007834
```

```

007835 =====
007836 DOCUMENT :SOS1.3.2of5.TWO:SOS.DEVMGR.TEXT
007837 =====
007838
007839 *****
007840 * APPLE /// SOS 1.3 SOURCE CODE FILE: DEVMGR.SRC
007841 *****
007842 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
007843
007844             SBTL             "SOS 1.1 DEVICE MANAGER"
007845             REL
007846             INCLUDE         SOSORG,6,1,254
007847             ORG             ORGDMGR
007848 ZZORG             EQU             *
007849             MSB             OFF
007850             REP             100
007851 *             COPYRIGHT (C) APPLE COMPUTER INC. 1980
007852 *             ALL RIGHTS RESERVED
007853             REP             100
007854 *
007855 * DEVICE MANAGER (VERSION = 1.10 )
007856 *             (DATE       = 8/04/81)
007857 *
007858 * THIS MODULE IS RESPONSIBLE FOR CALLING THE CORRECT DEVICE
007859 * DRIVER WHEN A D.READ...D.INIT SYSTEM CALL IS MADE.
007860 * (NOTE:  D.OPEN,D.CLOSE AND D.INIT ARE ONLY CALLABLE FROM
007861 * INSIDE THE OPERATING SYSTEM).  D.INFO AND GET.DNUM CALLS
007862 * ARE HANDLED INSIDE THIS MODULE. REPEAT.IO BYPASSES THIS MODULE.
007863             REP             100
007864 *
007865             ENTRY          DMGR
007866 *
007867             ENTRY          MAX.DNUM
007868             ENTRY          SDT.SIZE
007869             ENTRY          SDT.DIBL
007870             ENTRY          SDT.DIBH
007871             ENTRY          SDT.ADRL
007872             ENTRY          SDT.ADRH
007873             ENTRY          SDT.BANK
007874             ENTRY          SDT.UNIT
007875             ENTRY          BLKD.SIZE
007876             ENTRY          BLKDLST
007877 *
007878             EXTRN          SYSERR
007879             EXTRN          SERR
007880             EXTRN          NODNAME
007881             EXTRN          BADDNUM
007882             EXTRN          SYSDEATH
007883             EXTRN          BADSYSCALL

```

```

007884 *
007885         EXTRN      SXPAGE
007886 *
007887 E.REG      EQU      $FFDF      ; ENVIRONMENT REGISTER
007888 B.REG      EQU      $FFEF      ; BANK REGISTER
007889         PAGE
007890         REP      100
007891 *
007892 * SYSTEM DEVICE TABLE (SDT)
007893 *
007894 * CONTAINS THE ADDRESS OF EACH DRIVER'S DIB (SDT.DIB), THE
007895 * ADDRESS OF EACH DRIVER'S ENTRY POINT (SDT.ADR), AND THE
007896 * UNIT # OF EACH DRIVER (SDT.UNIT). THE TABLE IS INDEXED
007897 * BY DEVICE NUMBER. ENTRY 0 IS RESERVED FOR FUTURE USE.
007898 *
007899         REP      100
007900 *
007901 SDT.SIZE   EQU      25
007902 *
007903 MAX.DNUM   DS      1              ;MAX DEV NUMBER IN SYSTEM+1
007904 SDT.DIBL  DS      SDT.SIZE      ;ADR OF DEVICE INFORMATION BLOCK
007905 SDT.DIBH  DS      SDT.SIZE
007906 *
007907 SDT.ADR   DS      SDT.SIZE      ;ADR OF ENTRY POINT
007908 SDT.ADRH  DS      SDT.SIZE
007909 *
007910 SDT.BANK  DS      SDT.SIZE      ;BANK # OF DEVICE
007911 *
007912 SDT.UNIT  DS      SDT.SIZE      ;UNIT # OF DRIVER
007913 *
007914         REP      100
007915 * BLOCK DEVICE LIST TABLE
007916 *
007917 BLKD.SIZE EQU      13
007918 BLKDLST  DFB      $00
007919         DS      BLKD.SIZE-1
007920         PAGE
007921         REP      100
007922 *
007923 * DATA DECLARATIONS
007924 *
007925         REP      100
007926 *
007927 D.TPARMX EQU      $C0
007928 REQCODE  EQU      D.TPARMX
007929 *
007930 * D.READ/WRITE/CTRL/STATUS/OPEN/CLOSE/INIT/REPEAT PARMS
007931 *
007932 DNUM     EQU      D.TPARMX+1
007933 *

```

```

007934 * D.INFO PARS
007935 *
007936 I.DNUM      EQU      D.TPARAMX+1
007937 I.DNAME     EQU      D.TPARAMX+2
007938 I.DLIST     EQU      D.TPARAMX+4
007939 I.LENGTH    EQU      D.TPARAMX+6
007940 *
007941 * GET.DEV.NUM PARS
007942 *
007943 G.DNAME     EQU      D.TPARAMX+1
007944 G.DNUM      EQU      D.TPARAMX+3
007945 *
007946 * SDT ENTRY (=DIB) FIELDS
007947 *
007948 DIB.SLOT    EQU      $11          ;DIB'S DEVICE SLOT FIELD
007949 DIB.DTYPE  EQU      $13          ;DIB'S DEVICE TYPE FIELD
007950 *
007951 SDTP       EQU      D.TPARAMX+$10 ; PTR TO CURRENT SDT ENTRY
007952          PAGE
007953          REP      100
007954 *
007955 * DEVICE MANAGER (MAIN ENTRY POINT)
007956 *
007957          REP      100
007958 DMGR       EQU      *
007959 *
007960          LDA      REQCODE
007961          CMP      #4
007962          BCC      DRIVER          ; D.READ/WRITE/CTRL/STATUS CALL
007963          BNE      DM000
007964          JMP      GET.DNUM        ; GET.DEV.NUM CALL
007965 DM000     CMP      #5
007966          BEQ      D.INFO          ; D.INFO CALL
007967          CMP      #$A
007968          BCC      DRIVER          ; D.OPEN/CLOSE/INIT
007969          LDA      #BADSYSCALL    ; ELSE FATAL ERROR
007970          JSR      SYSDEATH      ; EXIT
007971          PAGE
007972          REP      100
007973 * D.READ/WRITE/CTRL/STATUS/OPEN/CLOSE/INIT CALLS
007974 * "JSR" TO DEVICE DRIVER
007975          REP      100
007976 DRIVER     EQU      *
007977 *
007978          LDX      DNUM            ; GET DNUM SYSCALL PARM
007979          BEQ      DM005          ; WITHIN BOUNDS?
007980          CPX      MAX.DNUM      ;
007981          BCC      DM010
007982 *
007983 * DNUM TOO LARGE

```

```

007984 *
007985 DM005      LDA      #>BADDNUM      ; INVALID DEVICE NUMBER
007986          JSR      SYSERR          ;   ERROR EXIT
007987 *
007988 * MAP DEV# TO UNIT#
007989 *
007990 DM010      LDA      SDT.UNIT,X
007991          STA      DNUM
007992 *
007993 * "JSR" TO DEVICE DRIVER VIA JMP TABLE
007994 *
007995          LDA      B.REG              ; STACK B.REG
007996          PHA
007997          LDA      #<DM.RTN-1        ; STACK RETURN ADDRESS
007998          PHA
007999          LDA      #>DM.RTN-1
008000          PHA
008001 *
008002          LDA      SDT.BANK,X        ; SELECT RAM BANK
008003          STA      B.REG
008004          LDA      SDT.ADRH,X       ; STACK DRIVER ENTRY POINT ADDRESS
008005          PHA
008006          LDA      SDT.ADRL,X
008007          PHA
008008 *
008009          LDA      E.REG              ; SWITCH IN I/O BANK
008010          ORA      #$40
008011          STA      E.REG
008012          RTS                      ; AND, "JSR" TO DEVICE DRIVER
008013 *
008014 DM.RTN    LDA      E.REG              ; SWITCH OUT I/O BANK
008015          AND      #$BF
008016          STA      E.REG
008017          PLA                      ; RESTORE B.REG
008018          STA      B.REG
008019          SEC
008020          LDA      SERR              ; RETRIEVE ERROR CODE
008021          BNE      DM017            ; ENSURE CARRY CLEARED IF NO ERROR
008022          CLC
008023 DM017    RTS                      ; AND, EXIT TO CALLER
008024          PAGE
008025          REP      100
008026 * D.INFO(IN.DNUM, OUT.DNAME, OUT.DEVLIST, IN.LENGTH) SYSTEM CALL
008027          REP      100
008028 D.INFO     EQU      *
008029 *
008030          LDX      I.DNUM              ; GET DNUM PARM
008031          BEQ      DM020            ; WITHIN BOUNDS?
008032          CPX      MAX.DNUM          ;
008033          BCC      DM030

```

```

008034 DM020          LDA      #>BADDNUM          ; NO, DNUM TOO LARGE
008035                JSR      SYSERR            ; ERROR EXIT
008036 *
008037 * MOVE PARMs FM SDT ENTRY (DEV INFO BLOCK) TO CALLER'S
008038 * PARM LIST
008039 *
008040 DM030          JSR      SETUP.SDT          ; SET UP ZPAGE PTR TO SDT ENTRY
008041 *
008042 * OUPUT DNAME PARM
008043 *
008044                LDA      (SDTP),Y          ; LOAD PARM'S BYTE COUNT
008045                TAY
008046 DM040          LDA      (SDTP),Y
008047                STA      (I.DNAME),Y
008048                DEY
008049                BPL      DM040
008050 *
008051 * OUTPUT DEVINFO PARM (SLOT,UNIT,DEVID,PRODCODE)
008052 *
008053                LDA      #DIB.SLOT
008054                CLC                          ; ADVANCE SDTP TO 2ND PARM IN SDT
008055                ADC      SDTP
008056                STA      SDTP
008057                BCC      DM045
008058                INC      SDTP+1
008059 DM045          LDY      I.LENGTH          ; LOAD BYTE COUNT
008060                BEQ      DM.EXIT            ; IF 0 THEN DONE
008061                DEY
008062                CPY      #$B
008063                BCC      DM050
008064                LDY      #$A
008065 DM050          LDA      (SDTP),Y
008066                STA      (I.DLIST),Y
008067                DEY
008068                BPL      DM050
008069 *
008070 DM.EXIT          CLC
008071                RTS                          ; NORMAL EXIT
008072                PAGE
008073                REP      100
008074 * GET.DEV.NUM(IN.DNAME; OUT.DNUM) SYSTEM CALL
008075                REP      100
008076 *
008077 GET.DNUM          EQU      *
008078 *
008079                LDX      #1                ; SETUP PTR TO 1ST SDT ENTRY
008080 *
008081 DM070          JSR      SETUP.SDT          ; SET UP ZPAGE PTR TO SDT ENTRY
008082 *
008083                LDA      (SDTP),Y          ; COMPARE DNAME LENGTHS

```

```

008084          CMP      (G.DNAME),Y
008085          BNE      NXTSDT
008086  *
008087          TAY                      ; LENGTHS MATCH, NOW COMPARE CHARS
008088  DM080      LDA      (G.DNAME),Y
008089          CMP      #$60
008090          BCC      DM090
008091          AND      #$DF                      ; UPSHIFT
008092  DM090      CMP      (SDTP),Y
008093          BNE      NXTSDT
008094          DEY
008095          BNE      DM080
008096  *
008097          TXA                      ; CHARS MATCH
008098          LDY      #0
008099          STA      (G.DNUM),Y          ; OUTPUT DEV NUM PARM
008100          LDY      #DIB.DTYPE        ; SET "N" FLAG IN STATUS REG.
008101          LDA      (SDTP),Y          ; N=1(BLOCK DEVICE) N=0(Char DEVICE)
008102          CLC
008103          RTS                      ; NORMAL EXIT
008104  *
008105  NXTSDT      INX                      ; LAST SDT ENTRY?
008106          CPX      MAX.DNUM
008107          BCC      DM070
008108  *
008109          LDA      #>NODNAME          ; ERROR, DNAME NOT FOUND IN SDT
008110          JSR      SYSERR            ; RETURN TO CALLER
008111          PAGE
008112          REP      100
008113  * SETUP.SDT(IN.X=DNUM, OUT.SDTP, B.REG, Y=0) X="UNCHANGED"
008114          REP      100
008115  SETUP.SDT    EQU      *
008116          LDA      SDT.DIBL,X          ; SET UP ZPAGE PTR TO SDT ENTRY
008117          STA      SDTP                ; (POINTS TO DNAME FIELD)
008118          LDA      SDT.DIBH,X
008119          STA      SDTP+1
008120          LDA      SDT.BANK,X
008121          STA      B.REG
008122          LDY      #0
008123          STY      SXPAGE+SDTP+1
008124          RTS
008125  *
008126          LST      ON
008127  ZZEND      EQU      *
008128  ZZLEN      EQU      ZZEND-ZZORG
008129          IFNE    ZZLEN-LENDMGR
008130          FAIL    2,"SOSORG          FILE IS INCORRECT FOR DEVPMGR"
008131          FIN
008132
008133  *****

```



008134 \* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DEVMGR.SRC  
008135 \*\*\*\*\*  
008136  
008137

```

008138 =====
008139 DOCUMENT :SOS1.3.2of5.TWO:SOS.DISK3.TEXT
008140 =====
008141
008142 *****
008143 * APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.DATA.SRC
008144 *****
008145 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
008146
008147             PAGE
008148 * GENERAL DATA:
008149 *
008150 PREVUNIT      DS          1          ;PRIOR UNIT ACCESSED (FOR REPEAT)
008151 PREVCMD       DS          1          ;PRIOR CMD (FOR REPEAT)
008152 *
008153 ESAVE         DS          1          ;SAVED E.REG
008154 VBLSAVE      DS          1          ;SAVED E.IER
008155 INITFLAG     DFB         0          ;<0 IS INITED
008156             DO          REVOROM
008157 ROMREV       DS          1          ;0=REV0, <>0=REV1
008158             FIN
008159 *
008160 * MOTOR-UP TIMES PER COMMAND
008161 T50MS        EQU         $02        ; 50MS FOR MONTIMEH
008162 T200MS       EQU         $08        ;200 MS FOR MONTIMEH
008163 T1SEC        EQU         $27        ;1-SEC FOR MONTIMEH
008164 *
008165 MTIMEB      DFB         T200MS,T1SEC,T50MS ;READ,WRITE,SENSE
008166 *
008167             REP         40
008168 * DRIVE TABLES:
008169 *
008170 DRIVESSEL    DS          4          ;NONZERO IF SELECTED
008171 *
008172 UPTIME       DS          4          ;MOTOR RUNTIME SINCE STARTED
008173 DRVTRACK     DS          4          ;CURRENT HEAD POSITION
008174             PAGE
008175             DO          REVOROM     ;ONLY IF SUPPORTING IT!
008176 * JUMP TABLE TO MONITOR ROUTINES.
008177 * THIS TABLE FILLED IN BY 'INIT'.
008178 *
008179 JMPTAB       EQU         *
008180 RDADR        JMP         *
008181 READ         JMP         *
008182 WRITE       JMP         *
008183 SEEK        JMP         *
008184 MSWAIT      JMP         *
008185 PRENIB      JMP         *
008186 POSTNIB     JMP         *

```

```

008187 *
008188 REV0      EQU      *           ;REV0 ADDRESSES
008189          JMP      $F1BD        ;RDADR
008190          JMP      $F148        ;READ
008191          JMP      $F219        ;WRITE
008192          JMP      $F400        ;SEEK
008193          JMP      $F456        ;MSWAIT
008194          JMP      $F2C6        ;PRENIB
008195          JMP      $F311        ;POSTNIB
008196 VSIZE    EQU      *-REV0     ;TABLE SIZE
008197 *
008198 REV1      EQU      *           ;REV1 ADDRESSES
008199          JMP      $F1B9        ;RDADR
008200          JMP      $F148        ;READ
008201          JMP      $F216        ;WRITE
008202          JMP      $F400        ;SEEK
008203          JMP      $F456        ;MSWAIT
008204          JMP      $F2C4        ;PRENIB
008205          JMP      $F30F        ;POSTNIB
008206          ELSE                ;FOR REV1 WE USE EQUATES
008207 RDADR    EQU      $F1B9        ;RDADR
008208 READ     EQU      $F148        ;READ
008209 WRITE    EQU      $F216        ;WRITE
008210 SEEK     EQU      $F400        ;SEEK
008211 MSWAIT   EQU      $F456        ;MSWAIT
008212 PRENIB   EQU      $F2C4        ;PRENIB
008213 POSTNIB  EQU      $F30F        ;POSTNIB
008214          FIN
008215
008216 ZZEND    EQU      *
008217 ZZLEN    EQU      *-ZZORG
008218          IFNE      ZZLEN-LENDISK3
008219          FAIL      2, "SOSORG      FILE IS INCORRECT FOR DISK3"
008220          FIN
008221
008222 *****
008223 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DISK3.DATA.SRC
008224 *****
008225
008226

```

```

008227 =====
008228 DOCUMENT :SOS1.3.2of5.TWO:SOS.FMGR.TEXT
008229 =====
008230
008231 *****
008232 * APPLE /// SOS 1.3 SOURCE CODE FILE: FMGR.SRC
008233 *****
008234 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
008235
008236             SBTL             "SOS 1.1  FILE MANAGER"
008237             REL
008238             INCLUDE          SOSORG,6,1,254
008239             ORG              ORGFMGR
008240 ZZORG        EQU            *
008241             MSB              OFF
008242             REP              60
008243 *             COPYRIGHT (C) APPLE COMPUTER INC. 1980
008244 *             ALL RIGHTS RESERVED
008245             REP              60
008246 *
008247 * FILE MANAGER (VERSION = 1.10  )
008248 *             (DATE          = 8/04/81)
008249 *
008250 * THIS MODULE IS ENTERED FROM THE SYSTEM CALL MANAGER, AND
008251 * IS RESPONSIBLE FOR SWITCHING TO EITHER THE BLOCK FILE
008252 * MANAGER, OR THE CHARACTER FILE MANAGER.
008253 *
008254             REP              60
008255 *
008256             ENTRY           FMGR
008257             ENTRY           LEVEL
008258 *
008259             EXTRN           BFMGR
008260             EXTRN           CFMGR
008261             EXTRN           SYSERR
008262             EXTRN           SERR
008263             EXTRN           BADPATH
008264             EXTRN           FNFERR
008265             EXTRN           LVLERR
008266 *
008267 F.TPARMX    EQU            $A0             ; LOC OF FILE SYSTEM CALL PARMS
008268 OPEN       EQU            $8
008269 CLOSE      EQU            $C
008270 SETTLEVEL  EQU            $12
008271 GETLEVEL   EQU            $13
008272 F.REQCODE  EQU            F.TPARMX
008273 F.LEVEL    EQU            F.TPARMX+$1
008274 PATHNAME   EQU            F.TPARMX+$1
008275 REFNUM     EQU            F.TPARMX+$1

```

```

008276 PERIOD      EQU      $2E
008277 LEVEL      DFB      $1
008278                PAGE
008279                REP      60
008280 *
008281 * FILE MANAGER
008282 *
008283                REP      60
008284 FMGR        EQU      *
008285 *
008286                LDA      F.REQCODE
008287                CMP      #OPEN
008288                BCC      FMGR010
008289                BEQ      FMGR020
008290                CMP      #CLOSE
008291                BCC      FMGR030
008292                BEQ      FMGR040
008293                CMP      #SETLEVEL
008294                BEQ      SLEVEL
008295                CMP      #GETLEVEL
008296                BEQ      GLEVEL
008297 *
008298 FMGR010       JMP      BFMGR          ; EXIT
008299 *
008300 FMGR020       LDY      #1
008301                LDA      (PATHNAME),Y
008302                CMP      #PERIOD
008303                BNE      FMGR010
008304                JSR      CFMGR
008305                BCC      FMGR024
008306                LDA      SERR
008307                CMP      #FNFERR
008308                BEQ      FMGR026
008309 FMGR024       RTS                    ; EXIT
008310 *
008311 FMGR026       LDA      #0
008312                STA      SERR
008313                JMP      BFMGR          ; EXIT
008314 *
008315 FMGR030       LDA      REFNUM
008316 FMGR031       BPL      FMGR010
008317                JMP      CFMGR          ; EXIT
008318 *
008319 FMGR040       LDA      REFNUM
008320                BNE      FMGR031
008321                JSR      BFMGR          ; CLOSE (0)
008322                JMP      CFMGR          ; EXIT
008323 *
008324 SLEVEL        LDA      F.LEVEL
008325                BEQ      LVL.ERR

```

```

008326          CMP          #4
008327          BCS          LVL.ERR
008328          STA          LEVEL
008329          RTS
008330 LVL.ERR      LDA          #LVLEERR
008331          JSR          SYSERR
008332          *
008333 GLEVEL      LDY          #0
008334          LDA          LEVEL
008335          STA          (F.LEVEL),Y
008336          RTS
008337          *
008338          LST          ON
008339 ZZEND      EQU          *
008340 ZZLEN      EQU          ZZEND-ZZORG
008341          IFNE          ZZLEN-LENFMGR
008342          FAIL          2,"SOSORG          FILE IS INCORRECT FOR FMGR"
008343          FIN
008344
008345 *****
008346 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: FMGR.SRC
008347 *****
008348
008349
008350

```

```

008351 =====
008352 DOCUMENT :SOS1.3.2of5.TWO:SOS.MMGR.A.TEXT
008353 =====
008354
008355 *****
008356 * APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.A.SRC
008357 *****
008358 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
008359
008360          SBTL          "SOS 1.1  MEMORY MANAGER"
008361          REL
008362          INCLUDE      SOSORG,6,1,254
008363          ORG          ORGMEMMG
008364 ZZORG          EQU          *
008365          MSB          OFF
008366          REP          60
008367 *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
008368 *          ALL RIGHTS RESERVED
008369          REP          60
008370 *
008371 * MEMORY MANAGER (VERSION = 1.10  )
008372 *          (DATE      = 8/04/81)
008373 *
008374 * THIS MODULE CONTAINS ALL OF THE MEMORY MANAGEMENT SYSTEM
008375 * CALLS SUPPORTED BY THE SARA OPERATING SYSTEM.  IT IS
008376 * ALSO CALLED BY THE BUFFER MANAGER.
008377 *
008378          REP          60
008379 *
008380          ENTRY        MMGR
008381 *
008382          ENTRY        ST.CNT
008383          ENTRY        ST.ENTRY
008384          ENTRY        ST.FREE
008385          ENTRY        ST.FLINK
008386          ENTRY        VRT.LIM
008387 *
008388          EXTRN        SYSERR
008389          EXTRN        BADSCNUM
008390          EXTRN        BADBKPG
008391          EXTRN        SEGRQDN
008392          EXTRN        SEGTBLFULL
008393          EXTRN        BADSEGNUM
008394          EXTRN        SEGNOTFND
008395          EXTRN        BADSRCHMODE
008396          EXTRN        BADCHGMODE
008397          EXTRN        BADPGCNT
008398          PAGE
008399          REP          60

```

```

008400 *
008401 * SEGMENT TABLE
008402 * (NOTE: ENTRY 0 IS NOT USED)
008403 *
008404             REP             60
008405 *
008406 ST.FREE      DS             1             ; PTR TO FIRST FREE SEG TABLE ENTRY
008407 ST.ENTRY    DS             1             ; PTR TO HIGHEST ALLOC SEG TABLE ENTRY
008408 ST.SIZ       EQU             7
008409 ST.CNT      EQU            32
008410 ST.TBL      DS             ST.SIZ*ST.CNT
008411 ST.BLINK    EQU            ST.TBL        ; BACK LINK TO PREV ALLOC SEG ENTRY
008412 ST.FLINK    EQU            ST.BLINK+ST.CNT ; FORWARD LINK      "
008413 ST.BASEL    EQU            ST.FLINK+ST.CNT ; BASE BANK/PAGE
008414 ST.BASEH    EQU            ST.BASEL+ST.CNT
008415 ST.LIML     EQU            ST.BASEH+ST.CNT ; LIMIT BANK/PAGE
008416 ST.LIMH     EQU            ST.LIML+ST.CNT
008417 ST.ID       EQU            ST.LIMH+ST.CNT ; SEG ID
008418             PAGE
008419             REP             60
008420 *
008421 * DATA DECLARATIONS
008422 *
008423             REP             60
008424 *
008425 ZPAGE       EQU            $40           ; BEGINNING OF ZPAGE TEMP SPACE FOR MEMORY MANAGER
008426 VRT.BASE    EQU            $0           ; INTERNAL BK/PG PTR TO LOWEST VIRT PAGE
008427 VRT.LIM     EQU            ZPAGE+$0     ; &$1, INTERNAL BK/PG PTR TO HIGHEST VIRT PAGE
008428 PHY1BASE    EQU            $0780        ; BANK "F",PAGE "0"
008429 PHY1LIM    EQU            $079F        ; BANK "F",PAGE "1F"
008430 PHY2BASE    EQU            $0820        ; BANK "10",PAGE "A0"
008431 PHY2LIM    EQU            $087F        ; BANK "10",PAGE "FF"
008432 *
008433 * REQUEST.SEG DATA DECLARATIONS
008434 *
008435 M.TPARMX      EQU            $60           ; BEGINNING ADDRESS OF MMGR SOS CALL PARMS
008436 M.RQCODE     EQU            M.TPARMX
008437 RQ.BASE     EQU            M.TPARMX+1   ; BASE.BANK/PAGE
008438 RQ.LIM       EQU            M.TPARMX+3   ; LIMIT.BANK/PAGE
008439 RQ.ID       EQU            M.TPARMX+5
008440 RQ.NUM      EQU            M.TPARMX+6
008441 *
008442 RQ.REGION    EQU            ZPAGE+$2     ;VRT(0),PHY0(1),PHY1(2)
008443 *
008444 * FIND.SEG DATA DECLARATIONS
008445 *
008446 SRCHMODE     EQU            M.TPARMX+1   ; SEARCH MODE (0,1,2)
008447 F.ID        EQU            M.TPARMX+2   ; SEG ID
008448 F.PGCT       EQU            M.TPARMX+3   ; PAGE COUNT (LO
008449 FX.PGCT      EQU            ZPAGE+$3     ; &$4, INTERNAL PAGE COUNT

```



```

008450 F.BASE      EQU      M.TPARMX+5      ; BASE.BANK/PAGE
008451 F.LIM      EQU      M.TPARMX+7      ; LIMIT.BANK/PAGE
008452 F.NUM      EQU      M.TPARMX+9      ; SEG NUM
008453 F.ERR      EQU      ZPAGE+$5       ; ERROR FLAG
008454 TRUE       EQU      $80
008455 FALSE      EQU      $0
008456 CFS.PGCT   EQU      ZPAGE+$6       ; &$7, CURRENT FREE SEGMENT'S PAGE COUNT
008457 CFS.BASE   EQU      ZPAGE+$8       ; &$9, " BASE.BANK/PAGE
008458 CFS.LIM    EQU      ZPAGE+$A       ; &$B, " LIMIT.BANK/PAGE
008459 CFS.BLINK  EQU      ZPAGE+$C       ; " " BACK LINK
008460 CFS.BASE0   EQU      ZPAGE+$D       ; &$E, " BASE (SMODE=0)
008461 CFS.BASE1  EQU      ZPAGE+$F       ; &$10, " BASE (SMODE=1)
008462 CFS.NEXT   EQU      ZPAGE+$11      ; " " NEXT ENTRY
008463 CFS.PREV   EQU      ZPAGE+$12      ; " " PREV ENTRY
008464 CFS.PTR    EQU      ZPAGE+$13      ; &$14 " POINTER TO NXT FREE PG
008465 BFS.PGCT   EQU      ZPAGE+$15      ; &$16, BIGGEST FREE SEGMENT'S PAGE COUNT
008466 BFS.BASE   EQU      ZPAGE+$17      ; &$18 " BASE.BANK/PAGE
008467 BFS.LIM    EQU      ZPAGE+$19      ; &$1A " LIMIT.BANK/PAGE
008468 BFS.BLINK  EQU      ZPAGE+$1B      ; " " BACK LINK
008469 *
008470 * CHANGE.SEG DATA DECLARATIONS
008471 *
008472 CHG.NUM      EQU      M.TPARMX+1     ; SEGNUM PARM
008473 CHG.MODE    EQU      M.TPARMX+2     ; CHANGE MODE PARM
008474 CHG.PGCT    EQU      M.TPARMX+3     ; PAGE COUNT PARM
008475 CHG.PGCTX   EQU      ZPAGE+$1C     ; &$1D, INTERNAL STORE FOR PGCT
008476 CHG.NEW     EQU      ZPAGE+$1E     ; &$1F, BANK/PAGE OF SEG'S NEW LIMIT OR BASE
008477 *
008478 * GET.SEG.INFO DATA DECLARATIONS
008479 *
008480 GSI.NUM        EQU      M.TPARMX+1
008481 GSI.BASE    EQU      M.TPARMX+2
008482 GSI.LIM     EQU      M.TPARMX+4
008483 GSI.PGCT    EQU      M.TPARMX+6
008484 GSI.ID      EQU      M.TPARMX+8
008485 *
008486 * GET.SEG.NUM DATA DECLARATIONS
008487 *
008488 GSN.BKPG       EQU      M.TPARMX+1
008489 GSN.NUM      EQU      M.TPARMX+3
008490 *
008491 * RELEASE.SEG DATA DECLARATIONS
008492 *
008493 RLS.NUM       EQU      M.TPARMX+1     ; SEG NUM
008494 *
008495 * REGION - DATA DECLARATIONS
008496 *
008497 RGN.BKPG      DS        2             ; TEMP CONTAINER FOR BANK/PAGE
008498 PAGE
008499 REP         60

```

```

008500 *
008501 * MMGR
008502 *
008503 * THIS ROUTINE IS THE MAIN ENTRANCE TO THE MEMORY MANAGER
008504 * MODULE. IT FUNCTIONS AS A SWITCH, BASED UPON THE RECEIVED
008505 * REQUEST CODE, TO TRANSFER CONTROL TO THE ROUTINE THAT
008506 * HANDLES THE SPECIFIC SYSTEM CALL.
008507 *
008508             REP             60
008509 *
008510 MMGR             EQU             *
008511             LDA             M.RQCODE
008512             BEQ             MMGR010             ; "REQ.SEG"
008513             CMP             #1
008514             BEQ             MMGR020             ; "FIND.SEG"
008515             CMP             #2
008516             BEQ             MMGR030             ; "CHANGE.SEG"
008517             CMP             #3
008518             BEQ             MMGR040             ; "GET.SEG.INFO"
008519             CMP             #4
008520             BEQ             MMGR050             ; "GET.SEG.NUM"
008521             CMP             #5
008522             BEQ             MMGR060             ; "RELEASE.SEG"
008523 *
008524             LDA             #BADSCNUM
008525             JSR             SYSERR
008526 *
008527 MMGR010          JMP             REQ.SEG
008528 MMGR020          JMP             FIND.SEG
008529 MMGR030          JMP             CHG.SEG
008530 MMGR040          JMP             GET.SEG.INFO
008531 MMGR050          JMP             GET.SEG.NUM
008532 MMGR060          JMP             RELEASE.SEG
008533             PAGE
008534             REP             60
008535 *
008536 * REQUEST.SEG(IN.BASE.BANKPAGE,LIMIT.BANKPAGE,SEGID; OUT.SEGNUM)
008537 *
008538             REP             60
008539 *
008540 REQ.SEG           EQU             *
008541 *
008542 * CONVERT CALLER'S BASE.BANK/PAGE TO INTERNAL FMT
008543 *
008544             LDX             RQ.BASE
008545             LDY             RQ.BASE+1
008546             JSR             CNVRT.IBP
008547             BCC             RQ005
008548 *
008549 RQ.ERR           RTS                     ; ERR EXIT - INVALID BANK/PAGE

```

```

008550 *
008551 RQ005      STX      RQ.BASE
008552          STY      RQ.BASE+1
008553          STA      RQ.REGION
008554 *
008555 * CONVERT CALLER'S LIMIT.BANK/PAGE TO INTERNAL FMT
008556 *
008557          LDX      RQ.LIM
008558          LDY      RQ.LIM+1
008559          JSR      CNVRT.IBP
008560          BCS      RQ.ERR          ; ERR - INVALID BANK/PAGE
008561          STX      RQ.LIM
008562          STY      RQ.LIM+1
008563 *
008564 * IF BASE AND LIMIT ARE IN DIFFERENT REGIONS THEN ERR
008565 *
008566          CMP      RQ.REGION
008567          BNE      RQ.ERR1          ; ERR - INVALID BANK/PAGE PAIR
008568 * IF CALLER'S BASE > LIMIT THEN ERR
008569 *
008570          LDA      RQ.LIM
008571          CMP      RQ.BASE
008572          LDA      RQ.LIM+1
008573          SBC      RQ.BASE+1
008574          BCC      RQ.ERR1          ; ERR - INVALID BANK/PAGE PAIR
008575 *
008576 * PREV SEGNUM:=NULL; NEXT SEGNUM:=FIRST ENTRY
008577 *
008578          LDX      #0
008579          LDY      ST.ENTRY          ; NOTE: PREV/NEXT CARRIED IN X & Y REGISTERS
008580 *
008581 * IF NO SEGS IN SEG TABLE THEN ALLOCATE REQUESTED SEG
008582 *
008583          BEQ      RQ030
008584 *
008585 * IF FIRST SEG IN SEG TABLE BELOW REQUESTED SEG
008586 * THEN ALLOCATE SEG
008587 *
008588          LDA      ST.LIML,Y
008589          CMP      RQ.BASE
008590          LDA      ST.LIMH,Y
008591          SBC      RQ.BASE+1
008592          BCC      RQ030
008593 *
008594 * ADVANCE TO NEXT SEG ENTRY
008595 *
008596 RQ010      TYA
008597          TAX
008598          LDA      ST.FLINK,Y
008599          TAY

```

```

008600 *
008601 * IF THERE IS NO NEXT SEG ENTRY
008602 *   IF REQUESTED SEG IS BELOW PREV SEG
008603 *     THEN ALLOCATE REQ SEG
008604 *     ELSE ERR
008605 *
008606         BNE         RQ020
008607         LDA         RQ.LIM
008608         CMP         ST.BASEL,X
008609         LDA         RQ.LIM+1
008610         SBC         ST.BASEH,X
008611         BCC         RQ030
008612 *
008613         BCS         RQ.ERR2             ; ERR - SEGMENT REQUEST DENIED
008614 *
008615 * IF REQUESTED LIMIT >= PREV SEG'S BASE THEN ERR
008616 *
008617 RQ020         LDA         RQ.LIM
008618         CMP         ST.BASEL,X
008619         LDA         RQ.LIM+1
008620         SBC         ST.BASEH,X
008621         BCS         RQ.ERR2             ; ERR - SEGMENT REQUEST DENIED
008622 *
008623 * IF REQUESTED BASE > NEXT SEG'S LIMIT
008624 *   THEN ALLOCATE REQUESTED SEGMENT
008625 *
008626         LDA         ST.LIML,Y
008627         CMP         RQ.BASE
008628         LDA         ST.LIMH,Y
008629         SBC         RQ.BASE+1
008630         BCS         RQ010             ; NO, ADVANCE TO NEXT SEGMENT
008631 *
008632 RQ030         TXA
008633         JSR         GET.FREE
008634         BCS         RQ.ERR3             ; ERR - SEG TABLE FULL
008635 *
008636 * ENTER BASE,LIMIT AND ID IN NEW SEG ENTRY
008637 *
008638         TAX
008639         LDA         RQ.BASE
008640         STA         ST.BASEL,X
008641         LDA         RQ.BASE+1
008642         STA         ST.BASEH,X
008643 *
008644         LDA         RQ.LIM
008645         STA         ST.LIML,X
008646         LDA         RQ.LIM+1
008647         STA         ST.LIMH,X
008648 *
008649         LDA         RQ.ID

```

```

008650          STA          ST.ID,X
008651 *
008652 * RETURN NEW SEG NUM TO CALLER AND RETURN
008653 *
008654          LDY          #0
008655          TXA
008656          STA          (RQ.NUM),Y
008657 *
008658          CLC
008659          RTS                      ; NORMAL EXIT
008660 *
008661 RQ.ERR1      LDA          #BADBKPG
008662          JSR          SYSERR          ; ERR EXIT
008663 RQ.ERR2      LDA          #SEGRQDN
008664          JSR          SYSERR          ; ERR EXIT
008665 *
008666 RQ.ERR3      LDA          #SEGTBLFULL
008667          JSR          SYSERR          ; ERR EXIT
008668          PAGE
008669          REP          60
008670 *
008671 * FIND.SEG(IN.SRCHMODE,SEGID; INOUT.PAGECT;
008672 *          OUT.BASE.BKPG,LIMIT.BKPG,SEGNUM)
008673 *
008674          REP          60
008675 *
008676 FIND.SEG      EQU          *
008677 *
008678 * RETRIEVE PAGE COUNT PARAMETER AND CLEAR ERR FLAG
008679 *
008680          LDY          #0
008681          LDA          (F.PGCT),Y
008682          STA          FX.PGCT
008683          INY
008684          LDA          (F.PGCT),Y
008685          STA          FX.PGCT+1
008686 *
008687          BNE          FIND001
008688          LDA          FX.PGCT
008689          BNE          FIND001
008690          LDA          #BADPGCNT          ; ERR, PAGECT=0, EXIT
008691          JSR          SYSERR
008692 *
008693 FIND001      LDA          #FALSE
008694          STA          F.ERR
008695 *
008696 * IF SEARCH MODE>2 THEN ERR
008697 *
008698          LDA          SRCHMODE
008699          CMP          #3

```

```

008700          BCC      FIND005
008701          LDA      #BADSrchMODE
008702          JSR      SYSERR          ; ERR EXIT
008703 *
008704 * INITIALIZE NEXT FREE SEGMENT SUBROUTINE,
008705 * AND BIGGEST FREE SEGMENT PAGE COUNT
008706 *
008707 FIND005      JSR      NXTFRSEG.I
008708          LDA      #0
008709          STA      BFS.PGCT
008710          STA      BFS.PGCT+1
008711 *
008712 * GET NEXT FREE SEGMENT
008713 *
008714 FIND010      JSR      NXTFRSEG
008715          BCC      FIND015          ; PROCESS FREE SEGMENT
008716 *
008717 * NO MORE FREE SEGMENTS LEFT
008718 * RETURN BIGGEST FREE SEGMENT FOUND
008719 * ALONG WITH ERR
008720 *
008721          LDA      #TRUE
008722          STA      F.ERR
008723          LDX      #0          ; SEG#:=0
008724          JMP      FIND070
008725 *
008726 * FREE SEGMENT FOUND.
008727 *   IF FREE SEGMENT > BIGGEST FREE SEGMENT THEN BFS:=CFS
008728 *
008729 FIND015      LDA      BFS.PGCT
008730          CMP      CFS.PGCT
008731          LDA      BFS.PGCT+1
008732          SBC      CFS.PGCT+1
008733          BCS      FIND030
008734 *
008735          LDX      #6
008736 FIND020      LDA      CFS.PGCT,X
008737          STA      BFS.PGCT,X
008738          DEX
008739          BPL      FIND020
008740 *
008741 * IF BFS.PGCT<F.PGCT THEN GET NEXT FREE SEGMENT
008742 *
008743 FIND030      LDA      BFS.PGCT
008744          CMP      FX.PGCT
008745          LDA      BFS.PGCT+1
008746          SBC      FX.PGCT+1
008747          BCC      FIND010
008748 *
008749 * BFS.BASE:=BFS.LIM-FX.PGCT+1

```

```

008750 * BFS.PGCT:=FX.PGCT
008751 *
008752         LDA         BFS.LIM
008753         SBC         FX.PGCT
008754         STA         BFS.BASE
008755         LDA         BFS.LIM+1
008756         SBC         FX.PGCT+1
008757         STA         BFS.BASE+1
008758         INC         BFS.BASE
008759         BNE         FIND050
008760         INC         BFS.BASE+1
008761 *
008762 FIND050     LDA         FX.PGCT
008763         STA         BFS.PGCT
008764         LDA         FX.PGCT+1
008765         STA         BFS.PGCT+1
008766 *
008767 * DELINK ENTRY FROM FREE LIST, AND LINK
008768 * IT INTO SEGMENT LIST
008769 *
008770         LDA         BFS.BLINK
008771         JSR         GET.FREE
008772         BCC         FIND060
008773         RTS                    ; ERR - SEG TABLE FULL
008774 *
008775 * ST.ID(NEW) :=F.ID
008776 * ST.BASE(NEW) :=BFS.BASE
008777 * ST.LIM(NEW) :=BFS.LIM
008778 *
008779 FIND060     TAX
008780         LDA         F.ID
008781         STA         ST.ID,X
008782 *
008783         LDA         BFS.BASE
008784         STA         ST.BASEL,X
008785         LDA         BFS.BASE+1
008786         STA         ST.BASEH,X
008787 *
008788         LDA         BFS.LIM
008789         STA         ST.LIML,X
008790         LDA         BFS.LIM+1
008791         STA         ST.LIMH,X
008792 *
008793 * RETURN SEGNUM, PAGE COUNT, BASE BANK/PAGE, AND LIMIT BANK/PAGE
008794 * TO CALLER
008795 FIND070     LDY         #0
008796         TXA
008797         STA         (F.NUM),Y
008798 *
008799         LDA         BFS.PGCT

```

```

008800          STA          (F.PGCT),Y
008801          INY
008802          LDA          BFS.PGCT+1
008803          STA          (F.PGCT),Y
008804 *
008805          LDX          BFS.BASE
008806          LDY          BFS.BASE+1
008807          JSR          CNVRT.XBP
008808          TYA
008809          LDY          #1
008810          STA          (F.BASE),Y
008811          DEY
008812          TXA
008813          STA          (F.BASE),Y
008814 *
008815          LDX          BFS.LIM
008816          LDY          BFS.LIM+1
008817          JSR          CNVRT.XBP
008818          TYA
008819          LDY          #1
008820          STA          (F.LIM),Y
008821          DEY
008822          TXA
008823          STA          (F.LIM),Y
008824 *
008825          LDA          F.ERR          ; IF ERR FLAG TRUE THEN REPORT IT.
008826          BNE          FIND.ERR
008827 *
008828          CLC
008829          RTS          ; NORMAL EXIT
008830 *
008831 FIND.ERR    LDA          #SEGRQDN
008832          JSR          SYSERR        ; ERR EXIT
008833
008834          CHN          MEMMGR.B.SRC
008835
008836 *****
008837 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.A.SRC
008838 *****
008839
008840
008841

```



```

008842 =====
008843 DOCUMENT :SOS1.3.2of5.TWO:SOS.MMGR.B.TEXT
008844 =====
008845
008846 *****
008847 * APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.B.SRC
008848 *****
008849 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
008850
008851             PAGE
008852             REP             60
008853 *
008854 * NEXT FREE SEGMENT - INITIALIZATION
008855 *
008856 * INPUT:  SEGMENT TABLE
008857 * OUTPUT: CFS.PTR "1ST FREE BANK/PAGE IN VIRTUAL MEMORY
008858 *          CFS.PREV "PREVIOUS SEGMENT EXAMINED"
008859 *          CFS.NEXT "SEGMENT FOLLOWING CFS.PREV"
008860 * ERROR:  NONE (IF NO FREE BK/PG FOUND, THEN CFS.PTR="FFFF")
008861 *
008862             REP             60
008863 *
008864 NXTFRSEG.I      EQU          *
008865 *
008866 * CFS.PTR := VRT.LIM
008867 * CFS.PREV := 0
008868 * CFS.NEXT := ST.ENTRY
008869 *
008870             LDA             >VRT.LIM
008871             STA             CFS.PTR
008872             LDA             >VRT.LIM+1
008873             STA             CFS.PTR+1
008874 *
008875             LDA             #0
008876             STA             CFS.PREV
008877 *
008878             LDX             ST.ENTRY
008879             STX             CFS.NEXT
008880 *
008881 * L0:  IF CFS.NEXT=0 THEN DONE
008882 *
008883 FRSGI010        BEQ          FRSGI.EXIT
008884 *
008885 * IF ST.LIM(CFS.NEXT)<=VRT.LIM THEN GOTO L1
008886 *
008887             LDA             >VRT.LIM
008888             CMP             ST.LIML,X
008889             LDA             >VRT.LIM+1
008890             SBC             ST.LIMH,X

```

```

008891          BCS          FRSGI020
008892 *
008893 * CFS.PREV:=CFS.NEXT
008894 * CFS.NEXT:=ST.FLINK(CFS.NEXT)
008895 * GOTO L0
008896 *
008897          STX          CFS.PREV
008898          LDA          ST.FLINK,X
008899          TAX
008900          STX          CFS.NEXT
008901          JMP          FRSGI010
008902 *
008903 * L1:  IF ST.LIM(CFS.NEXT)<VRT.LIM THEN DONE
008904 *
008905 FRSGI020      LDA          ST.LIML,X
008906          CMP          >VRT.LIM
008907          LDA          ST.LIMH,X
008908          SBC          >VRT.LIM+1
008909          BCC          FRSGI.EXIT
008910 *
008911 *
008912          JSR          NXTFRPG
008913 *
008914 FRSGI.EXIT    RTS          ; NORMAL EXIT
008915          PAGE
008916          REP          60
008917 *
008918 * NEXT FREE SEGMENT
008919 *
008920 * INPUT:  SEG TABLE
008921 * OUTPUT: CFS.BLINK
008922 *          CFS.BASE
008923 *          CFS.LIMIT
008924 *          CFS.PGCT
008925 * OWN:   CFS.PREV
008926 *          CFS.NEXT
008927 *          CFS.PTR
008928 *
008929 * BUILDS A CANDIDATE FREE SEGMENT, WHOSE LIMIT BANK/PAGE =
008930 * THE CURRENT FREE PAGE (CFS.PTR).
008931 *
008932          REP          60
008933 *
008934 NXTFRSEG      EQU          *
008935 *
008936 * IF CFS.PTR="FFFF" THEN EXIT
008937 *
008938          LDA          CFS.PTR+1
008939          BPL          FRSG010
008940 *

```

```

008941          SEC
008942          RTS                ; EXIT - NO MORE FREE SEGMENTS LEFT
008943 *
008944 * CFS.BLINK:=CFS.PREV
008945 * CFS.LIM:=CFS.PTR
008946 *
008947 FRSG010      LDA      CFS.PREV
008948              STA      CFS.BLINK
008949 *
008950              LDA      CFS.PTR
008951              STA      CFS.LIM
008952              LDA      CFS.PTR+1
008953              STA      CFS.LIM+1
008954 *
008955 * IF CFS.NEXT=0 THEN CFS.BASE:=0
008956 *     ELSE CFS.BASE:=ST.LIM(CFS.NEXT)+1
008957 *
008958              LDA      CFS.NEXT
008959              BNE      FRSG020
008960              LDA      #0
008961              STA      CFS.BASE
008962              STA      CFS.BASE+1
008963              BEQ      FRSG030
008964 *
008965 FRSG020      LDX      CFS.NEXT
008966              CLC
008967              LDA      ST.LIML,X
008968              ADC      #1
008969              STA      CFS.BASE
008970              LDA      ST.LIMH,X
008971              ADC      #0
008972              STA      CFS.BASE+1
008973 *
008974 * CFS.BASE0:=CFS.LIM AND $FF80
008975 *
008976 FRSG030      LDY      CFS.LIM+1
008977              STY      CFS.BASE0+1
008978              LDA      CFS.LIM
008979              AND      #$80
008980              STA      CFS.BASE0
008981 *
008982 * CFS.BASE1:=CFS.BASE0-32K
008983 *
008984              SEC
008985              SBC      #$80
008986              STA      CFS.BASE1
008987              TYA
008988              SBC      #0
008989              STA      CFS.BASE1+1
008990              BCS      FRSG035

```

```

008991          LDA      #0
008992          STA      CFS.BASE1
008993          STA      CFS.BASE1+1
008994          *
008995          * IF CFS.BASE>=CFS.BASE0 THEN GOTO L1
008996          *
008997          FRSG035    LDA      CFS.BASE
008998                      CMP      CFS.BASE0
008999                      LDA      CFS.BASE+1
009000                      SBC      CFS.BASE0+1
009001                      BCS      FRSG050
009002          *
009003          * IF SEARCH MODE=0 THEN CFS.BASE:=CFS.BASE0
009004          * GOTO L1
009005          *
009006                      LDA      SRCHMODE
009007                      BNE      FRSG040
009008                      LDA      CFS.BASE0
009009                      STA      CFS.BASE
009010                      LDA      CFS.BASE0+1
009011                      STA      CFS.BASE+1
009012                      JMP      FRSG050
009013          *
009014          * IF CFS.BASE<CFS.BASE1 AND SEARCH MODE=1
009015          *     THEN CFS.BASE:=CFS.BASE1
009016          *
009017          FRSG040    LDA      CFS.BASE
009018                      CMP      CFS.BASE1
009019                      LDA      CFS.BASE+1
009020                      SBC      CFS.BASE1+1
009021                      BCS      FRSG050
009022          *
009023                      LDA      SRCHMODE
009024                      CMP      #1
009025                      BNE      FRSG050
009026          *
009027                      LDA      CFS.BASE1
009028                      STA      CFS.BASE
009029                      LDA      CFS.BASE1+1
009030                      STA      CFS.BASE+1
009031          *
009032          * L1:  CFS.PGCT:=CFS.LIM-CFS.BASE+1
009033          *
009034          FRSG050    SEC
009035                      LDA      CFS.LIM
009036                      SBC      CFS.BASE
009037                      STA      CFS.PGCT
009038                      LDA      CFS.LIM+1
009039                      SBC      CFS.BASE+1
009040                      STA      CFS.PGCT+1

```

```

009041          INC          CFS.PGCT
009042          BNE          FRSG052
009043          INC          CFS.PGCT+1
009044 *
009045 * ADVANCE FREE PAGE POINTER TO NEXT FREE PAGE
009046 *
009047 * IF SEARCH MODE<>1 THEN L2:
009048 *
009049 FRSG052      LDA          SRCHMODE
009050          CMP          #1
009051          BNE          FRSG060
009052 *
009053 * IF CFS.BASE < CFS.BASE0 THEN CFS.PTR:=CFS.BASE0-1
009054 *
009055          LDA          CFS.BASE
009056          CMP          CFS.BASE0
009057          LDA          CFS.BASE+1
009058          SBC          CFS.BASE0+1
009059          BCS          FRSG060
009060 *
009061          LDY          CFS.BASE0+1
009062          LDX          CFS.BASE0
009063          BNE          FRSG055
009064          DEY
009065 FRSG055      DEX
009066          STX          CFS.PTR
009067          STY          CFS.PTR+1
009068 *
009069          JMP          FRSG070          ; AND EXIT
009070 * L2: CFS.PTR:=CFS.BASE-1
009071 *
009072 FRSG060      SEC
009073          LDA          CFS.BASE
009074          SBC          #1
009075          STA          CFS.PTR
009076          LDA          CFS.BASE+1
009077          SBC          #0
009078          STA          CFS.PTR+1
009079 *
009080 * IF CFS.PTR="FFFF" OR CFS.NEXT=0 THEN EXIT
009081 *
009082          BCC          FRSG070
009083          LDA          CFS.NEXT
009084          BEQ          FRSG070
009085 *
009086 * IF CFS.PTR > ST.LIM(CFS.NEXT) THEN EXIT
009087 *
009088          LDX          CFS.NEXT
009089          LDA          ST.LIML,X
009090          CMP          CFS.PTR

```

```

009091          LDA      ST.LIMH,X
009092          SBC      CFS.PTR+1
009093          BCC      FRSG070
009094 *
009095 * OTHERWISE, ADVANCE CFS PTR TO NEXT FREE PAGE BELOW NEXT
009096 * SEGMENT IN SEGMENT LIST
009097 *
009098          JSR      NXTFRPG
009099 *
009100 FRSG070      CLC
009101          RTS              ; EXIT - FREE SEGMENT FOUND
009102          PAGE
009103          REP      60
009104 *
009105 * NEXT FREE PAGE
009106 *
009107 * "WALKS" THE FREE PAGE PTR (CFS.PTR) TO THE NEXT FREE PAGE
009108 * IMMEDIATELY BELOW THE CURRENT FREE SEGMENT.
009109 *
009110          REP      60
009111 *
009112 NXTFRPG      EQU      *
009113 *
009114 * L0: CFS.PTR:=ST.BASE(CFS.NEXT)-1
009115 *     IF CFS.PTR="FFFF" THEN DONE
009116 *
009117          LDX      CFS.NEXT
009118          SEC
009119          LDA      ST.BASEL,X
009120          SBC      #1
009121          STA      CFS.PTR
009122          LDA      ST.BASEH,X
009123          SBC      #0
009124          STA      CFS.PTR+1
009125          BCC      NFRPG.EXIT
009126 *
009127 * CFS.PREV:=CFS.NEXT
009128 * CFS.NEXT:=ST.FLINK(CFS.NEXT)
009129 *
009130          STX      CFS.PREV
009131          LDA      ST.FLINK,X
009132          TAX
009133          STX      CFS.NEXT
009134 *
009135 * IF CFS.NEXT=0 OR ST.LIM(CFS.NEXT)<CFS.PTR
009136 *     THEN DONE
009137 *     ELSE GOTO L0
009138 *
009139          BEQ      NFRPG.EXIT
009140          LDA      ST.LIML,X

```

```

009141          CMP          CFS.PTR
009142          LDA          ST.LIMH,X
009143          SBC          CFS.PTR+1
009144          BCS          NXTFRPG
009145          *
009146 NFRPG.EXIT          RTS                      ; NORMAL EXIT
009147          PAGE
009148          REP          60
009149          *
009150          * CHANGE.SEG(IN.SEGNUM,CHG.MODE; INOUT.PAGECT) SYSTEM CALL
009151          *
009152          REP          60
009153          *
009154 CHG.SEG          EQU          *
009155          *
009156          * MOVE CALLER'S PAGE COUNT TO INTERNAL BUFFER
009157          *
009158          LDY          #0
009159          LDA          (CHG.PGCT),Y
009160          STA          CHG.PGCTX
009161          INY
009162          LDA          (CHG.PGCT),Y
009163          STA          CHG.PGCTX+1
009164          *
009165          * IF SEG# OUT OF RANGE OR ST.FLINK(SEG#)=FREE THEN ERR
009166          *
009167          LDX          CHG.NUM
009168          BEQ          CHGS.ERR
009169          CPX          #ST.CNT
009170          BCS          CHGS.ERR
009171          LDA          ST.FLINK,X
009172          BPL          CHGS005
009173          *
009174 CHGS.ERR          LDA          #BADSEGNUM
009175          JSR          SYSERR                      ; ERR EXIT
009176          REP          35
009177          * CASE OF CHANGE MODE
009178          REP          35
009179 CHGS005          LDY          CHG.MODE
009180          CPY          #1
009181          BCC          CHGS010
009182          BEQ          CHGS020
009183          CPY          #3
009184          BCC          CHGS030
009185          BEQ          CHGS040
009186          *
009187          LDA          #BADCHGMODE
009188          JSR          SYSERR                      ; ERR EXIT
009189          PAGE
009190          REP          35

```

```

009191 * CHANGE MODE = 0(BASE UP)
009192         REP             35
009193 * CHG.NEW:=ST.BASE(SEG#)+PGCT
009194 *
009195 CHGS010      CLC
009196         LDA             ST.BASEL,X
009197         ADC             CHG.PGCTX
009198         STA             CHG.NEW
009199         LDA             ST.BASEH,X
009200         ADC             CHG.PGCTX+1
009201         STA             CHG.NEW+1
009202 *
009203         BCS             CHGS014           ; OVERFLOW, PEG IT
009204 *
009205 * IF CHG.NEW <= ST.LIM(SEG#) THEN EXIT
009206 *
009207         LDA             ST.LIML,X
009208         CMP             CHG.NEW
009209         LDA             ST.LIMH,X
009210         SBC             CHG.NEW+1
009211         BCS             CHGS016
009212 *
009213 * OTHERWISE, CHG.NEW:=ST.LIM(SEG#)
009214 *
009215 CHGS014      LDA             ST.LIML,X
009216         STA             CHG.NEW
009217         LDA             ST.LIMH,X
009218         STA             CHG.NEW+1
009219 *
009220 CHGS016      JMP             CHGS.EXIT
009221         REP             35
009222 * CHANGE MODE = 1(BASE DOWN)
009223         REP             35
009224 * CHG.NEW:=ST.BASE(SEG#)-PGCT
009225 *
009226 CHGS020      SEC
009227         LDA             ST.BASEL,X
009228         SBC             CHG.PGCTX
009229         STA             CHG.NEW
009230         LDA             ST.BASEH,X
009231         SBC             CHG.PGCTX+1
009232         STA             CHG.NEW+1
009233         BCS             CHGS050
009234         BCC             CHGS052           ; OVERFLOW, PEG IT
009235         REP             35
009236 * CHANGE MODE = 2(LIMIT UP)
009237         REP             35
009238 * CHG.NEW:=ST.LIM(SEG#)+PGCT
009239 *
009240 CHGS030      CLC

```



```

009241          LDA      ST.LIML,X
009242          ADC      CHG.PGCTX
009243          STA      CHG.NEW
009244          LDA      ST.LIMH,X
009245          ADC      CHG.PGCTX+1
009246          STA      CHG.NEW+1
009247          BCC      CHGS050
009248          BCS      CHGS052          ; OVERFLOW, PEG IT
009249          REP      35
009250 * CHANGE MODE = 3(LIMIT DOWN)
009251          REP      35
009252 * CHG.NEW:=ST.LIM(SEG#)-PGCT
009253 *
009254 CHGS040     SEC
009255          LDA      ST.LIML,X
009256          SBC      CHG.PGCTX
009257          STA      CHG.NEW
009258          LDA      ST.LIMH,X
009259          SBC      CHG.PGCTX+1
009260          STA      CHG.NEW+1
009261          BCC      CHGS044          ; OVERFLOW, PEG IT
009262 *
009263 * IF CHG.NEW >= ST.BASE(SEG#) THEN EXIT
009264 *
009265          LDA      CHG.NEW
009266          CMP      ST.BASEL,X
009267          LDA      CHG.NEW+1
009268          SBC      ST.BASEH,X
009269          BCS      CHGS046
009270 *
009271 * OTHERWISE CHG.NEW:=ST.BASE(SEG#)
009272 *
009273 CHGS044     LDA      ST.BASEL,X
009274          STA      CHG.NEW
009275          LDA      ST.BASEH,X
009276          STA      CHG.NEW+1
009277 *
009278 CHGS046     JMP      CHGS.EXIT
009279 *
009280 * DETERMINE NEW BANK/PAGE'S REGION,
009281 * IF NEW BANK/PAGE IS INVALID THEN
009282 * SET TO BASE OR LIMIT (CASE CHANGE MODE)
009283 *
009284 CHGS050     LDX      CHG.NEW
009285          LDY      CHG.NEW+1
009286          JSR      REGION
009287          BCS      CHGS052
009288          BNE      CHGS052
009289          BEQ      CHGS100
009290 CHGS052     LDA      CHG.MODE

```

```

009291          CMP          #1
009292          BNE          CHGS054
009293          LDX          #>VRT.BASE
009294          LDY          #<VRT.BASE
009295          JMP          CHGS056
009296 CHGS054    LDX          >VRT.LIM
009297          LDY          >VRT.LIM+1
009298 CHGS056    STX          CHG.NEW
009299          STY          CHG.NEW+1
009300          PAGE
009301          *
009302          * COMPUTE BANK/PAGE OF ADJACENT SEGMENT, IF ANY
009303          *   CASE CHANGE MODE
009304          *
009305 CHGS100    LDX          CHG.NUM
009306          LDA          CHG.MODE
009307          CMP          #1
009308          BNE          CHGS200
009309          *   "1" IF ST.FLINK(SEG#)=0 THEN EXIT
009310          LDA          ST.FLINK,X
009311          BEQ          CHGS.EXIT
009312          *   X,Y:=ST.LIM(ST.FLINK(SEG#))+1
009313          TAY
009314          LDA          ST.LIML,Y
009315          TAX
009316          LDA          ST.LIMH,Y
009317          TAY
009318          INX
009319          BNE          CHGS110
009320          INY
009321          *   IF CHG.NEW < X,Y THEN CHG.NEW:=X,Y
009322 CHGS110    CPY          CHG.NEW+1
009323          BCC          CHGS.EXIT
009324          BEQ          CHGS120
009325          BCS          CHGS300
009326 CHGS120    CPX          CHG.NEW
009327          BCC          CHGS.EXIT
009328          BCS          CHGS300
009329          *   "2" IF ST.BLINK(SEG#)=0 THEN EXIT
009330 CHGS200    LDA          ST.BLINK,X
009331          BEQ          CHGS.EXIT
009332          *   X,Y:= ST.BASE(ST.BLINK(SEG#))-1
009333          TAY
009334          LDA          ST.BASEL,Y
009335          TAX
009336          LDA          ST.BASEH,Y
009337          TAY
009338          TXA
009339          BNE          CHGS210
009340          DEY

```

```

009341 CHGS210      DEX
009342 *          IF CHG.NEW > X,Y THEN CHG.NEW:=X,Y
009343          CPY      CHG.NEW+1
009344          BCC      CHGS300
009345          BEQ      CHGS220
009346          BCS      CHGS.EXIT
009347 CHGS220      CPX      CHG.NEW
009348          BCS      CHGS.EXIT
009349 *
009350 CHGS300      STX      CHG.NEW
009351          STY      CHG.NEW+1
009352          PAGE
009353          REP      35
009354 *
009355 * COMPUTE DELTA PAGE COUNT AND RETURN IT TO CALLER
009356 * (CASE OF CHG.MODE)
009357 *
009358          REP      35
009359 CHGS.EXIT     LDX      CHG.NUM
009360          LDY      #0
009361          LDA      CHG.MODE
009362          CMP      #1
009363          BCC      CHGS500
009364          BEQ      CHGS510
009365          CMP      #3
009366          BCC      CHGS520
009367          BEQ      CHGS530
009368 *
009369 * "0" -- PAGECOUNT:=NEW-BASE
009370 *
009371 CHGS500      SEC
009372          LDA      CHG.NEW
009373          SBC      ST.BASEL,X
009374          STA      (CHG.PGCT),Y
009375          LDA      CHG.NEW+1
009376          SBC      ST.BASEH,X
009377          JMP      CHGS600
009378 *
009379 * "1" -- PAGECOUNT:=BASE-NEW
009380 *
009381 CHGS510      SEC
009382          LDA      ST.BASEL,X
009383          SBC      CHG.NEW
009384          STA      (CHG.PGCT),Y
009385          LDA      ST.BASEH,X
009386          SBC      CHG.NEW+1
009387          JMP      CHGS600
009388 *
009389 * "2" -- PAGECOUNT:=NEW-LIM
009390 *

```

```

009391 CHGS520      SEC
009392             LDA      CHG.NEW
009393             SBC      ST.LIML,X
009394             STA      (CHG.PGCT),Y
009395             LDA      CHG.NEW+1
009396             SBC      ST.LIMH,X
009397             JMP      CHGS600
009398 *
009399 * "3" -- PAGECOUNT:=LIM-NEW
009400 *
009401 CHGS530      SEC
009402             LDA      ST.LIML,X
009403             SBC      CHG.NEW
009404             STA      (CHG.PGCT),Y
009405             LDA      ST.LIMH,X
009406             SBC      CHG.NEW+1
009407 *
009408 CHGS600      INY
009409             STA      (CHG.PGCT),Y
009410 *
009411 * IF NEW PAGE COUNT < REQUESTED PAGECOUNT THEN ERR
009412 *
009413             TAX
009414             DEY
009415             LDA      (CHG.PGCT),Y
009416             CMP      CHG.PGCTX
009417             TXA
009418             SBC      CHG.PGCTX+1
009419             BCS      CHGS610
009420             LDA      #SEGRQDN
009421             JSR      SYSERR          ; ERR EXIT
009422 *
009423 * OTHERWISE, ENTER CHG.NEW IN SEGMENT TABLE AND EXIT
009424 *
009425 CHGS610      LDX      CHG.NUM
009426             LDA      CHG.MODE
009427             CMP      #2
009428             LDA      CHG.NEW
009429             LDY      CHG.NEW+1
009430             BCS      CHGS620
009431 *
009432             STA      ST.BASEL,X
009433             TYA
009434             STA      ST.BASEH,X
009435             CLC
009436             RTS          ; NORMAL EXIT
009437 *
009438 *
009439 CHGS620      STA      ST.LIML,X
009440             TYA

```

```
009441          STA          ST.LIMH,X
009442          CLC
009443          RTS                      ; NORMAL EXIT
009444
009445          CHN          MEMMGR.C.SRC
009446
009447 *****
009448 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.B.SRC
009449 *****
009450
009451
009452
```

```

009453 =====
009454 DOCUMENT :SOS1.3.2of5.TWO:SOS.MMGR.C.TEXT
009455 =====
009456
009457 *****
009458 * APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.C.SRC
009459 *****
009460 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
009461
009462             PAGE
009463             REP             60
009464 *
009465 * GET.SEG.INFO(IN.SEGNUM; OUT.BASE.BKPG,LIMIT.BKPG,PGCT,SEGID)
009466 *
009467             REP             60
009468 *
009469 GET.SEG.INFO EQU *
009470 *
009471 * IF SEG# OUT OF BOUNDS OR ST.FLINK(SEG#)=ST.FREE THEN ERR
009472 *
009473             LDX             GSI.NUM
009474             BEQ             GSI.ERR             ; ERR - INVALID SEGNUM
009475             CPX             #ST.CNT
009476             BCS             GSI.ERR             ; ERR - INVALID SEGNUM
009477             LDA             ST.FLINK,X
009478             BMI             GSI.ERR             ; ERR - INVALID SEGNUM
009479 *
009480 * RETURN BASE.BKPG TO CALLER
009481 *
009482             LDY             ST.BASEH,X
009483             LDA             ST.BASEL,X
009484             TAX
009485             JSR             CNVRT.XBP
009486             TYA
009487             LDY             #1
009488             STA             (GSI.BASE),Y
009489             DEY
009490             TXA
009491             STA             (GSI.BASE),Y
009492 *
009493 * RETURN LIMIT.BKPG TO CALLER
009494 *
009495             LDX             GSI.NUM
009496             LDY             ST.LIMH,X
009497             LDA             ST.LIML,X
009498             TAX
009499             JSR             CNVRT.XBP
009500             TYA
009501             LDY             #1

```

```

009502          STA      (GSI.LIM),Y
009503          DEY
009504          TXA
009505          STA      (GSI.LIM),Y
009506 *
009507 * RETURN SEGID TO CALLER
009508 *
009509          LDX      GSI.NUM
009510          LDA      ST.ID,X
009511          STA      (GSI.ID),Y
009512 *
009513 * COMPUTE PAGE COUNT
009514 *
009515          SEC
009516          LDA      ST.LIML,X
009517          SBC      ST.BASEL,X
009518          TAY
009519          LDA      ST.LIMH,X
009520          SBC      ST.BASEH,X
009521          TAX
009522          INY
009523          BNE      GSI010
009524          INX
009525 *
009526 * RETURN PAGE COUNT TO CALLER
009527 *
009528 GSI010      TYA
009529          LDY      #0
009530          STA      (GSI.PGCT),Y
009531          INY
009532          TXA
009533          STA      (GSI.PGCT),Y
009534 *
009535          CLC
009536          RTS                      ; NORMAL EXIT
009537 *
009538 GSI.ERR      LDA      #BADSEGNUM
009539          JSR      SYSERR          ; ERR EXIT
009540          PAGE
009541          REP      60
009542 *
009543 * GET.SEG.NUM(IN.BANKPAGE; OUT.SEGNUM) SYSTEM CALL
009544 *
009545 *
009546          REP      60
009547 *
009548 GET.SEG.NUM  EQU      *
009549 *
009550 * CONVERT BANKPAGE TO INTERNAL FORMAT
009551 *

```

```

009552          LDX      GSN.BKPG
009553          LDY      GSN.BKPG+1
009554          JSR      CNVRT.IBP
009555          BCS      GSN.ERR          ; ERR - INVALID BANK PAGE
009556          STX      GSN.BKPG
009557          STY      GSN.BKPG+1
009558 *
009559 * QUIT IF NO ENTRIES IN SEG TABLE
009560 *
009561          LDA      ST.ENTRY
009562          BEQ      GSN.ERR1          ; ERR - SEG NOT FOUND
009563 *
009564 * L1: IF BANKPAGE>ST.LIM(SEG#) THEN ERR
009565 *
009566 GSN010      TAX
009567          LDA      ST.LIML,X
009568          CMP      GSN.BKPG
009569          LDA      ST.LIMH,X
009570          SBC      GSN.BKPG+1
009571          BCC      GSN.ERR1          ; ERR - SEG NOT FOUND
009572 *
009573 * IF BANKPAGE>=ST.BASE(SEG#) THEN FOUND!
009574 *
009575          LDA      GSN.BKPG
009576          CMP      ST.BASEL,X
009577          LDA      GSN.BKPG+1
009578          SBC      ST.BASEH,X
009579          BCS      GSN020
009580 *
009581 * SEG#:=ST.FLINK(SEG#); GOTO L1
009582 *
009583          LDA      ST.FLINK,X
009584          BEQ      GSN.ERR1          ; ERR - SEG NOT FOUND
009585          JMP      GSN010
009586 *
009587 * RETURN SEG# TO CALLER
009588 *
009589 GSN020      LDY      #0
009590          TXA
009591          STA      (GSN.NUM),Y
009592          CLC
009593          RTS          ; NORMAL EXIT
009594 *
009595 GSN.ERR      RTS          ; ERROR EXIT
009596 *
009597 GSN.ERR1     LDA      #SEGNOTFND
009598          JSR      SYSERR          ; ERROR EXIT
009599          PAGE
009600          REP      60
009601 *

```



```

009602 * RELEASE.SEG(IN.SEGNUM) SYSTEM CALL
009603 *
009604             REP             60
009605 *
009606 RELEASE.SEG     EQU             *
009607 *
009608 * IF ST.FLINK(SEG#)=ST.FREE THEN ERR
009609 *
009610             LDX             RLS.NUM
009611             BEQ             RLS.ALL             ; RELEASE.SEG(SEG#=0)
009612             CPX             #ST.CNT
009613             BCS             RLS.ERR             ; ERR - SEG# TOO LARGE
009614             LDA             ST.FLINK,X
009615             BMI             RLS.ERR             ; ERR - INVALID SEGNUM
009616             BPL             REL.SEG             ; RELEASE.SEG(SEG#>0)
009617             REP             35
009618 *
009619 * RELEASE ALL
009620 *
009621             REP             35
009622 RLS.ALL           LDX             ST.ENTRY
009623             BEQ             RLS0.EXIT
009624             STX             RLS.NUM
009625 *
009626 RLS0.LOOP        LDA             ST.ID,X
009627             CMP             #$10             ; CARRY SET/CLEARED HERE
009628 *
009629             LDA             ST.FLINK,X
009630             PHA
009631             BCC             RLS006             ; IF ID=SYS SEG THEN SKIP
009632             JSR             REL.SEG             ; RELEASE ONE SEGMENT
009633 RLS006           PLA
009634             BEQ             RLS0.EXIT
009635             STA             RLS.NUM
009636             TAX
009637             BNE             RLS0.LOOP         ; ALWAYS TAKEN
009638 *
009639 RLS0.EXIT         CLC
009640             RTS             ; NORMAL EXIT ; ALL NON SYSTEM SEGMENTS RELEASED.
009641             REP             35
009642 *
009643 * REL SEG
009644 *
009645             REP             35
009646 * Y:=ST.FLINK(SEG#)
009647 * X:=ST.BLINK(SEG#)
009648 *
009649 REL.SEG           TAY
009650             LDA             ST.BLINK,X
009651             TAX

```

```

009652 *
009653 * IF X<>0 THEN ST.FLINK(X):=Y
009654 *     ELSE ST.ENTRY:=Y
009655 *
009656         BEQ         RLS010
009657         TYA
009658         STA         ST.FLINK,X
009659         JMP         RLS020
009660 RLS010     STY         ST.ENTRY
009661 *
009662 * IF Y<>0 THEN ST.BLINK(Y):=X
009663 *
009664         TYA
009665 RLS020     BEQ         RLS030
009666         TXA
009667         STA         ST.BLINK,Y
009668 *
009669 * ST.FLINK(SEG#):=ST.FREE
009670 * ST.FREE:=SEG# AND #$80
009671 *
009672 RLS030     LDA         ST.FREE
009673         LDX         RLS.NUM
009674         STA         ST.FLINK,X
009675         TXA
009676         ORA         #$80
009677         STA         ST.FREE
009678 *
009679         CLC
009680         RTS                     ; NORMAL EXIT
009681 *
009682 RLS.ERR    LDA         #BADSEGNUM
009683         JSR         SYSERR         ; ERR EXIT
009684         PAGE
009685         REP         60
009686 *
009687 * CONVERT INTERNAL BANK PAGE
009688 *
009689 * INPUT:  EXTERNAL BANK (X)
009690 *         "    PAGE (Y)
009691 * OUTPUT: INTERNAL BKPG LOW (X)
009692 *         "    BKPG HIGH (Y)
009693 *         REGION (A) 0=>VIRT BANK
009694 *         1=>PHY BANK (0-$2000)
009695 *         2=>    "    ($A000-$FFFF)
009696 * ERROR:  CARRY SET ("INVALID BANK PAGE")
009697 *
009698         REP         60
009699 *
009700 CNVRT.IBP  EQU         *
009701 *

```

```

009702 * CONVERT FROM EXTERNAL TO INTERNAL FORMAT
009703 *
009704 * CASE OF BANK:  ADD PAGE BIAS
009705 *
009706             TYA
009707             CPX             #$F
009708             BEQ             CNVI010
009709             BCS             CNVI020
009710 *
009711             CMP             #$20             ; BANK < "F"
009712             BCC             CNVI.ERR1
009713             CMP             #$A0
009714             BCS             CNVI.ERR1
009715             SEC
009716             SBC             #$20
009717             JMP             CNVI030
009718 *
009719 CNVI010     CMP             #$20             ; BANK = "F"
009720             BCS             CNVI.ERR1
009721             CLC
009722             ADC             #$80
009723             JMP             CNVI030
009724 *
009725 CNVI020     CPX             #$10             ; BANK = "10"
009726             BNE             CNVI.ERR1
009727             CMP             #$A0
009728             BCC             CNVI.ERR1
009729             SEC
009730             SBC             #$80
009731 *
009732 CNVI030     TAY             ; SHIFT BANK RIGHT ONE BIT
009733             TXA             ; INTO HIGH BIT OF PAGE BYTE.
009734             LSR             A
009735             TAX
009736             TYA
009737             BCC             CNVI040
009738             ORA             #$80
009739 *
009740 * EXCHANGE X & Y
009741 *
009742 CNVI040     PHA
009743             TXA
009744             TAY
009745             PLA
009746             TAX
009747 *
009748 * COMPUTE REGION (VIRT=0,PHY1=1,PHY2=2)
009749 *
009750             JSR             REGION             ; REGION RETURNED IN A REG.
009751             BCS             CNVI.ERR1             ; ERR - INVALID BANK PAGE

```

```

009752 *
009753         RTS                ; NORMAL EXIT
009754 *
009755 CNVI.ERR1     LDA          #BADBKPG
009756             JSR           SYSERR
009757             PAGE
009758             REP           60
009759 *
009760 * CONVERT EXTERNAL BANK PAGE
009761 *
009762 * INPUT:  INTERNAL BKPG LOW  (X)
009763 *         "           HIGH (Y)
009764 * OUTPUT: EXTERNAL BANK (X)
009765 *         "           PAGE (Y)
009766 * ERROR: NO ERROR CHECKING DONE. ASSUMES THAT INTERNAL #S
009767 * ARE VALID.
009768 *
009769             REP           60
009770 *
009771 CNVRT.XBP     EQU          *
009772 *
009773 * CONVERT FROM INTERNAL TO EXTERNAL FORMAT
009774 *
009775             TXA
009776             ASL           A
009777             TXA
009778             AND           #$7F
009779             TAX
009780             TYA
009781             ROL           A
009782             TAY
009783 *
009784 * CASE OF BANK: ADD PAGE BIAS
009785 *
009786             TXA
009787             CPY           #$F
009788             BEQ           CNVX020        ; BANK = "F"
009789             BCS           CNVX010
009790 *
009791             CLC                ; BANK < "F"
009792             ADC           #$20
009793             JMP           CNVX020
009794 *
009795 CNVX010       CLC                ; BANK = "10"
009796             ADC           #$80
009797 *
009798 * EXCHANGE X & Y
009799 *
009800 CNVX020       PHA
009801             TYA

```

```

009802          TAX
009803          PLA
009804          TAY
009805          RTS                ; NORMAL EXIT
009806          PAGE
009807          REP          60
009808 *
009809 * REGION
009810 *
009811 * INPUT:  INTERNAL BKPG LOW  (X)
009812 *          "                HIGH (Y)
009813 * OUTPUT: REGION (A)
009814 *          INTERNAL BKPG LOW  (X) UNCHANGED
009815 *          "                HIGH (Y)          "
009816 * ERROR:  CARRY SET ("INVALID BANK/PAGE")
009817 *
009818          REP          60
009819 *
009820 REGION          EQU          *
009821          STX          RGN.BKPG
009822          STY          RGN.BKPG+1
009823 *
009824 * IF BANKPAGE>PHY2LIM THEN ERR
009825 *
009826          LDA          #>PHY2LIM
009827          CMP          RGN.BKPG
009828          LDA          #<PHY2LIM
009829          SBC          RGN.BKPG+1
009830          BCC          RGN.ERR                ; ERR - INVALID BANK PAGE
009831 *
009832 * IF BANKPAGE>=PHY2BASE THEN REGION:=2
009833 *
009834          LDA          RGN.BKPG
009835          CMP          #>PHY2BASE
009836          LDA          RGN.BKPG+1
009837          SBC          #<PHY2BASE
009838          BCC          RGN010
009839          LDA          #2
009840          BNE          RGN040
009841 *
009842 * IF BANKPAGE>PHY1LIMIT THEN ERR
009843 *
009844 RGN010          LDA          #>PHY1LIM
009845          CMP          RGN.BKPG
009846          LDA          #<PHY1LIM
009847          SBC          RGN.BKPG+1
009848          BCC          RGN.ERR                ; ERR - INVALID BANK PAGE
009849 *
009850 * IF BANKPAGE>=PHY1BASE THEN REGION:=1
009851 *

```

```

009852          LDA      RGN.BKPG
009853          CMP      #>PHY1BASE
009854          LDA      RGN.BKPG+1
009855          SBC      #<PHY1BASE
009856          BCC      RGN020
009857          LDA      #1
009858          BNE      RGN040
009859          *
009860          * IF BANKPAGE>VIRTUAL LIMIT THEN ERR
009861          *
009862 RGN020      LDA      >VRT.LIM
009863          CMP      RGN.BKPG
009864          LDA      >VRT.LIM+1
009865          SBC      RGN.BKPG+1
009866          BCC      RGN.ERR
009867          LDA      #0
009868          *
009869 RGN040      CLC                      ; "N" FLAG ALWAYS REFLECTS REGION VAL IN A REG!
009870          RTS                      ; NORMAL EXIT
009871          *
009872 RGN.ERR      SEC                      ; INVALID BANK PAGE
009873          RTS
009874          PAGE
009875          REP      60
009876          *
009877          * GET FREE
009878          *
009879          * INPUT:  PREVIOUS SEG # (A)
009880          * OUTPUT: NEW SEG #      (A)
009881          * ERROR:  CARRY SET ("SEG TBL FULL")
009882          *
009883          REP      60
009884          *
009885 GET.FREE      EQU      *
009886          *
009887          * SAVE PREV SEG # IN X
009888          * NOTE:  PREV SEG # CARRIED IN X
009889          *       NEW SEG # CARRIED IN Y
009890          *
009891          TAX
009892          *
009893          * IF NO FREE ENTRIES THEN ERR
009894          *
009895          LDA      ST.FREE
009896          CMP      #$80
009897          BEQ      GTFR.ERR
009898          *
009899          * TURN OFF FREE FLAG (BIT7) AND DELINK FROM FREE LIST
009900          *
009901          AND      #$7F

```

```

009902          TAY
009903          LDA          ST.FLINK,Y
009904          STA          ST.FREE
009905 *
009906 * IF PREV SEG # IS NULL THEN LINK NEW ENTRY TO START
009907 * OF SEGMENT LIST
009908 *
009909          CPX          #0
009910          BNE          GTFR010
009911          LDA          ST.ENTRY
009912          STA          ST.FLINK,Y
009913          LDA          #0
009914          STA          ST.BLINK,Y
009915          STY          ST.ENTRY
009916          JMP          GTFR020
009917 *
009918 * OTHERWISE LINK NEW ENTRY TO PREV SEG #
009919 *
009920 GTFR010      LDA          ST.FLINK,X
009921          STA          ST.FLINK,Y
009922          TXA
009923          STA          ST.BLINK,Y
009924          TYA
009925          STA          ST.FLINK,X
009926 *
009927 * IF ST.FLINK(NEW)<>NULL THEN
009928 *   ST.BLINK(ST.FLINK(NEW)):=NEWSEG #
009929 GTFR020      LDA          ST.FLINK,Y
009930          BEQ          GTFR030
009931          LDA          ST.FLINK,Y
009932          TAX
009933          TYA
009934          STA          ST.BLINK,X
009935 *
009936 * RETURN WITH NEW SEG #
009937 *
009938 GTFR030      TYA
009939          CLC
009940          RTS                      ; NORMAL EXIT
009941 *
009942 GTFR.ERR     LDA          #SEGTBLFULL
009943          JSR          SYSERR
009944 *
009945          LST          ON
009946 ZZEND       EQU          *
009947 ZZLEN       EQU          ZZEND-ZZORG
009948          IFNE       ZZLEN-LENMEMMG
009949          FAIL       2,"SOSORG          FILE IS INCORRECT FOR MEMMGR"
009950          FIN
009951

```

009952 \*\*\*\*\*  
009953 \* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: MEMMGR.C.SRC  
009954 \*\*\*\*\*  
009955  
009956  
009957



```

009958 =====
009959 DOCUMENT :SOS1.3.2of5.TWO:SOS.SCMGR.TEXT
009960 =====
009961
009962 *****
009963 * APPLE /// SOS 1.3 SOURCE CODE FILE: SCMGR.SRC
009964 *****
009965 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
009966
009967          SBTL          "SOS 1.1 SYSTEM CALL MANAGER"
009968          REL
009969          INCLUDE      SOSORG,6,1,254
009970          ORG          ORGSCMGR
009971 ZZORG          EQU          *
009972          MSB          OFF
009973          REP          60
009974 *          COPYRIGHT (C) APPLE COMPUTER INC. 1980
009975 *          ALL RIGHTS RESERVED
009976          REP          60
009977 *
009978 * SYSTEM CALL MANAGER (VERSION = 1.10 )
009979 *          (DATE      = 8/04/81)
009980 *
009981 * THE SYSTEM CALL MANAGER:
009982 * (1) RETRIEVE THE SYSCALL #,
009983 * (2) DETERMINE THE LOCATION OF THE SYSTEM CALL PARMS AND
009984 *     MOVE THEM TO THE SOS ZPAGE,
009985 * (3) TRANSFER CONTROL TO THE APPROPRIATE INTERFACE MANAGER,
009986 *     (FILE,DEVICE,UTILITY,MEMORY)
009987 *
009988          REP          60
009989 *
009990          ENTRY        SCMGR
009991 *
009992          EXTRN        FMGR
009993          EXTRN        DMGR
009994          EXTRN        UMGR
009995          EXTRN        MMGR
009996          EXTRN        DBUGBRK
009997 *
009998          EXTRN        SYSERR
009999          EXTRN        SERR
010000          EXTRN        BADSCNUM
010001          EXTRN        BADCZPAGE
010002          EXTRN        BADXBYTE
010003          EXTRN        BADSCPCNT
010004          EXTRN        BADSCBNDS
010005 *
010006          EXTRN        SZPAGE

```

```

010007          EXTRN      SXPAGE
010008          EXTRN      CZPAGE
010009          EXTRN      CXPAGE
010010          EXTRN      CSPAGE
010011          PAGE
010012          REP        60
010013 *
010014 * SYSTEM CALL PARAMETER DEFINITION TABLES
010015 *
010016 * EACH ENTRY IS FOUR BYTES LONG.  THE FIRST BYTE CONTAINS THE
010017 * NUMBER OF PARMS IN THE CALL.  THE REMAINING SIX NIBBLES, EACH
010018 * DEFINE A PARAMETER IN THE CALL.  THE FIRST BIT OF THE
010019 * NIBBLE DEFINES WHETHER THE PARM IS INPUT (0) OR OUTPUT (1).
010020 * THE NEXT BIT DEFINES WHETHER THE PARM IS BY VALUE (0)
010021 * OR BY REFERENCE (1).  THE FINAL TWO BITS SPECIFY THE
010022 * PARM LENGTH IN BYTES (E.G. 0=LENGTH OF 1, 3=LENGTH OF 4 BYTES)
010023 *
010024          REP        60
010025 *
010026 * FILE SYSTEM CALL DEFINITIONS
010027 *
010028 FSC.CNT      EQU        $13
010029 FSC.TBL     EQU        *
010030          DFB          $3,$5D,$00,$00      ; SCNUM=$C0 - CREATE
010031          DFB          $1,$50,$00,$00      ; "  =$C1 - DESTROY
010032          DFB          $2,$55,$00,$00      ; "  =$C2 - RENAME
010033          DFB          $3,$5D,$00,$00      ; "  =$C3 - SET.FILE.INFO
010034          DFB          $3,$5D,$00,$00      ; "  =$C4 - GET.FILE.INFO
010035          DFB          $4,$55,$99,$00      ; "  =$C5 - VOLUME
010036          DFB          $1,$50,$00,$00      ; "  =$C6 - SET.PREFIX
010037          DFB          $2,$50,$00,$00      ; "  =$C7 - GET.PREFIX
010038          DFB          $4,$58,$D0,$00      ; "  =$C8 - OPEN
010039          DFB          $3,$00,$00,$00      ; "  =$C9 - NEW.LINE
010040          DFB          $4,$05,$19,$00      ; "  =$CA - READ
010041          DFB          $3,$05,$10,$00      ; "  =$CB - WRITE
010042          DFB          $1,$00,$00,$00      ; "  =$CC - CLOSE
010043          DFB          $1,$00,$00,$00      ; "  =$CD - FLUSH
010044          DFB          $3,$00,$30,$00      ; "  =$CE - SET.MARK
010045          DFB          $2,$0B,$00,$00      ; "  =$CF - GET.MARK
010046          DFB          $3,$00,$30,$00      ; "  =$D0 - SET.EOF
010047          DFB          $2,$0B,$00,$00      ; "  =$D1 - GET.EOF
010048          DFB          $1,$00,$00,$00      ; "  =$D2 - SET.LEVEL
010049          DFB          $1,$80,$00,$00      ; "  =$D3 - GET.LEVEL
010050          PAGE
010051 *
010052 * DEVICE SYSTEM CALL DEFINITIONS
010053 *
010054 DSC.CNT      EQU        5
010055 DSC.TBL     EQU        *
010056          DFB          $5,$05,$11,$90      ; SCNUM=$80 - D.READ

```

```

010057          DFB          $4,$05,$11,$00      ; "  = $81 - D.WRITE
010058          DFB          $3,$00,$50,$00      ; "  = $82 - D.STATUS
010059          DFB          $3,$00,$50,$00      ; "  = $83 - D.CONTROL
010060          DFB          $2,$58,$00,$00      ; "  = $84 - GET.DEV.NUM
010061          DFB          $4,$05,$D0,$00      ; "  = $85 - D.INFO
010062 *
010063 *   UTILITY SYSTEM CALL DEFINITIONS
010064 *
010065 USC.CNT      EQU          5
010066 USC.TBL      EQU          *
010067          DFB          $1,$00,$00,$00      ; SCNUM=$60 - SET.FENCE
010068          DFB          $1,$80,$00,$00      ; "  = $61 - GET.FENCE
010069          DFB          $1,$50,$00,$00      ; "  = $62 - SET.TIME
010070          DFB          $1,$50,$00,$00      ; "  = $63 - GET.TIME
010071          DFB          $2,$0B,$00,$00      ; "  = $64 - JOYSTICK
010072          DFB          $0,$00,$00,$00      ; "  = $65 - COLD.START
010073 *
010074 *   MEMORY SYSTEM CALL DEFINITIONS
010075 *
010076 MSC.CNT      EQU          5
010077 MSC.TBL      EQU          *
010078          DFB          $4,$11,$08,$00      ; SCNUM=$40 - REQUEST.SEG
010079          DFB          $6,$00,$99,$98      ; "  = $41 - FIND.SEG
010080          DFB          $3,$00,$90,$00      ; "  = $42 - CHANGE.SEG
010081          DFB          $5,$09,$99,$80      ; "  = $43 - GET.SEG.INFO
010082          DFB          $2,$18,$00,$00      ; "  = $44 - GET.SEG.NUM
010083          DFB          $1,$00,$00,$00      ; "  = $45 - RELEASE.SEG
010084 *
010085 *   DEBUG SYSTEM CALL DEFINITION
010086 *
010087 DEBUG        EQU          $FE
010088          PAGE
010089          REP          60
010090 *
010091 *   DATA DECLARATIONS
010092 *
010093          REP          60
010094 Z.REG          EQU          $FFD0
010095 SP.SAVE        EQU          $01FF
010096 Z.SAVE          EQU          $01FD
010097 B.SAVE          EQU          $01FC
010098 *
010099 ADR.LOW         EQU          $2000          ; LOW   ADDRESS   (BOUNDS CHECKING)
010100 ADR.HIGH        EQU          $B800          ; HIGH  ADDRESS
010101 ADR.MID        EQU          $A000          ; MIDDLE ADDRESS
010102 *
010103 *   SCMGR'S VARIABLES
010104 *
010105 SCM.VARS        EQU          $E0
010106 SCNUM          EQU          SCM.VARS+0      ; SYSTEM CALL NUMBER

```

```

010107 SCRNUM      EQU      SCM.VARS+0      ; SYSTEM CALL REQUEST NUMBER
010108 SCPTR      EQU      SCM.VARS+1      ;&2 SYSTEM CALL POINTER
010109 MOVE.VARS  EQU      SCPTR+2         ; !! (LOOKOUT) !!
010110 *
010111 *
010112 F.TPARMX    EQU      $A0             ; FILE SYS CALL PARM START LOC
010113 D.TPARMX    EQU      $C0             ; DEVICE SYS CALL PARM START LOC
010114 U.TPARMX    EQU      $C0             ; UTILITY SYS CALL PARM START LOC
010115 M.TPARMX    EQU      $60             ; MEMORY SYS CALL PARM START LOC
010116 *
010117 * MOVE.PARM'S VARIABLES
010118 *
010119 TPARMX      EQU      MOVE.VARS+0     ; TARGET ADR OF SYS CALL PARMS
010120 DFN.PTR      EQU      MOVE.VARS+1     ;&2
010121 DFN.PTRX    EQU      MOVE.VARS+3
010122 SCPTRX      EQU      MOVE.VARS+4
010123 RGHT.NIB    EQU      MOVE.VARS+5
010124 SCT.DFN     EQU      MOVE.VARS+6
010125 SCT.DCNT    EQU      MOVE.VARS+7
010126 PARM.CNT   EQU      MOVE.VARS+8
010127            PAGE
010128            REP      60
010129 *
010130 * SYSTEM CALL MANAGER
010131 *
010132            REP      60
010133 *
010134 SCMGR        EQU      *
010135            LDA      #<SZPAGE          ; SET Z REG TO SOS ZPAGE
010136            STA      Z.REG
010137 *
010138 * SET SYSTEM X BYTES TO ABSOLUTE ADDRESS MODE.
010139 *
010140            LDA      #0
010141            STA      SXPAGE+SCPTR+1
010142            STA      SERR              ; AND INIT SYSTEM ERR CODE
010143 *
010144 * CALLER'S Z REG MUST BE $1A !!
010145 * (B REG NOT CHECKED)
010146 *
010147            LDA      Z.SAVE
010148            CMP      #<CZPAGE
010149            BEQ      SCM005
010150            LDA      #>BADCZPAGE
010151            JSR      SYSERR            ; EXIT TO DISPATCHER
010152 *
010153 * RETRIEVE CALLER'S PC ON HIS STACK
010154 *
010155 SCM005         LDX      SP.SAVE
010156            LDA      CSPAGE+6,X

```

```

010157          STA      SCPTR+1
010158          LDA      CSPAGE+5,X
010159          STA      SCPTR
010160          BNE      SCM010          ; AND POINT IT TO SYS CALL NUM
010161          DEC      SCPTR+1
010162 SCM010    DEC      SCPTR
010163          *
010164          * ADVANCE CALLER'S PC ON HIS STACK.
010165          *
010166          CLC
010167          LDA      CSPAGE+5,X
010168          ADC      #2
010169          STA      CSPAGE+5,X
010170          BCC      SCM020
010171          INC      CSPAGE+6,X
010172          *
010173          * RETRIEVE SYSTEM CALL NUMBER
010174          *
010175 SCM020    LDY      #0
010176          LDA      (SCPTR),Y
010177          CMP      #DEBUG
010178          BNE      SCM025
010179          JSR      DEBUGBRK          ; DEBUG SYSTEM CALL
010180 SCM025    STA      SCNUM
010181          *
010182          * RETRIEVE SYSTEM CALL PARAMETER ADDRESS
010183          *
010184          INY
010185          LDX      #>SCPTR
010186          JSR      POINTER
010187          BCC      SCM030
010188          RTS          ; ERROR EXIT
010189          *
010190          * CASE INTERFACE CODE OF SYSTEM CALL NUMBER
010191          * (INTERFACE CODE STRIPPED, LEAVING REQUEST CODE)
010192          *
010193 SCM030    LDA      #$20
010194          BIT      SCNUM
010195          BPL      SCM050
010196          LDA      SCNUM
010197          AND      #$3F
010198          STA      SCRNUM
010199          BVC      SCM040
010200          *
010201          LDA      #F.TPARMX          ; "11XXXXXX" - JMP TO FILE MANAGER.
010202          STA      TPARMX
010203          LDX      #>FSC.TBL
010204          LDY      #<FSC.TBL
010205          LDA      #FSC.CNT
010206          JSR      MOVE.PARMS

```

```

010207          BCS      SCM.ERR1          ; ERR EXIT
010208          JMP      FMGR
010209  *
010210  SCM040      LDA      #D.TPARMX          ; "10XXXXXX" - JMP TO DEVICE MANAGER.
010211          STA      TPARMX
010212          LDX      #>DSC.TBL
010213          LDY      #<DSC.TBL
010214          LDA      #DSC.CNT
010215          JSR      MOVE.PARMS
010216          BCS      SCM.ERR1          ; ERR EXIT
010217          JMP      DMGR
010218  *
010219  SCM050      BVC      SCM.ERR
010220          PHP
010221          LDA      SCNUM
010222          AND      #$1F
010223          STA      SCRNUM
010224          PLP
010225          BEQ      SCM060
010226  *
010227          LDA      #U.TPARMX          ; "011XXXXX" - JMP TO UTILITY MANAGER.
010228          STA      TPARMX
010229          LDX      #>USC.TBL
010230          LDY      #<USC.TBL
010231          LDA      #USC.CNT
010232          JSR      MOVE.PARMS
010233          BCS      SCM.ERR1          ; ERR EXIT
010234          JMP      UMGR
010235  *
010236  SCM060      LDA      #M.TPARMX          ; "010XXXXX" - JMP TO MEMORY MANAGER.
010237          STA      TPARMX
010238          LDX      #>MSC.TBL
010239          LDY      #<MSC.TBL
010240          LDA      #MSC.CNT
010241          JSR      MOVE.PARMS
010242          BCS      SCM.ERR1          ; ERR EXIT
010243          JMP      MMGR
010244  *
010245  SCM.ERR      LDA      #>BADSCNUM          ; ERROR, INVALID SYSTEM CALL NUMBER.
010246  SCM.ERR1    JSR      SYSERR          ; EXIT TO DISPATCHER ON ERROR
010247          PAGE
010248          REP      60
010249  *
010250  * MOVE.PARMS
010251  *
010252  * MOVES THE CALLER'S PARAMETERS TO THE OPERATING SYSTEM'S
010253  * ZERO PAGE, ACCORDING TO THE SPECIFICATIONS CONTAINED
010254  * IN THE SPECIFIED SYS CALL DFN TABLE.
010255  *
010256  * INPUT: (A) = MAX # ENTRIES IN PARM DFN TABLE

```

```

010257 *          (X) = PARM DFN TBL ADR (LO)
010258 *          (Y) =          "          (HI)
010259 *          SCPTR = ADR OF CALLER'S SYS CALL PARMS
010260 * ERROR: CARRY SET (SYSERR)
010261 *
010262 *          REP          60
010263 *
010264 MOVE.PARMS      EQU          *
010265 *          STX          DFN.PTR          ; SAVE ADR OF DEFINITION TABLE
010266 *          STY          DFN.PTR+1
010267 *
010268 * IF REQ NUM > MAX REQ NUM (A REG)
010269 *
010270 *          CMP          SCRNUM
010271 *          BCS          MOVE010
010272 *
010273 * THEN ERR(BAD SYS CALL NUM)
010274 *
010275 *          LDA          #>BADSCNUM
010276 *          BCC          SYSERR1          ;BRANCH ALWAYS TAKEN
010277 *
010278 * CALCULATE DEFINITION TABLE INDEX
010279 * AND INIT SYS CALL PARM INDEX
010280 *
010281 MOVE010      LDA          SCRNUM
010282 *          ASL          A
010283 *          ASL          A
010284 *          STA          DFN.PTRX
010285 *          LDA          #0
010286 *          STA          SXPAGE+DFN.PTR+1 ; AND X BYTE
010287 *          STA          SCPTRX
010288 *
010289 * IF SCPTR(SCPTRX) <>DFN.PTR(DFN.PTRX) THEN ERR
010290 *
010291 *          TAY
010292 *          LDA          (SCPTR),Y
010293 *          LDY          DFN.PTRX
010294 *          CMP          (DFN.PTR),Y
010295 *          BEQ          INITLOOPCT
010296 *
010297 *          LDA          #>BADSCPCNT          ; ERR, CALLER'S PARM COUNT INVALID
010298 *          JSR          SYSERR          ; EXIT
010299 *
010300 * INIT LOOP CTR(PARM.CNT) TO # OF PARMS IN SYS CALL
010301 *
010302 INITLOOPCT      STA          PARM.CNT
010303 *
010304 * ADVANCE PTRS
010305 *
010306 *

```

```

010307             INC      SCPTRX
010308             INC      DFN.PTRX
010309 *
010310 * MOVE REQ CODE TO SYS ZPAGE PARM LIST
010311 * AND ADVANCE SYS ZPAGE PTR (X=TPARMX)
010312 *
010313             LDA      SCRNUM
010314             LDX      TPARMX
010315             STA      0,X
010316             INX
010317 *
010318 * INIT NIBBLE FLAG TO "RIGHT" NIBBLE
010319 * ZERO STATE="LEFT" NIBBLE
010320 *
010321             LDA      #$FF
010322             STA      RGHT.NIB
010323             REP      60
010324 *
010325 * BEGIN PARAMETER PROCESSING LOOP
010326 *
010327 PARMLOOP     LDA      RGHT.NIB
010328             EOR      #$FF                ; COMPLEMENT NIBBLE FLAG
010329             STA      RGHT.NIB
010330 *
010331 * IF "LEFT" NIBBLE
010332 *
010333             BNE      ELSE.RNIB
010334 *
010335 * THEN FETCH SYS CALL PARM DFN
010336 * AND # OF BYTES IN PARM WITHIN IT
010337 *
010338             LDY      DFN.PTRX
010339             LDA      (DFN.PTR),Y
010340             STA      SCT.DFN
010341             AND      #$30
010342             LSR      A
010343             LSR      A
010344             LSR      A
010345             LSR      A
010346             STA      SCT.DCNT
010347             BPL      VALUE                ;BRANCH ALWAYS
010348 *
010349 * ELSE FETCH SYS CALL PARM DFN
010350 * AND # OF BYTES IN PARM WITHIN IT
010351 * FROM "RIGHT" NIBBLE OF DFN BYTE
010352 *
010353 ELSE.RNIB     LDA      SCT.DFN
010354             TAY
010355             AND      #$03
010356             STA      SCT.DCNT

```



```

010357          TYA
010358          ASL      A
010359          ASL      A
010360          ASL      A
010361          ASL      A
010362          STA      SCT.DFN
010363          INC      DFN.PTRX          ; ADVANCE SYS CALL DFN PTR
010364          REP      60
010365 *
010366 *  PARAMETER PASSED BY VALUE
010367 *
010368          REP      60
010369 VALUE      BIT      SCT.DFN
010370          BVS      REFERENCE
010371          BMI      VAL.OUT
010372 *
010373 *  INPUT BY VALUE
010374 *
010375          LDY      SCPTRX          ; MOVE BYTES TO ZPAGE
010376 VAL.IN     LDA      (SCPTR),Y
010377          STA      0,X
010378          INY
010379          INX
010380          DEC      SCT.DCNT
010381          BPL      VAL.IN
010382          STY      SCPTRX
010383          JMP      ENDLOOP1
010384 *
010385 *  OUTPUT BY VALUE
010386 *
010387 VAL.OUT     CLC          ; BUILD PTR TO PARM ON ZPAGE
010388          LDA      SCPTR
010389          ADC      SCPTRX
010390          STA      0,X
010391          INX
010392          LDA      SCPTR+1
010393          ADC      #0
010394          STA      0,X
010395 *
010396          CLC          ; ADVANCE INDEX TO NEXT PARM
010397          LDA      SCPTRX
010398          ADC      SCT.DCNT
010399          STA      SCPTRX
010400 *
010401          LDA      SXPAGE+SCPTR+1    ; INCLUDE X BYTE
010402          STA      SXPAGE,X
010403          JMP      ENDLOOP2
010404          REP      60
010405 *
010406 *  PARAMETER PASSED BY REFERENCE

```

```

010407 *
010408             REP             60
010409 REFERENCE   BPL             REF1
010410 *
010411 * "LIST" PTR FOUND, CHK IF "LENGTH" PARM = 0
010412 *
010413             LDY             SCPTRX
010414             INY
010415             INY
010416             LDA             (SCPTR),Y
010417             BEQ             ENDLOOP0             ; "LENGTH" PARM=0, SKIP "LIST" PARM
010418 *
010419 REF1         LDY             SCPTRX             ; MOVE PTR TO ZPAGE
010420             JSR             POINTER
010421             BCS             PARM.ERR             ; ERROR EXIT
010422 *
010423 * ADVANCE SYSTEM ZPAGE POINTER (X), CALLER'S PARM PTR.
010424 * DECREMENT PARM CTR AND CHECK IF LAST PARM PROCESSED.
010425 *
010426 ENDLOOP0     INX
010427             INC             SCPTRX
010428 ENDLOOP2     INX
010429             INC             SCPTRX
010430 ENDLOOP1     DEC             PARM.CNT
010431             BEQ             PARM.EXIT
010432             BMI             PARM.EXIT             ;SPECIAL FOR 'COLD START'
010433             JMP             PARMLoop
010434 *
010435 * END OF PARAMETER PROCESSING LOOP
010436 *
010437             REP             60
010438 *
010439 PARM.EXIT     CLC                     ; NO ERRORS
010440 PARM.ERR     RTS                     ; RETURN TO SYS CALL MANAGER
010441             PAGE
010442             REP             60
010443 *
010444 * POINTER
010445 *
010446 * INPUT:   SRC ADR   (SCPTR),Y & (SCPTR),Y+1
010447 *         DEST ADR   (X)
010448 *
010449 * OUTPUT:  SCPTR     UNCHANGED
010450 *         X REG      "
010451 *         A,Y REGS   FLATTENED
010452 *
010453 * ERROR:   CARRY SET (SYSERR)
010454 *
010455 * POINTER.  RETRIEVES THE CALLER'S POINTER PARAMETER IN
010456 * (SCPTR),Y, PERFORMS ADDRESS COMPENSATION, IF NECESSARY

```

```

010457 * AND PLACES THE RESULTING POINTER AT X, X+1 AND SXPAGE+1,X.
010458 *
010459             REP             60
010460 *
010461 POINTER     EQU             *
010462             LDA             (SCPTR),Y
010463             PHA
010464             INY
010465             LDA             (SCPTR),Y
010466             BEQ             INDIRECT
010467 *
010468             STA             1,X             ; DIRECT POINTER
010469             PLA
010470             STA             0,X
010471             LDY             #0
010472             BEQ             PTR010
010473 *
010474 INDIRECT    PLA             ; INDIRECT POINTER
010475             TAY
010476             LDA             CZPAGE,Y
010477             STA             0,X
010478             LDA             CZPAGE+1,Y
010479             STA             1,X
010480             LDA             CXPAGE+1,Y
010481             TAY
010482 *
010483 PTR010      LDA             1,X
010484 *
010485 * CHECK BOUNDS OF CALLER'S POINTER PARAMETER
010486 *
010487             CPY             #$8F
010488             BCC             PTR.X808E
010489             BEQ             PTR.X8F
010490             BCS             PTR.ERR1             ; ERROR, INVALID X BYTE
010491 PTR.X8F      CMP             #<ADR.LOW
010492             BCC             PTR.ERR
010493             CMP             #<ADR.HIGH
010494             BCS             PTR.ERR
010495             BCC             PTR.EXIT
010496 *
010497 * X BYTE = 80..8E
010498 *
010499 PTR.X808E    CPY             #$80
010500             BCC             PTR.X0
010501             CMP             #0
010502             BEQ             PTR.ERR
010503             CMP             #$FF
010504             BNE             PATCH
010505             INY             ; $8N:FFXX --> $8N+1:7FXX
010506             LDA             #$7F

```

```

010507          BNE          PTR.EXIT
010508      *
010509      * X BYTE = 0
010510      *
010511 PTR.X0          CPY          #0
010512          BNE          PTR.ERR1
010513          CMP          #<ADR.LOW
010514          BCC          PTR.ERR
010515          CMP          #<ADR.HIGH
010516          BCS          PTR.ERR
010517          CMP          #<ADR.MID
010518          BCS          PTR.EXIT
010519      *
010520          PHA
010521          LDA          B.SAVE
010522          AND          #$0F
010523          BNE          PTR030
010524          PLA                      ; $B=0:2000..9FFF --> $8F:2000.9FFF
010525          LDY          #$8F
010526          BNE          PTR.EXIT
010527      *
010528 PTR030          ORA          #$80                      ; $B<>0:2000..9FFF --> $8B:0000..7FFF
010529          TAY
010530          PLA
010531          SEC
010532          SBC          #$20
010533          BNE          PATCH
010534          DEY                      ; $8B:00XX --> $8B-1:80XX
010535          LDA          #$80
010536      *
010537 PATCH          CPY          #$80                      ; KLUDGE FOR BFM: $8N:01XX --> $8N-1:81XX
010538          BCC          PTR.EXIT
010539          CMP          #1
010540          BNE          PTR.EXIT
010541          CPY          #$80
010542          BEQ          PTR.ERR                      ; ERROR, $80:01XX NOT ALLOWED
010543          DEY
010544          LDA          #$81
010545      *
010546 PTR.EXIT          STA          1,X
010547          TYA
010548          STA          SXPAGE+1,X
010549          CLC
010550          RTS
010551      *
010552      *
010553 PTR.ERR          LDA          #>BADSCBND$
010554          JSR          SYSERR
010555 PTR.ERR1          LDA          #>BADXBYTE
010556          JSR          SYSERR

```

```
010557 *
010558          LST          ON
010559 ZZEND          EQU          *
010560 ZZLEN          EQU          ZZEND-ZZORG
010561          IFNE          ZZLEN-LENSCMGR
010562          FAIL          2,"SOSORG          FILE IS INCORRECT FOR SCMGR"
010563          FIN
010564
010565 *****
010566 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SCMGR.SRC
010567 *****
010568
010569
```

```

010570 =====
010571 DOCUMENT :SOS1.3.2of5.TWO:SOS.SYSERR.TEXT
010572 =====
010573
010574 *****
010575 * APPLE /// SOS 1.3 SOURCE CODE FILE: SYSERR.SRC
010576 *****
010577 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
010578
010579             SBTL             "SOS 1.1 SYSTEM ERROR ROUTINES"
010580             REL
010581             INCLUDE         SOSORG,6,1,254
010582             ORG             ORGSERR
010583 ZZORG             EQU             *
010584             MSB             OFF
010585             REP             60
010586 *             COPYRIGHT (C) APPLE COMPUTER INC. 1980
010587 *             ALL RIGHTS RESERVED
010588             REP             60
010589 *
010590 * SYSTEM ERROR ROUTINES (VERSION = 1.10 )
010591 *             (DATE       = 12/02/81)
010592 *
010593 * THIS MODULE CONTAINS THE SYSTEM ERROR AND SYSTEM FAILURE ROUTINES.
010594 *
010595             REP             60
010596 *
010597             ENTRY          SYSERR
010598             ENTRY          SYSDEATH
010599 *
010600             EXTRN          SERR
010601             EXTRN          SDEATH.REGS
010602             EXTRN          SCRNMODE
010603             PAGE
010604             REP             60
010605 *
010606 * DATA DECLARATIONS
010607 *
010608             REP             60
010609 *
010610 E.REG             EQU             $FFDF
010611 Z.REG             EQU             $FFD0
010612 B.REG             EQU             $FFEF
010613 *
010614 S.SAVE            EQU             $09             ; REGISTER SAVE AREA
010615 PCH.SAVE          EQU             $08
010616 PCL.SAVE          EQU             $07
010617 P.SAVE            EQU             $06
010618 A.SAVE            EQU             $05

```

```

010619 X.SAVE      EQU      $04
010620 Y.SAVE      EQU      $03
010621 E.SAVE      EQU      $02
010622 Z.SAVE      EQU      $01
010623 B.SAVE      EQU      $00
010624 *
010625 NMI.VECTOR  EQU      $FFFA
010626 *
010627 TXT.CLR     EQU      $C050
010628 MIX.CLR     EQU      $C052
010629 HIRES.CLR   EQU      $C056
010630 *
010631 PG2.CLR     EQU      $C054
010632 *
010633 MSGBASE     EQU      $7E4
010634 MSGBASE2    EQU      $BE4
010635 MSG         ASC      ' SYSTEM FAILURE = $ '
010636 MSGLEN      EQU      *-MSG
010637             PAGE
010638             REP      60
010639 *
010640 * SYSTEM ERROR ROUTINE
010641 *
010642 * THIS ROUTINE IS CALLED WHEN AN ERROR CONDITION HAS BEEN
010643 * ENCOUNTERED. THE ERROR NUMBER IS PASSED IN THE A REG
010644 * AND THE CALL TO THIS ROUTINE MUST ALWAYS BE A JSR.
010645 *
010646             REP      60
010647 SYSERR        EQU      *
010648 *
010649             STA      SERR
010650             PLA
010651             STA      SDEATH.REGS+PCL.SAVE
010652             PLA
010653             STA      SDEATH.REGS+PCH.SAVE
010654             SEC
010655             LDA      SERR
010656             BNE      SERR.EXIT
010657             CLC
010658 SERR.EXIT      RTS              ; RETURNS ONE LEVEL BEYOND CALLER
010659             PAGE
010660             REP      60
010661 *
010662 * SYSTEM DEATH ROUTINE
010663 *
010664 * CALLED TO IMMEDIATELY TERMINATE EXECUTION OF THE MACHINE
010665 * BECAUSE A FATAL ERROR HAS BEEN DETECTED BY THE OPERATING
010666 * SYSTEM. THE ERROR CODE IS PASSED IN THE A REG. THE
010667 * CALL TO THIS ROUTINE MUST ALWAYS BE A JSR.
010668 *

```

```

010669          REP          60
010670  SYSDEATH  EQU          *
010671  *
010672          STA          SDEATH.REGS+A.SAVE ; SAVE REGISTERS
010673          STX          SDEATH.REGS+X.SAVE
010674          STY          SDEATH.REGS+Y.SAVE
010675          PHP
010676          PLA
010677          STA          SDEATH.REGS+P.SAVE
010678          TSX
010679          STX          SDEATH.REGS+S.SAVE
010680          LDA          E.REG
010681          STA          SDEATH.REGS+E.SAVE
010682          LDA          Z.REG
010683          STA          SDEATH.REGS+Z.SAVE
010684          LDA          B.REG
010685          STA          SDEATH.REGS+B.SAVE
010686          PLA
010687          STA          SDEATH.REGS+PCL.SAVE
010688          PLA
010689          STA          SDEATH.REGS+PCH.SAVE
010690  *
010691          SEI                      ; TURN OFF INTERRUPTS
010692          CLD
010693  *
010694          LDX          #0 ; SAVE SYSTEM STACK PAGE IN PAGE $17
010695  SD005      LDA          $100,X
010696          STA          $1700,X
010697          DEX
010698          BNE          SD005
010699  *
010700          LDA          $C059 ; ENSURE SILENTYPE PORT SHUT DOWN
010701          LDA          $C0DD
010702          LDA          $C0DF
010703          LDA          $C05F
010704          LDA          $C05A
010705  *
010706          LDA          $C040 ; SOUND BELL
010707  *
010708          LDA          #$74 ; ENSURE RESET LOCK OFF & RAM SWITCHED IN.
010709          STA          E.REG
010710  *
010711          LDA          TXT.CLR ; SWITCH TO 40 COL B&W DISPLAY MODE
010712          LDA          MIX.CLR
010713          LDA          HIRES.CLR
010714          LDA          PG2.CLR ; & SELECT PAGE 1
010715  *
010716          LDA          #$02
010717          BIT          SCRNMODE
010718          BVS          SD015 ; IF GRAPHICS MODE THEN KEEP 40 COL MODE

```



```

010719          BEQ      SD015          ; IF 40 COL MODE THEN KEEP
010720          LDA      MIX.CLR+1      ; ELSE SWITCH TO 80 COL DISPLAY MODE
010721 *
010722          LDX      #MSGLEN+1      ; ENSURE BKGRND SET TO INVERSE SPACES
010723          LDA      #$20            ; SPACE CHAR W/INVERSE
010724 SD010     STA      MSGBASE2-1,X
010725          DEX
010726          BPL      SD010
010727 *
010728 SD015     LDX      #0              ; MOVE MSG TO TEXT SCREEN
010729 SD020     LDA      MSG,X
010730          STA      MSGBASE-1,X
010731          INX
010732          CPX      #MSGLEN
010733          BNE      SD020
010734 *
010735          LDA      SDEATH.REGS+A.SAVE ; DISPLAY ERROR CODE (2 HEX DIGITS)
010736          CLC
010737          LSR      A
010738          LSR      A
010739          LSR      A
010740          LSR      A
010741          JSR      PRINT            ; FIRST DIGIT
010742          INX
010743          LDA      SDEATH.REGS+A.SAVE
010744          AND      #$0F
010745          JSR      PRINT            ; SECOND DIGIT
010746 *
010747          LDA      #>SD100
010748          STA      NMI.VECTOR
010749          LDA      #<SD100
010750          STA      NMI.VECTOR+1
010751 *
010752 *
010753          JMP      *                ; HANG UNTIL REBOOT (CTRL/RESET)
010754          REP      60
010755 SD100     RTI                    ; NMI VECTOR POINT HERE TO MASK THEM OUT
010756 *
010757 *
010758 * PRINT SUBROUTINE
010759 *
010760 PRINT      EQU      *
010761          CMP      #$A
010762          BCS      PRNT100
010763          ADC      #$30              ; "0"- "9"
010764          BCC      PRNT110          ; ALWAYS TAKEN
010765 PRNT100   ADC      #$36              ; "A"- "F"
010766 PRNT110   STA      MSGBASE-1,X
010767          RTS
010768 *

```

```
010769          LST      ON
010770  ZZEND      EQU      *
010771  ZZLEN      EQU      ZZEND-ZZORG
010772          IFNE      ZZLEN-LENSERR
010773          FAIL      2,"SOSORG          FILE IS INCORRECT FOR SYSERR"
010774          FIN
010775
010776 *****
010777 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SYSERR.SRC
010778 *****
010779
010780
```

```

010781 =====
010782 DOCUMENT :SOS1.3.3of5.THREE:SOS.ALLOC.TEXT
010783 =====
010784
010785 *****
010786 * APPLE /// SOS 1.3 SOURCE CODE FILE: ALLOC
010787 *****
010788 * ASSEMBLER: APPLE |[ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
010789
010790 *
010791 DEALLOC      STX      BMCNT      ; SAVE HIGH ORDER ADDRESS OF BLOCK TO BE FREED.
010792              PHA              ; SAVE IT
010793              LDX      VCBPTR     ; WHILE THE BITMAP
010794              LDA      VCB+VCBTLK+1,X ; DISK ADDRESS IS CHECKED
010795              CMP      BMCNT     ; TO SEE IF IT MAKES SENSE
010796              PLA              ; RESTORE
010797              BCC      DEALERR1  ; BRANCH IF IMPOSSIBLE
010798              TAX
010799              AND      #$7       ; GET THE BIT TO BE OR-ED IN.
010800              TAY
010801              LDA      WHICHBIT,Y ; (SHIFTING TAKES 7 BYTES, BUT IS SLOWER)
010802              STA      NOFREE    ; SAVE BIT PATTERN
010803              TXA              ; GET LOW BLOCK ADDRESS AGAIN.
010804              LSR      BMCNT
010805              ROR      A         ; GET POINTER TO BYTE IN BITMAP THAT REPRESENTS
010806              LSR      BMCNT     ; THE BLOCK ADDRESS.
010807              ROR      A
010808              LSR      BMCNT
010809              ROR      A
010810              STA      BMPTR     ; SAVE POINTER.
010811              LSR      BMCNT     ; NOW TRANSFER BIT WHICH SPECIFIES WHICH PAGE OF BITMAP.
010812              ROL      HALF
010813              LDX      BMTAB     ; (THIS POINTS TO THE TABLE FOR THE BITMAP BUFFER USED).
010814              LDA      BMAPMAP,X ; WHAT IS THE CURRENT MAP
010815              CMP      BMCNT     ; IS IN CORE BIT MAP THE ONE WE WANT?
010816              BEQ      DEALL1    ; BRANCH IF IN-CORE IS CORRECT.
010817              JSR      BMAPUP    ; PUT CURRENT MAP AWAY.
010818              BCS      DEALERR    ; PASS BACK ANY ERROR.
010819              LDA      BMCNT     ; GET DESIRED MAP NUMBER.
010820              LDY      #VCBCMAP
010821              STA      (VCBPTR),Y ; AND MAKE IT CURRENT.
010822              LDX      BMTAB
010823              LDA      BMADEV,X
010824              JSR      GTBMAP    ; READ IT INTO THE BUFFER,
010825              BCS      DEALERR
010826              LDY      BMPTR     ; INDEX TO BYTE.
010827              LSR      HALF
010828              BCC      DEALL2     ; BRANCH IF ON PAGE ONE OF BITMAP.
010829              INC      BMADR+1

```

```

010830 DEALL2      LDA      NOFREE      ; THE INDIVIDUAL BIT.
010831             ORA      (BMADR),Y
010832             STA      (BMADR),Y
010833             BCC      DEALL3      ; BRANCH IF ADDRESS IS PROPER
010834             DEC      BMADR+1
010835 DEALL3      LD      BMTAB      ; MARK BITMAP AS MODIFIED.
010836             LDA      #$80
010837             ORA      BMASTAT,X
010838             STA      BMASTAT,X
010839             CLC
010840 DEALERR      RTS
010841 DEALERR1     LDA      #BITMAPADR ; BIT MAP BLOCK NUMBER IMPOSSIBLE
010842             SEC      ; SAY BIT MAP DISK ADDRESS WRONG
010843             RTS      ; (PROBABLY DATA MASQUERADING AS INDEX BLOCK)
010844 *
010845 WHICHBIT     DFB      $80,$40,$20,$10
010846             DFB      8,4,2,1
010847 *
010848 *
010849             PAGE
010850 *
010851 ALCIDXS       LDA      #0          ; ALLOCATION OF THE INDEXES ALWAYS FILLS IN
010852             STA      SAPTR        ; STARTING AT THE BEGINNING OF THE BLOCK.
010853             JSR      ALC1BLK     ; THIS GETS FIRST INDEX AND SETS UP A
010854             BCS      ERRALC1     ; POINTER TO THE FREE BLOCKS (TO AVOID
010855 ALIDX1        LD      SAPTR      ; SCANNING THE WHOLE BLOCK EVERY TIME).
010856             STA      (TINDX),Y   ; SAVE INDEX BLOCK ADDRESS (LOW)
010857             INC      TINDX+1
010858             LDA      SCRCH+1     ; GET HIGH BYTE OF ADDRESS
010859             STA      (TINDX),Y   ; (AND SAVE IT)
010860             DEC      TINDX+1
010861             DEC      REQL        ; HAS REQUEST BEEN SATISFIED?
010862             BEQ      ALDXEND     ; (CARRY IS CLEAR)
010863             INC      SAPTR      ; BUMP INDEX POINTER
010864             LD      BMPTR      ; GET INDEX POINTER TO LAST ACCESSED BIT GROUP
010865             LDA      HALF        ; WHICH HALF OF MAP? (BOTH BMPTR & HALF SET UP BY 'ALC1BLK')
010866             BNE      SECNDHAF
010867             JSR      GETBITS1    ; GET NEXT FREE BLOCK ADDRESS.
010868             BCC      ALIDX1     ; BRANCH IF NO ERROR
010869 ERRALC1       RTS
010870 *
010871 SECNDHAF      JSR      GETBITS2    ; GET NEXT FREE BLOCK ADDRESS FROM SECOND HALF OF BIT MAP
010872             BCC      ALIDX1     ; BRANCH IF NO ERROR.
010873 ALDXEND       RTS      ; RETURN STATUS (CARRY SET INDICATES ERROR)
010874 *
010875 *
010876 ALC1BLK       JSR      FNDBMAP    ; GET ADDRESS OF BIT MAP IN 'BMADR'
010877             BCS      ERRALC1     ; BRANCH IF ERROR ENCOUNTERED
010878 SRCHFRE       LD      #0          ; START SEARCH AT BEGINNING OF BIT MAP BLOCK
010879             ST      HALF        ; INDICATE WHICH HALF (PAGE) WE'RE SEARCHING.

```

```

010880 GETBITS1    LDA      (BMADR),Y
010881             BNE      BITFOUND      ; FREE BLOCKS ARE INDICATED BY 'ON' BITS
010882             INY
010883             BNE      GETBITS1      ; CHECK ALL OF 'EM IN FIRST PAGE.
010884             INC      BMADR+1      ; BUMP HIGH ADDRESS OF CURRENT BITMAP
010885             INC      HALF          ; INDICATE SEARCH HAS PROGRESSED TO PAGE 2
010886             INC      BASVAL       ; BASE VALUE= BASE ADDRESS/2048
010887 GETBITS2    LDA      (BMADR),Y    ; SEARCH SECOND HALF FOR FREE BLOCK
010888             BNE      BITFOUND
010889             INY
010890             BNE      GETBITS2
010891             DEC      BMADR+1      ; RESET BIT MAP ADDRESS TO BEGINNING.
010892             INC      BASVAL       ; ADD 2048 OFFSET FOR NEXT PAGE
010893             JSR      NXTBMAP      ; GET NEXT BITMAP (IF IT EXISTS) AND UPDATE VCB.
010894             BCC      SRCHFRE      ; BRANCH IF NO ERROR ENCOUNTERED.
010895             RTS                    ; RETURN ERROR.
010896             PAGE
010897 *
010898 BITFOUND      STY      BMPTR       ; SAVE INDX POINTER TO VALID BIT GROUP
010899             LDA      BASVAL       ; SET UP FOR BLOCK ADDRESS CALCULATION
010900             STA      SCRTCH+1
010901             TYA
010902             ASL      A              ; GET ADDRESS OF BIT PATTERN
010903             ROL      SCRTCH+1
010904             ASL      A              ; MULTIPLY THIS AND BASVAL BY 8
010905             ROL      SCRTCH+1
010906             ASL      A
010907             ROL      SCRTCH+1
010908             TAX                    ; NOW X= LOW ADDRESS WITHIN 7 OF ACTUAL ADDRESS.
010909             LDA      (BMADR),Y    ; GET BIT PATTERN AGAIN
010910             SEC                    ; MARK RIGHT END OF BYTE.
010911 ADCALC      ROL      A              ; FIND LEFT MOST 'ON' BIT
010912             BCS      BOUNCE       ; BRANCH IF FOUND.
010913             INX                    ; ADJUST LOW ADDRESS
010914             BNE      ADCALC        ; BRANCH ALWAYS
010915 BOUNCE       LSR      A              ; RESTORE ALL BUT LEFT MOST BIT TO ORIGINAL POSITION
010916             BCC      BOUNCE       ; LOOP UNTIL MARK (SET ABOVE) MOVES INTO CARRY
010917             STA      (BMADR),Y    ; UPDATE BITMAP TO SHOW ALLOCATED BLOCK IN USE.
010918             STX      SCRTCH        ; SAVE LOW ADDRESS.
010919             LDX      BMTAB        ; UPDATE BIT MAP BUFFER STATUS
010920             LDA      #$80          ; INDICATE MAP HAS BEEN MODIFIED
010921             ORA      BMASTAT,X     ; (X IS EITHER 0 OR 6 FOR
010922             STA      BMASTAT,X     ; BUFFER 'A' OR 'B' RESPECTIVELY.)
010923             LDY      #VCBTFRE     ; SUBTRACT 1 FROM TOTAL FREE
010924             LDA      (VCBPTR),Y   ; BLOCKS IN VCB TO ACCOUNT FOR NEWLY
010925             SBC      #1            ; ALLOCATED BLOCK (CARRY IS SET FROM 'BOUNCE')
010926             STA      (VCBPTR),Y
010927             BCS      RET1BLK      ; BRANCH IF HI FREE COUNT DOESN'T NEED ADJUSTMENT.
010928             INY
010929             LDA      (VCBPTR),Y   ; ADJUST HIGH COUNT.

```

```

010930          SBC          #0          ; (CARRY IS CLEAR, SO ACC=ACC-1)
010931          STA          (VCBPTR),Y
010932  RET1BLK          CLC          ; INDICATE NO ERROR ENCOUNTERED
010933          LDA          SCRTCH          ; GET ADDRESS LOW IN ACC.
010934          LDY          SCRTCH+1          ; AND HIGH ADDRESS IN Y
010935          RTS          ; RETURN ADDRESS OF NEWLY ALLOCATED BLOCK.
010936  *
010937          PAGE
010938  *
010939  GTTINDX          LDY          #VCBDEV          ; GET DEVICE NUMBER SO WE DON'T
010940          LDX          #0          ; ANTICIPATE USING BUFFER 'A'.
010941          LDA          (VCBPTR),Y          ; USE THE BUFFER USED BY IT!
010942          CMP          BMADEV          ; IS IT IN BUFFER 'A'?
010943          BEQ          FREEBE          ; IF SO, FREE 'B'!
010944          CMP          BMBDEV          ; IF NOT, IS IT IN 'B'?
010945          BEQ          FREEA          ; IF SO, FREE UP BUFFER 'A'
010946          JSR          FNDBMAP          ; OTHERWISE, FORCE ALLOCATION FOR ONE OF THE BUFFERS
010947          BCC          GTTINDX          ; NOW TRY AGAIN.
010948          RTS          ; RETURN ERROR.
010949  *
010950  FREEBE          LDX          #BMTABSZ          ; DE-ALLOCATE BUFFER IF NECESSARY
010951  FREEA          STX          NOFREE          ; SAVE WHICH BUFFER WE'RE LOOKIN AT.
010952          LDY          BMASTAT,X          ; DO WE NEED TO WRITE BUFFER TO FREE IT?
010953          BPL          USEBUF          ; NO, THEN USE IT.
010954          STX          ZPGTEMP          ; SAVE BM BUFFER ID FOR A BIT
010955          JSR          WRITBM          ; WRITE BM TO OWNING UNIT
010956          BCS          SOMERR1          ; RETURN ANY ERROR (W/O RELEASING BM)
010957          LDX          ZPGTEMP          ; FETCH THE BM BUFFER ID
010958          LDA          #0
010959          STA          BMASTAT,X          ; AND MARK BM BUFFER AS FREE
010960  USEBUF          LDX          NOFREE          ; GET INDEX TO BUFFER INFO
010961          LDA          #0          ; MARK STATUS OF BUFFER AS FREE.
010962          STA          BMADEV,X          ; (DEVICE 0 IS NOT ANY DEVICE)
010963          STA          TINDX
010964          STA          BMADR
010965          LDA          BMAMADR,X          ; GET MEMORY ADDRESS OF FREE BUFFER.
010966          STA          TINDX+1
010967          TXA          ; SET UP PROPER HI ADDRESS OF BIT MAP TOO...
010968          EOR          #BMTABSZ          ; SELECT ALTERNATE BIT MAP TABLE.
010969          STA          BMTAB          ; (TO INDICATE WHICH IS BITMAP)
010970          TAX
010971          LDA          BMAMADR,X          ; GET HIGH ADDRESS OF BIT MAP.
010972          STA          BMADR+1
010973          LDA          BMBUFBNK          ; AND BANK PAIR NUMBER.
010974          STA          SSTIDXH
010975          STA          SISBMADR
010976          CLC          ; INDICATE NO ERRORS
010977  SOMERR1          RTS
010978  *
010979          PAGE

```

```

010980 NXTBMAP      LDY      #VCBTLK+1      ; BEFORE BUMPING TO NEXT MAP,
010981             LDA      (VCBPTR),Y      ; CHECK TO BE SURE THERE IS
010982             LSR      A                ; INDEED A NEXT MAP!
010983             LSR      A
010984             LSR      A
010985             LSR      A
010986             LDY      #VCBCMAP
010987             CMP      (VCBPTR),Y      ; ARE THERE MORE MAPS?
010988             BEQ      NOMORBIT        ; BRANCH IF NO MORE TO LOOK AT.
010989             LDA      (VCBPTR),Y      ; ADD 1 TO CURRENT MAP
010990             CLC
010991             ADC      #1
010992             STA      (VCBPTR),Y
010993             LDY      #VCBDEV
010994             LDA      (VCBPTR),Y
010995             TAX
010996             JSR      UPBMAP
010997             JMP      FNDBMAP        ; READ NEXT BIT MAP INTO BUFFER
010998 *
010999 GETA.BUF      LDX      #0
011000             BEQ      FRESHMAP
011001 *
011002 GETB.BUF      LDX      #BMTABSZ
011003             BNE      FRESHMAP        ; BRANCH ALWAYS
011004 *
011005 *
011006 FNDBMAP      LDY      #VCBDEV        ; GET DEVICE NUMBER
011007             LDA      (VCBPTR),Y
011008             LDX      #0                ; START WITH MAP 'A'
011009 FNDMAP1      CMP      BMADEV,X
011010             BNE      TRYMAP2
011011 FRESHMAP     STX      BMTAB          ; SAVE POINTER TO BIT MAP INFO TABLE
011012             LDY      BMASTAT,X        ; IS THIS ONE ALREADY MODIFIED?
011013             BMI      BMFOUND          ; YES, RETURN POINTER IN 'BMADR'
011014             JSR      GTBMAP          ; OTHERWISE READ IN FRESH BIT MAP
011015             BCC      BMFOUND          ; BRANCH IF SUCCESSFUL.
011016             RTS                     ; OTHERWISE, RETURN ERROR.
011017 *
011018 TRYMAP2      DEX
011019             BPL      FRBMBUF          ; NO, MUST FREE UP ONE OF THE BUFFERS
011020             LDX      #BMTABSZ        ; TRY BIT MAP BUFFER 'B'.
011021             JMP      FNDMAP1
011022             PAGE
011023 *
011024 BMFOUND      LDX      BMTAB            ; WHICH TABLE?
011025             LDY      #VCBCMAP
011026             LDA      (VCBPTR),Y
011027             ASL      A
011028             STA      BASVAL
011029             LDA      BMAMADR,X        ; GET HIGH ADDRESS

```

```

011030          STA          BMADR+1
011031          LDA          BMBUFBNK          ; GET BANK NUMBER OF BUFFER BIT MAP BUFFERS
011032          STA          SISBMADR
011033          LDA          #0                ; BUFFERS ALWAYS FALL ON A PAGE BOUNDARY
011034          STA          BMADR
011035          CLC
011036          RTS                ; INDICATE ALL IS VALID AND GOOD!
011037          *
011038  NOMORBIT          LDA          #OVRERR          ; INDICATE REQUEST CAN'T BE FILLED.
011039          SEC
011040          RTS                ; INDICATE ERROR
011041          *
011042  FRBMBUF          SEC
011043          LDX          BMTAB                ; FIND OUT WHICH WAS LAST USED.
011044          BEQ          CHKBMB                ; IF 'A' WAS USED CHECK 'B' FIRST
011045          CLC
011046          BIT          BMASTAT                ; IS BUFFER 'A' FREE (UNMODIFIED)?
011047          BPL          GETA.BUF                ; YES, USE IT.
011048  CHKBMB          BIT          BMBSTAT                ; IS BUFFER 'B' FREE?
011049          BCC          FREBUF1                ; BRANCH IF BOTH ARE USED
011050          BPL          GETB.BUF                ; YES...
011051          BIT          BMASTAT                ; (CHECK 'A')
011052          BPL          GETA.BUF
011053  FREBUF1          LDX          #0
011054          BCC          FREBUFA                ; BRANCH IF BUFFER 'A' HAS LEAST PRIORITY.
011055          LDX          #BMTABSZ
011056  FREBUFA          STX          ZPGTEMP                ; SAVE BM BUFF ID FOR A BIT
011057          JSR          WRBMAP                ; XREG PASSES BM BUFF ID
011058          BCS          NOGO                ; ERROR ENCOUNTERED ON WRITING
011059          LDX          ZPGTEMP                ; FETCH BM BUFF ID
011060          LDA          #0
011061          STA          BMASTAT,X                ; AND MARK BM BUFFER AS FREE
011062          BCC          FNDBMAP                ; LOOK AGAIN FOR FRRE BIT MAP BUFFER SPACE
011063  NOGO            RTS                ; RETURN ERROR ON WRITING BM
011064          *
011065  UPBMAP            CPX          BMADEV                ; UPDATE BIT MAP OF DEVICE X
011066          BNE          UPBM1
011067          CLC
011068          BIT          BMASTAT
011069          BMI          FREBUF1                ; (CARRY CLEAR FOR BUFFER 'A')
011070          RTS
011071          PAGE
011072          *
011073  UPBM1            CPX          BMBDEV
011074          BNE          NOUPDAT                ; DON'T UPDATE IF NOT NECESSARY.
011075          BIT          BMBSTAT
011076          BMI          FREBUF1                ; (CARRY IS SET)
011077  NOUPDAT          CLC
011078          RTS                ; RETURN 'NO ERROR'
011079          *

```



```

011080 CLEARBMS      EQU      *                ; MAKE SURE ALL BIT MAPS ASSOCIATED
011081 * WITH A DEVICE ARE MARKED INVALID
011082 * IF A NEW VOLUME IS LOGGED IN ON IT.
011083 * INPUT ARG: A REG = DEVNUM
011084 * X REG PRESERVED
011085             LDY      #0
011086             CMP      BMADEV
011087             BNE      CLRBM1              ; BRANCH IF BIT MAP A NOT OWNED
011088             BIT      BMASTAT
011089             BMI      CLRBM2              ; BRANCH IF BITMAP A BUSY
011090             STY      BMADEV              ; ELSE, CLEAR IT
011091 CLRBM2         RTS
011092 CLRBM1         CMP      BMBDEV
011093             BNE      CLRBM2              ; BRANCH IF BIT MAP B NOT OWNED BY DEVNUM
011094             BIT      BMBSTAT
011095             BMI      CLRBM2              ; BRANCH IF BITMAP B BUSY
011096             STY      BMBDEV              ; ELSE CLEAR IT
011097             RTS
011098 *
011099 GTBMAP         STA      BMADEV,X         ; SAVE ACC AS CURRENT DEVICE FOR BUFFER
011100             LDA      BMAMADR,X         ; GET HIGH ORDER ADDRESS OF BUFFER
011101             STA      BMADR+1           ; SELECTED BY X
011102             LDA      BMBUFBNK         ; AND GET BANK PAIR NUMBER
011103             STA      SISBMADR          ; OF BOTH BIT MAP BUFFERS 'A' AND 'B'
011104             LDY      #VCBCMAP         ; GET LOWEST MAP NUMBER WITH FREE BLOCKS IN IT.
011105             LDA      (VCBPTR),Y
011106             STA      BMACMAP,X         ; ASSOCIATE THE OFFSET WITH THE BITMAP CONTROL BLOCK
011107             CLC
011108             LDY      #VCBDMAP         ; ADD THIS NUMBER TO THE BASE
011109             ADC      (VCBPTR),Y         ; ADDRESS OF FIRST BIT MAP
011110             STA      BMADADR,X         ; SAVE LOW ADDRESS OF BIT MAP TO BE USED.
011111             INY
011112             LDA      (VCBPTR),Y         ; NOW GET HIGH DISK ADDRESS OF MAP
011113             ADC      #0                ; ADD TO THIS THE STATE OF THE CARRY
011114             STA      BMADADR+1,X       ; SAVE HIGH DISK ADDRESS TOO.
011115 ; DROP INTO 'RDBMAP'
011116 *
011117             PAGE
011118 *
011119             LDA      #RDCMD            ; (X CONTAINS AN INDEX TO DETERMINE WHICH BUFFER)
011120 DOBMAP        STA      DHPCMD         ; SAVE DEVICE COMMAND
011121             LDA      DEVNUM            ; FIX THE 'BIT MAP TRASH BUG'
011122             PHA                        ; BY NOT MUNGING DEVNUM
011123             LDA      BMADEV,X         ; GET DEVICE NUMBER.
011124             STA      DEVNUM
011125             LDA      BMADADR,X         ; AND MAP'S DISK ADDRESS
011126             STA      BLOKNML
011127             LDA      BMADADR+1,X
011128             STA      BLOKNMH
011129             LDA      BMAMADR,X         ; LASTLY GET THE ADDRESS OF THE BUFFER

```

```

011130          LDX      BMBUFBNK          ; AND BANK NUMBER.
011131          JSR      DOBITMAP         ; (NOTE: LOW ADDRESS IS FIXED TO ZERO AS THIS IS A BUFFER)
011132          PLA          ; RESTORE
011133          STA      DEVNUM          ; THE DEVNUM WE CAME IN WITH!
011134          RTS
011135          *
011136 WRTBMAP    LDA      #WRTCMD        ; WRITE BIT MAP POINTED TO BY X
011137          JMP      DOBMAP
011138          *
011139 WRTGBUF    LDA      #WRTCMD        ; SET CALL FOR WRITE.
011140          BNE      SVGCMD          ; BRANCH ALWAYS.
011141 RDGBUF     LDA      #RDCMD        ; SET CALL FOR READ.
011142 SVGCMD     STA      DHPCMD        ; PASSED TO DEVICE HANDLER.
011143          LDA      BLOKNML         ; SAVE CURRENT
011144          STA      TTLINK          ; GBUF BLOCK
011145          LDA      BLOKNMH         ; ADDRESS
011146          STA      TTLINK+1        ; FOR DIRECTORY EXTEND
011147          LDA      #GBUF/256      ; GET HIGH ADDRESS OF GENERAL BUFFER
011148          LDX      #0              ; TO FORCE ACCESS TO NON BANK MEMORY.
011149          EQU      *
011150          DOIDX    STA      DBUFPH
011151          STX      SISBPH          ; SELECT BANK
011152          LDA      #0              ; GENERAL PURPOSE BUFFERS ALWAYS
011153          STA      DBUFP          ; START ON A PAGE BOUNDARY.
011154          JMP      FILEIO2         ; END VIA DEVICE DISPATCHER.
011155          *
011156          TTLINK  DS      2          ; GBUF CURRENT ADDRESS
011157          *
011158          WRTINDX  LDA      #WRTCMD
011159          LDX      IDXADRL         ; GET BLOCK ADDRESS OF INDEX BLOCK
011160          LDY      IDXADRH
011161          DOFRST   STA      DHPCMD        ; (ENTRY USED BY RD/WRTDFRST)
011162          STX      BLOKNML
011163          STY      BLOKNMH
011164          LDA      TINDX+1         ; HIGH RAM ADDRESS OF INDEX BLOCK
011165          LDX      SSTIDXH         ; AND BANK NUMBER.
011166          JMP      DOIDX          ; AND GO DO REQUESTED OPERATION.
011167          *
011168          WRTDFRST  LDA      #WRTCMD        ; WRITE FILE'S FIRST BLOCK (USED
011169          BNE      FADDR          ; BY CREATE, SO ADDRESS IN 'D.' STUFF).
011170          RDFRST   LDA      #RDCMD
011171          FADDR    LDX      DFIL+D.FRST ; (BUFFER ADDRESS IS IN 'TINDX')
011172          LDY      DFIL+D.FRST+1
011173          JMP      DOFRST
011174          *
011175          *
011176          CHN      POSN/OPEN,4,2
011177
011178          *****
011179          * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: ALLOC

```

011180 \*\*\*\*\*  
011181  
011182

```

011183 =====
011184 DOCUMENT :SOS1.3.3of5.THREE:SOS.CREATE.TEXT
011185 =====
011186
011187 *****
011188 * APPLE /// SOS 1.3 SOURCE CODE FILE: CREATE
011189 *****
011190 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
011191
011192 PAGE
011193 CREATE EQU *
011194 INC CFLAG ; SAY WE ARE IN CREATE (DIR EXTEND)
011195 JSR LOOKFILE ; CHECK FOR DUPLICATE / GET FREE ENTRY
011196 BCS TSTFNF ; ERROR CODE IN ACC MAY BE 'FILE NOT FOUND'
011197 LDA #DUPEER ; TELL EM A FILE OF THAT NAME ALREADY EXISTS
011198 CRERR1 SEC ; INDICATE ERROR ENCOUNTERED
011199 RTS ; RETURN ERROR IN ACC.
011200 *
011201 TSTFNF CMP #FNFERR ; 'FILE NOT FOUND' IS WHAT WE WANT
011202 BNE CRERR1 ; PASS BACK OTHER ERROR.
011203 LDA NOFREE ; TEST FOR DIRECTORY SPACE
011204 BNE CREAT1 ; BRANCH IF VALID FREE ENTRY WAS FOUND.
011205 LDA #DIRFULL ; RETURN DIRECTORY FULL ERROR
011206 SEC
011207 RTS
011208 *
011209 CREAT1 LDY #$9 ; SET UP DEFAULT PARAMETERS FOR CREATE
011210 LDA #0 ; IN THE SPACE DIRECTLY FOLLOWING THE
011211 ZERCALL STA C.FILID,Y ; CALL SPECIFICATION AND THEN
011212 DEY ; CHECK FOR ADDITIONAL PARAMETERS FROM
011213 BPL ZERCALL ; USER'S CALL SPEC VIA 'C.CLIST'
011214 LDA #SEEDTYP ; DEFAULT TYPE IS 'SEED' TREE INDEX
011215 STA C.STOR
011216 LDY C.XLEN ; GET THE LENGTH OF THE CALL XTENSION LIST
011217 BEQ CRENAM ; IF ZERO THEN USE DEFAULTS
011218 DEY ; (SINCE THE POINTER IS AT BYTE 0)
011219 CPY #$9 ; MAKE SURE WE DON'T HAVE TOO MANY PARAMETERS
011220 BCC MOVPARM ; MOVE 'EM IF REASONABLE COUNT.
011221 LDA #BADLSTCNT ; INVALID LIST COUNT
011222 RTS ; RETURN ERROR.
011223 *
011224 MOVPARM LDA (C.XLIST),Y ; MOVE IN THE USER SPECIFIED
011225 STA C.FILID,Y ; PARAMETERS. VALIDITY IS CHECKED
011226 DEY ; AT VARIOUS POINTS FURTHER ALONG IN
011227 BPL MOVPARM ; THIS PROCESS.
011228 CRENAM LDY #0 ; MOVE LOCAL FILE NAME TO ENTRY BUFFER.
011229 LDA (PATHNML),Y ; GET LENGTH OF LOCAL NAME
011230 TAY
011231 CRENAM1 LDA (PATHNML),Y

```

```

011232      STA      DFIL+D.STOR,Y
011233      DEY
011234      BPL      CRENAM1          ; (MOVE ALL, INCLUDING LENGTH BYTE.)
011235      LDA      C.FILID          ; MOVE FILE AND AUX ID.
011236      STA      DFIL+D.FILID
011237      LDA      C.AUXID
011238      STA      DFIL+D.AUXID
011239      LDA      C.AUXID+1
011240      STA      DFIL+D.AUXID+1
011241      LDA      #READEN+WRITEN+RENAMEN+DSTROYEN
011242      STA      DFIL+D.ATTR
011243      LDA      D.HEAD          ; SAVE FILE'S HEADER ADDRESS TOO.
011244      STA      DFIL+D.DHDR
011245      LDA      D.HEAD+1
011246      STA      DFIL+D.DHDR+1
011247      JSR      TWRPROT1        ; CAN WE WRITE TO THIS DISKETTE?
011248      BCS      CRERR1
011249      LDA      C.STOR          ; NOW TEST STORAGE TYPE FOR TREE TYPE FILES
011250      CMP      #4              ; NOTE: THIS IS HARD CODED SINCE ALL TREES ARE LESS THAN 4 *****
011251      BCC      SEED            ; BRANCH IF SOME TYPE OF TREE (SEED, SAPLING...)
011252      JMP      NOTREE          ; GO TEST FOR SOME OTHER TYPE (SUCH AS DIRECTORY).
011253      PAGE
011254      *
011255 SEED      LDX      #SEEDTYP    ; START OUT ASSUMING A SEED FILE
011256      LDA      C.EOFHH          ; TEST FOR OUT OF RANGE PREALLOCATION
011257      BEQ      SEED1          ; (HOPEFULLY BRANCH ALWAYS)
011258 OVFLOW    LDA      #OVRERR    ; REPORT UNABLE TO SATISFY REQUEST.
011259      SEC
011260      RTS
011261      *
011262 SEED1      LDA      C.EOFHL    ; CALCULATE THE NUMBER OF
011263      STA      DFIL+D.EOF+2      ; BLOCKS NEEDED FOR PRE-ALLOCATION
011264      LSR      A
011265      TAY          ; Y HOLDS THE NUMBER OF INDEX BLOCKS NEEDED
011266      STA      DATBLKH
011267      LDA      C.EOFLH          ; (CARRY UNDISTURBED FROM LAST SHIFT)
011268      STA      DFIL+D.EOF+1
011269      ROR      A
011270      STA      DATBLKL
011271      LDA      C.EOFLL
011272      STA      DFIL+D.EOF          ; (CARRY IN TACT FROM LOW COUNT)
011273      BNE      INCDATA          ; BUMP THE COUNT ON DATA BLOCKS IF REQUEST
011274      BCC      TSTSAP          ; IS NOT A MULTIPLE OF 512.
011275 INCDATA    INC      DATBLKL
011276      BNE      TSTSAP
011277      INY          ; MUST INCREASE NUMBER OF INDEXES ALSO.
011278      INC      DATBLKH
011279 TSTSAP      TYA
011280      BNE      SAPLING          ; IF NON ZERO, THEN IT'S AT LEAST A SAPLING.
011281      LDA      DATBLKL          ; TO QUALIFY AS AN HONEST SEED,

```

```

011282          BNE      TSTSEED          ; THEN ONE OR LESS DATA BLOCKS REQUESTED
011283          INC      DATBLKL          ; (MUST BE AT LEAST ONE BLOCK ALLOCATED
011284          BNE      CREALC          ; TYPE IS SEED. BRANCH ALWAYS
011285 TSTSEED   CMP      #1              ; IF GREATER THAN ONE, IT'S NOT A SEED.
011286          BEQ      CREALC          ; IT IS A SEED. CONTINUE CREATION
011287          INX      INX              ; THE TYPE IS SAPLING.
011288          INY      INY              ; ONE INDEX BLOCK IS NEEDED.
011289          BNE      CREALC          ; BRANCH ALWAYS
011290          PAGE
011291          *
011292 SAPLING   INX      INX              ; TYPE IS AT LEAST SAPLING.
011293          CMP      #1              ; NO MORE THAN ONE INDEX BLOCK FOR A SAPLING
011294          BNE      TREE              ;
011295          LDA      DATBLKL          ; MUST BE SURE THIS IS REAL MAX SAPLING (128K FILE)
011296          BEQ      CREALC          ; BRANCH IF IT IS.
011297 TREE     INY      INY              ; ACCOUNT FOR ADDITIONAL 2ND LEVEL INDEX
011298          *
011299          INX      INX              ; TYPE IS TREE (2 LEVEL INDEX)
011300          INY      INY              ; ADD AN EXTRA INDEX BLOCK FOR TOP INDEX
011301 CREALC   STY      INDXBLK          ; STORE INDEX BLOCK COUNT
011302          TXA      TXA              ; PUT STORAGE TYPE IN DIRECTORY ENTRY
011303          ASL      A
011304          ASL      A
011305          ASL      A
011306          ASL      A
011307          ORA      DFIL+D.STOR
011308          STA      DFIL+D.STOR
011309          STX      LEVELS           ; SAVE NUMBER OF INDEX LEVELS FOR PREALLOCATION.
011310          TYA      TYA              ; NOW FIGURE THE TOTAL NUMBER OF
011311          CLC      CLC              ; BLOCKS NEEDED (DATA + INDEX BLOCKS)
011312          ADC      DATBLKL
011313          STA      DFIL+D.USAGE      ; (MIGHT AS WELL RECORD IT IN DIR
011314          STA      REQL             ; WHILE WE'RE AT IT.)
011315          LDA      DATBLKH
011316          ADC      #0              ; UPDATE HI BYTE TOO
011317          STA      DFIL+D.USAGE+1
011318          STA      REQH
011319          LDX      D.DEV            ; PASS ALONG THE DEVICE WE'RE TALKIN ABOUT.
011320          JSR      TSFRBLK          ; 'TEST FREE BLOCKS' FINDS OUT IF ENOUGH FREE SPACE EXISTS
011321          BCS      OVFLOW          ; BRANCH IF NOT ENOUGH SPACE.
011322          JSR      ALC1BLK         ; GO ALLOCATE FIRST BLOCK
011323          BCS      CRERR
011324          STA      DFIL+D.FRST      ; (RETURNS ACC=LOW Y=HIGH)
011325          STA      IDXADR           ; SAVE AS ADDRESS FOR INCORE INDEX ALSO.
011326          STY      DFIL+D.FRST+1
011327          STY      IDXADRH
011328          JSR      ZERGBUF          ; GO CLEAN OUT GBUF
011329          JSR      GTTINDX         ; GET TEMPORARY SPACE FOR AN INDEX BLOCK
011330          JSR      ZTMPIDX         ; AND ZERO IT OUT.
011331          LDX      LEVELS

```

```

011332          DEX          ; TEST FOR NUMBER OF LEVELS NEEDED.
011333          BEQ          ENDCRE          ; BRANCH IF SEED FILE.
011334          DEX          ; IS IT A SAPLING PRE-ALLOCATION.
011335          BEQ          SAPFILE
011336          LDY          INDXBLK          ; LOAD NUMBER OF INDEX BLOCKS NEEDED
011337          DEY          ; REMOVE THE ONE JUST ALLOCATED.
011338          STY          REQL
011339          STY          INDXBLK
011340          JSR          ALCIDXS          ; GO ALLOCATE INDEXES FOR LOWER INDEX BLOCKS.
011341          BCS          CRERR
011342          JSR          WRTDFRST          ; GO WRITE TREE TOP INDEX BLOCK.
011343          BCS          CRERR          ; BRANCH IF UNABLE TO DO THIS.
011344          LDA          #0          ; INIT INDEX POINTER
011345          STA          TREPTR
011346          PAGE
011347  FILLTREE  LDY          TREPTR
011348          LDA          (TINDX),Y          ; GET ADDRESS OF LOWER BLOCK
011349          STA          IDXADRL
011350          INC          TINDX+1          ; BUMP TO PAGE 2 TO GET HI ADDRESS.
011351          LDA          (TINDX),Y          ; GET HIGH ADDRESS.
011352          STA          IDXADRH
011353          DEC          TINDX+1          ; CLEAN UP AFTER SELF...
011354          DEC          INDXBLK          ; IS THIS THE LAST BLOCK ALLOCATED?
011355          BEQ          LSTSAP          ; YES, ALLOCATE PARTIAL FILLED INDEX BLOCK
011356          LDA          #0          ; ALLOCATE ALL 256 INDEXES
011357          STA          REQL
011358          JSR          SAPINDX          ; AND WRITE ZEROED DATA BLOCKS.
011359          BCS          CRERR          ; STOP IF ERROR ENCOUNTERED.
011360          JSR          WRTINDX          ; WRITE INDEX BLOCK
011361          BCS          CRERR          ; HOPEFULLY NEVER TAKEN.
011362          INC          TREPTR
011363          JSR          RDFRST          ; READ IN TOP INDEX AGAIN.
011364          BCC          FILLTREE          ; BRANCH IF NO ERROR.
011365  CRERR     SEC          ; JUST IN CASE IT WAS CLEAR.
011366          RTS          ; RETURN ERROR.
011367          *
011368          *
011369  SAPFILE   EQU          *
011370  LSTSAP    LDA          DATBLKL          ; GET NUMBER OF DATA BLOCKS (LOW BYTE) REQUESTED.
011371          STA          REQL
011372          JSR          SAPINDX          ; GO ALLOCATE DATA BLOCKS AND WRITE EM.
011373          BCS          CRERR
011374  ENDCRE    JSR          WRTINDX          ; GO WRITE INDEX BLOCK. (FOR SEED THIS IS DATA.)
011375          BCS          CRERR
011376          LDX          #3          ; MOVE CREATION TIME FOR THIS ENTRY
011377  TRETIME   LDA          DATELO,X
011378          STA          DFIL+D.CREDT,X
011379          DEX
011380          BPL          TRETIME
011381  ENDCRE0   INC          H.FCNT          ; ADD ONE TO TOTAL NUMBER OF FILES IN SPECIFIED DIRECTORY.

```

```

011382          BNE          ENDCRE1
011383          INC          H.FCNT+1
011384          LDX          #3                ; ENSURE MOD
011385 ENDCRX     LDA          DATELO,X        ; DATE/TIME
011386          STA          DFIL+D.MODDT,X    ; IS
011387          DEX          ; INITIALIZED
011388          BPL          ENDCRX
011389 ENDCRE1   LDX          D.DEV            ; UPDATE APPROPRIATE BIT MAP
011390          JSR          UPBMAP
011391          BCS          CRERR2            ; BRANCH ON BITMAP UPDATE ERR
011392          JSR          DREVERSE          ; UPDATE DIRECTORY LAST
011393          RTS          ; RETURN ERRORS OR OK RESULT
011394          *
011395          PAGE
011396 SAPINDX   JSR          ZTMPIDX          ; ZERO OUT ANY STUFF LEFT OVER.
011397          LDA          REQL              ; PRESERVE REQUEST COUNT
011398          STA          TLINK
011399          JSR          ALCIDXS            ; GO ALLOCATE REQUESTED NUMBER OF BLOCKS.
011400          BCS          CRERR
011401          LDY          #0                ; THEN WRITE ZEROS TO DATA BLOCKS.
011402          STY          SAPTR            ; USE AS POINTER TO INDEX BLOCK
011403          LDA          (TINDX),Y        ; GET DATA BLOCK ADDRESS (LOW BYTE).
011404          STA          BLOKNML
011405          INC          TINDX+1
011406          LDA          (TINDX),Y        ; GET HIGH ADDRESS OF PRE-ALLOCATED DATA BLOCK.
011407          STA          BLOKNMH
011408          DEC          TINDX+1          ; (RESET BUFFER ADDRESS)
011409          JSR          WRTGBUF          ; WRITE DATA BLOCK
011410          BCS          CRERR
011411          LDA          TLINK            ; GET NUMBER REQUESTED AGAIN
011412          STA          REQL
011413 DATINIT  LDY          SAPTR            ; GET POINTER TO INDEX BLOCK AGAIN.
011414          INY          ; ANTICIPATE DOIN' THE NEXT DATA BLOCK
011415          DEC          REQL              ; DO WE INDEED HAVE ANOTHER BLOCK TO WRITE.
011416          BEQ          DATDONE          ; NO, ALL DONE (CARRY CLEAR).
011417          STY          SAPTR            ; USE AS POINTER TO INDEX BLOCK
011418          LDA          (TINDX),Y        ; GET DATA BLOCK ADDRESS (LOW BYTE).
011419          STA          BLOKNML
011420          INC          TINDX+1          ; BUMP HI ADDR OF INDEX BUFFER TO ACCESS HIGH ADDR.
011421          TAX          ; WAS LOW ADDRESS A ZERO?
011422          BNE          DATIT1           ; IF NOT, NO NEED TO CHECK VALIDITH OF HI BYTE
011423          CMP          (TINDX),Y
011424          BNE          DATIT1           ; BOTH BYTES CAN'T BE ZERO.
011425          LDA          #ALCERR
011426          JSR          SYSDEATH
011427 DATIT1   LDA          (TINDX),Y        ; GET HIGH ADDRESS OF PRE-ALLOCATED DATA BLOCK.
011428          STA          BLOKNMH
011429          DEC          TINDX+1          ; (RESET BUFFER ADDRESS)
011430          LDA          #GBUF/256
011431          STA          DBUFPH           ; RESET TO ADDR TO GBUF JUST TO BE SURE.

```



```

011432          JSR      REPEATIO      ; WRITE DATA BLOCK
011433          BCC      DATINIT
011434 DATDONE   RTS                  ; RETURN STATUS (CARRY SET IF ERROR)
011435 *
011436 REPEATIO  EQU          *
011437          LDA      #RPTCMD
011438          STA      DHPCMD
011439          JMP      RPEATIO1
011440 *
011441 ZERGBUF   LDY      #0            ; ZERO OUT THE GENERAL PURPOSE BUFFER
011442          TYA
011443 ZGBUF     STA      GBUF,Y        ; WIPE OUT BOTH PAGES
011444          STA      GBUF+$100,Y    ; WITH SAME LOOP.
011445          INY
011446          BNE      ZGBUF
011447          RTS
011448 *
011449 *
011450 ZTMPIDX    LDY      #0            ; ZERO OUT TEMPORARY INDEX BLOCK
011451          TYA
011452 ZINDX1     STA      (TINDX),Y     ; THIS HAS TO BE DONE A
011453          INY                    ; TIME SINCE IT'S INDIRECT.
011454          BNE      ZINDX1
011455          INC      TINDX+1
011456 ZINDX2     STA      (TINDX),Y
011457          INY
011458          BNE      ZINDX2
011459          DEC      TINDX+1        ; RESTORE PROPER ADDRESS
011460 CRERR2    RTS
011461          PAGE
011462 NOTREE     CMP      #DIRTYP        ; IS A DIRECTORY TO BE CREATED?
011463          BEQ      ISDIR          ; YES, DO SO...
011464          JMP      NOTDIR        ; NO, TRY NEXT TYPE.
011465 *
011466 ISDIR      LDA      C.EOFHH        ; CAN'T CREATE A DIRECTORY LARGER THAN
011467          ORA      C.EOFHL        ; 127 BLOCKS (THAT'S HUGE!)
011468          BEQ      ISDIR1        ; BRANCH IF WITHIN LIMITS, OTHERWISE
011469 DIROVR     LDA      #OVRERR        ; REQUESTED DIRECTORY SIZE CAN'T BE
011470          SEC                    ; CREATED. SET CARRY TO INDICATE ERROR.
011471          RTS
011472 *
011473 ISDIR1     LDA      C.EOFLH        ; CALCULATE HOW MANY BLOCKS WILL
011474          LSR      A              ; BE NEEDED FOR THIS NEW DIRECTORY.
011475          TAY                    ; (SAVE INITIAL COUNT IN Y)
011476          LDA      C.EOFLL        ; IF REQUESTED EOF IS NOT AN EVEN BLOCK
011477          BNE      DADD1          ; SIZE, THEN ROUND UP.
011478          BCC      TSDIRSZ        ; BRANCH IF ROUNDING UNNECESSARY.
011479 DADD1      INY                    ; ADD ONE TO BLOCK COUNT.
011480 TSDIRSZ    TYA                    ; TEST TO BE SURE SIZE IS GREATER THAN ZERO
011481          BEQ      DADD1          ; IF ZERO THEN SIZE=1

```

```

011482      STA      DFIL+D.USAGE      ; SAVE NUMBER OF BLOCKS TO BE USED.
011483      STA      REQL
011484      ASL      A                  ; NOW SAVE ADJUSTED END OF FILE
011485      STA      DFIL+D.EOF+1
011486      LDA      #0
011487      STA      DFIL+D.EOF
011488      STA      DFIL+D.EOF+2
011489      STA      REQH              ; REQUESTED NUMBER OF BLOCKS NEVER EXCEEDS 128.
011490      JSR      TSFRBLK          ; TEST TO BE SURE ENOUGH DISK SPACE IS FREE.
011491      BCS      DIROVR          ; BRANCH IF REQUEST TOO LARGE.
011492      JSR      ZERGBUF         ; CLEAR CRAP FROM GBUF.
011493      JSR      ALC1BLK         ; GET ADDRESS OF FIRST (HEADER) BLOCK.
011494      BCS      CRERR2
011495      STA      DFIL+D.FRST
011496      STA      TLINK
011497      STY      DFIL+D.FRST+1
011498      STY      TLINK+1          ; (TLINK IS FOR REVERSE LINKAGE.)
011499      LDA      SOSTMPL         ; STORE SOS STAMP IN NEW DIRECTORY
011500      STA      GBUF
011501      LDA      SOSTMPH
011502      STA      GBUF+1
011503      LDY      #4              ; MOVE OTHER VARIOUS THINGS
011504      BNE      DRSTUF1         ; BRANCH ALWAYS
011505      DRSTUF  LDA      D.ENTBLK,Y ; MOVE OWNING ENTRY'S
011506      STA      GBUF+HRBLK+4,Y ; BLOCK ADDRESSES AND NUMBER TO NEW HEADER.
011507      DRSTUF1 LDA      SOSVER,Y ; MOVE VERSION, COMPATABILITY,
011508      STA      GBUF+HVER+4,Y   ; ATTRIBUTES, AND ENTRY SIZE
011509      DEY
011510      BPL      DRSTUF
011511      LDA      H.ENTLN          ; OVER WRITE LAST BYTE MOVED IN ABOVE LOOP WITH
011512      STA      GBUF+HRELN+4    ; THE PARENT DIRECTORY ENTRY LENGTH.
011513      LDA      DFIL+D.STOR     ; SET HEADER TYPE AND NAME
011514      TAY
011515      ORA      #HEDTYP*16
011516      STA      GBUF+HNLEN+4
011517      TYA                      ; (AND WHILE WE'RE AT IT SET DIRECTORY TYPE)
011518      ORA      #DIRTYP*16
011519      STA      DFIL+D.STOR
011520      *
011521      MVHNAME LDA      DFIL+D.STOR,Y
011522      STA      GBUF+HNLEN+4,Y   ; MOVE HEADER NAME
011523      DEY
011524      BNE      MVHNAME
011525      LDX      #3              ; GET CURRENT DATE.
011526      CRETIME LDA      DATELO,X
011527      STA      GBUF+HCRDT+4,X  ; SAVE AS HEADER CREATION TIME
011528      STA      DFIL+D.CREDT,X  ; AND DATE OF FILE CREATE.
011529      DEX
011530      BPL      CRETIME
011531      LDA      #$76

```

```

011532          STA      GBUF+HPENAB+4      ; DUMMY PASSWORD
011533          DEC      REQL                ; TEST FOR ONE BLOCK DIRECTORY
011534          BEQ      DIRCREND           ; IT IS, FINISH UP.
011535          JSR      DIRWRT             ; GO WRITE FIRST DIRECTORY BLOCK AND ALLOCATE NEXT
011536          BCS      DERROR             ; PASS BACK ERROR.
011537          JSR      ZERGBUF           ; CLEAN OUT GENERAL BUFFER AGAIN.
011538  CRNXDIR  LDA      TLINK             ; MOVE LAST BLOCK ADDRESS
011539          STA      GBUF              ; AS BACKWARD LINK.
011540          LDA      TLINK+1
011541          STA      GBUF+1
011542          LDA      FLINK              ; MAKE FORWARD LINK INTO CURRENT ADDRESS
011543          STA      TLINK
011544          LDA      FLINK+1
011545          STA      TLINK+1
011546          DEC      REQL              ; IS THIS THE LAST BLOCK?
011547          BEQ      DIRCREND
011548          JSR      DIRWRT             ; WRITE THIS BLOCK AND ALLOCATE NEXT.
011549          BCS      DERROR
011550          LDA      #0                 ; ZERO OUT FORWARD LINK
011551          STA      GBUF+2
011552          STA      GBUF+3
011553          BEQ      CRNXDIR           ; BRANCH ALWAYS
011554  *
011555  DIRCREND  JSR      DIRWRT1          ; WRITE LAST BLOCK OF THIS DIRECTORY
011556          BCS      DERROR
011557          JMP      ENDCRE0           ; FINISH UP WRITING OWNER DIRECTORY STUFF.
011558  *
011559  DIRWRT   JSR      ALC1BLK          ; GET ADDRESS OF NEXT BLOCK.
011560          BCS      DERROR
011561          STA      GBUF+2
011562          STY      GBUF+3            ; SAVE LINK ADDRESS
011563          STA      FLINK
011564          STY      FLINK+1
011565  DIRWRT1  LDA      TLINK             ; GET ADDRESS OF CURRENT BLOCK
011566          STA      BLOKNML
011567          LDA      TLINK+1
011568          STA      BLOKNMH
011569          JMP      WRTGBUF          ; GO WRITE IT OUT
011570          PAGE
011571  *
011572  ERRGBUF   EQU      *
011573  DERROR   RTS
011574  *
011575  *
011576  SOSTMPL   DFB      $0              ; THE FOLLOWING TWO BYTES ARE THE 'SOS STAMP'
011577  SOSTMPH   DFB      $0
011578  *
011579  SOSVER    DFB      0,0,0,$27,13
011580  *
011581  *

```

```

011582 RNDTAB      EQU      *
011583 ENTALCALC  LDA      #GBUF/256      ; SET HIGH ADDRESS OF DIRECTORY ENTRY INDEX POINTER
011584           STA      DRBUFPH
011585           LDA      #4              ; CALCULATE ADDRESS OF ENTRY BASED
011586           LDX      D.ENTNUM       ; ON THE ENTRY NUMBER
011587 ECALC0     CLC
011588 ECALC1     DEX      ; ADDR=GBUF+((ENTNUM-1)*ENTLEN)
011589           BEQ      ECALC2
011590           ADC      H.ENTLN
011591           BCC      ECALC1
011592           INC      DRBUFPH        ; BUMP HI ADDRESS
011593           BCS      ECALC0        ; BRANCH ALWAYS.
011594 *
011595 ECALC2     STA      DRBUFPL        ; SAVE NEWLY CALCULATED LOW ADDRESS
011596           RTS
011597           PAGE
011598 DERROR2    RTS
011599 *
011600 DREVISE     LDA      DATELO        ; IF NO CLOCK,
011601           BEQ      DREVISE1       ; THEN DON'T TOUCH MOD T/D
011602           LDX      #3              ; MOVE LAST MODIFICATION DATE/TIME TO ENTRY BEING UPDATED.
011603 MODTIME   LDA      DATELO,X
011604           STA      DFIL+D.MODDT,X
011605           DEX
011606           BPL      MODTIME
011607 *
011608 DREVISE1    LDA      DFIL+D.ATTR    ; MARK ENTRY AS BACKUPABLE
011609           ORA      BKBITFLG       ; BIT 5 = BACKUP NEEDED BIT
011610           STA      DFIL+D.ATTR
011611           LDA      D.DEV          ; GET DEVICE NUMBER OF DIRECTORY
011612           STA      DEVNUM        ; TO BE REVISED.
011613           LDA      D.ENTBLK      ; AND ADDRESS OF DIRECTORY BLOCK
011614           STA      BLOKNML       ; THAT CONTAINS THE ENTRY.
011615           LDA      D.ENTBLK+1
011616           STA      BLOKNMH
011617           JSR      RDGBUF        ; READ BLOCK INTO GENERAL PURPOSE BUFFER.
011618           BCS      ERRGBUF
011619           JSR      ENTALCALC     ; FIX UP POINTER TO ENTRY LOCATION WITHIN GBUF.
011620           LDY      H.ENTLN        ; NOW MOVE 'D.' STUFF TO DIRECTORY.
011621           DEY
011622 MVDENT     LDA      DFIL+D.STOR,Y
011623           STA      (DRBUFPL),Y
011624           DEY
011625           BPL      MVDENT
011626           LDA      D.HEAD         ; IS THE ENTRY BLOCK THE SAME AS THE
011627           CMP      BLOKNML        ; ENTRY'S HEADER BLOCK?
011628           BNE      SVENTDIR       ; NO, SAVE ENTRY BLOCK
011629           LDA      D.HEAD+1       ; MAYBE, TEST HIGH ADDRESSES
011630           CMP      BLOKNMH
011631           BEQ      UPHEAD         ; BRANCH IF THEY ARE THE SAME BLOCK.

```

```

011632 SVENTDIR      JSR      WRGGBUF      ; WRITE UPDATED DIRECTORY BLOCK
011633              BCS      DERROR2      ; RETURN ANY ERROR.
011634              LDA      D.HEAD      ; GET ADDRESS OF HEADER BLOCK
011635              STA      BLOKNML
011636              LDA      D.HEAD+1
011637              STA      BLOKNMH
011638              JSR      RDGBUF      ; READ IN HEADER BLOCK FOR MODIFICATION
011639              BCS      DERROR2
011640 UPHEAD        LDY      #1          ; UPDATE CURRENT NUMBER OF FILES IN THIS DIRECTORY
011641 UPHED1         LDA      H.FCNT,Y
011642              STA      GBUF+HCENT+4,Y ; (CURRENT ENTRY COUNT)
011643              DEY
011644              BPL      UPHED1
011645              LDA      H.ATTR      ; ALSO UPDATE HEADER'S ATTRIBUTES.
011646              STA      GBUF+HATTR+4
011647              JSR      WRGGBUF
011648 DERROR1       RTS              ; IMPLICITLY RETURN ANY ERRORS
011649 *
011650              PAGE
011651 *
011652 NOTDIR         LDA      #TYPEERR    ; NOT TREE OR DIRECTORY- NOT A RECOGNIZED TYPE!
011653 TSTERR        SEC
011654              RTS              ; DO NOTHING.
011655 *
011656 *
011657 TSTSOS        LDA      GBUF        ; TEST SOS STAMP
011658              CMP      SOSTMPL
011659              BNE      TSTERR
011660              LDA      GBUF+1
011661              CMP      SOSTMPH
011662              BNE      TSTERR
011663              LDA      GBUF+4      ; TEST FOR HEADER
011664              AND      #$E0
011665              CMP      #HEDTYP*16
011666              BNE      TSTERR      ; BRANCH IF NOT SOS HEADER (NO ERROR NUMBER)
011667              CLC              ; INDICATE NO ERROR
011668              RTS
011669 *
011670              CHN      FNDFIL,4,1
011671 NE           TSTERR
011672              LDA      GBUF+4      ; TEST FOR HEADER
011673              AND      #$E0
011674              CMP      #HEDTYP*16
011675              BNE      TSTERR      ; BRANCH IF NOT SOS HEADER (NO ERROR NUMBER)
011676              CLC              ; INDICATE NO ERROR
011677              RTS
011678 *
011679              CHN      FNDFIL,4,1
011680 O            ERROR
011681              RTS

```

```

011682 *
011683         CHN         FNDFIL,4,1
011684         ENTRY     TOO.
011685         LDY         #D.MODDT+3
011686 RIPTIME   LDA         DATELO,X
011687         STA         (DRBUFPL),Y
011688         DEY
011689         DEX
011690         BPL         RIPTIME           ;MOVE ALL FOR BYTES...
011691 RUPDATE   JSR         WRGBUF           ;WRITE UPDATED ENTRY BACK TO DISK. (ASSUMES BLOKNM UNDISTURBEDD)
011692         BCS         DERROR1           ;GIVE UP ON ANY ERROR.
011693         LDY         #D.DHDR           ;NOW COMPARE CURRENT BLOCK NUMBER TO THIS
011694         LDA         (DRBUFPL),Y       ; ENTRY'S HEADER BLOCK
011695         INY
011696         CMP         BLOKNML           ;ARE LOW ADDRESSES THE SAME?
011697         STA         BLOKNML           ;(SAVE IT IN CASE IT'S NOT)
011698         BNE         RIPPLE2           ;BRANCH IF ENTRY DOES NOT RESIDE IN SAME BLOCK AS HEADER.
011699         LDA         (DRBUFPL),Y       ;CHECK HIGH ADDRESS JUST TO BE SURE.
011700         CMP         BLOKNMH
011701         BEQ         RIPPLE           ;THEY ARE THE SAME, CONTINUE RIPPLE TO ROOT DIRECTORY.
011702 RIPPLE2   LDA         (DRBUFPL),Y       ;THEY AREN'T THE SAME, READ IN THIS DIRECTORY'S HEADER.
011703         STA         BLOKNMH
011704         JSR         RDGBUF
011705         BCC         RIPPLE           ;CONTINUE IF READ WAS GOOD.
011706 DERROR1   EQU         *
011707         RTS
011708         PAGE
011709 *
011710 NOTDIR    LDA         #TYPEERR           ;NOT TREE OR DIRECTORY- NOT A RECOGNIZED TYPE!
011711 TSTERR    SEC
011712         RTS           ;DO NOTHING.
011713 *
011714 *
011715 TSTSOS    LDA         GBUF           ;TEST SOS STAMP
011716         CMP         SOSTMPL
011717         BNE         TSTERR
011718         LDA         GBUF+1
011719         CMP         SOSTMPH
011720         BNE         TSTERR
011721         LDA         GBUF+4           ;TEST FOR HEADER
011722         AND         #$E0
011723         CMP         #HEDTYP*16
011724         BNE         TSTERR           ;BRANCH IF NOT SOS HEADER (NO ERROR NUMBER)
011725 DRVISDNE  CLC           ;INDICATE NO ERROR./
011726         RTS
011727 *
011728         CHN         FNDFIL,4,1
011729
011730
011731 *****

```

011732 \* END OF APPLE /// SOS 1.3 SOURCE CODE FILE: CREATE  
011733 \*\*\*\*\*  
011734

```

011735 =====
011736 DOCUMENT :SOS1.3.3of5.THREE:SOS.EQUATES.TXT
011737 =====
011738
011739 *****
011740 * APPLE /// SOS 1.3 SOURCE CODE FILE: EQUATES
011741 *****
011742 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
011743
011744 *
011745             ENTRY      BFMGR
011746 *
011747 * BFM INITIALIZATION ENTRIES
011748 * (INIT CODE FOUND IN INIT.SRC)
011749 *
011750             ENTRY      BFMFCB1             ; FCB PAGE 1 ADDR
011751             ENTRY      BFMFCB2             ; AND PAGE 2
011752             ENTRY      FCBZPP
011753             ENTRY      SISTER
011754             ENTRY      PATHBUF
011755             ENTRY      VCB
011756             ENTRY      WORKSPC
011757             ENTRY      PFXPTR
011758             ENTRY      BMAPAGE
011759             ENTRY      BMBPAGE
011760             ENTRY      FCBADDRH
011761             ENTRY      BMAMADR
011762             ENTRY      BMBMADR
011763 *
011764 *
011765             EXTRN      LEVEL                 ; FILE LEVEL (LOW BYTE)
011766             EXTRN      OPMSGRPLY           ; OPERATOR MESSAGE
011767             EXTRN      DATETIME            ; THANKS TOM...
011768             EXTRN      DMGR               ; THANKS BOB...
011769             EXTRN      REQBUF              ; "
011770             EXTRN      REQFXBUF           ; "
011771             EXTRN      GETBUFADR           ; "
011772             EXTRN      RELBUF              ; "
011773             EXTRN      BLKDLST            ; "
011774             EXTRN      SERR
011775             EXTRN      BACKMASK
011776 *
011777 * ERRORS
011778 *
011779             EXTRN      SYSERR
011780 *
011781             EXTRN      BADPATH              ; INVALID PATHNAME SYNTAX
011782             EXTRN      FCBFULL             ; FILE CONTROL BLOCK FULL
011783             EXTRN      BADREFNUM           ; INVALID REFNUM

```



```

011784      EXTRN      PATHNOTFND      ; PATHNAME NOT FOUND
011785      EXTRN      VNFERR          ; VOLUME NOT FOUND
011786      EXTRN      FNFERR          ; FILE NOT FOUND
011787      EXTRN      DUPVOL          ; DUPLICATE FILE NAME ERROR
011788      EXTRN      DUPVOL          ; DUPLICATE VOLUME CAN'T BE LOGGED IN.
011789      EXTRN      OVRERR          ; NOT ENOUGH DISK SPACE FOR PREALLOCATION
011790      EXTRN      DIRFULL        ; DIRECTORY FULL ERROR
011791      EXTRN      CPTERR          ; FILE INCOMPATIBLE SOS VERSION
011792      EXTRN      TYPERR          ; NOT CURRENTLY SUPPORTED FILE TYPE
011793      EXTRN      EOFERR          ; POSITION ATTEMPTED BEYOND END OF FILE
011794      EXTRN      POSNERR        ; ILLEGAL POSITION (L.T. 0 OR G.T. $FFFFFF)
011795      EXTRN      ACCSERR        ; FILE ACCESS R/W REQUEST CONFLICTS WITH ATTRIBUTES.
011796      EXTRN      BTSERR          ; USER SUPPLIED BUFFER TOO SMALL
011797      EXTRN      FILBUSY        ; EITHER WRITE WAS REQUESTED OR WRITE ACCESS ALREADY ALLOCATED.
011798      EXTRN      NOTSOS          ; NOT A SOS DISKETTE
011799      EXTRN      BADLSTCNT       ; INVALID VALUE IN LIST PARAMETER
011800      EXTRN      XDISKSW        ; DISK SWITCHED
011801      EXTRN      NOTBLKDEV       ; NOT A BLOCK DEVICE
011802      EXTRN      XNOWRITE        ; DISK/MEDIA IS HARDWARE WRITE PROTECTED
011803      EXTRN      XIOERROR        ; INFORMATION ON BLOCK DEVICE NOT ACCESSABLE
011804      EXTRN      DIRERR          ; DIRECTORY ENTRY COUNT INCONSISTENT WITH ACTUAL ENTRIES
011805      EXTRN      BITMAPADR       ; BIT MAP DISK ADDRESS IMPOSSIBLE
011806      *
011807      * FATAL ERRORS
011808      *
011809      EXTRN      SYSDEATH
011810      *
011811      EXTRN      VCBERR          ; VOLUME CONTROL BLOCK NOT USABLE
011812      EXTRN      ALCERR          ; ALLOCATION BLOCKS INVALID
011813      EXTRN      TOOLONG        ; PATHNAME BUFFER OVERFLOW
011814      PAGE
011815      *
011816      * CONSTANTS
011817      *
011818      DLIMIT      EQU            $2F      ; DELIMITER IS CURRENTLY AN ASCII '/'
011819      SEEDTYP     EQU            1
011820      SAPTYP      EQU            2
011821      TRETYP      EQU            3
011822      DIRTYP      EQU            $D
011823      HEDTYP      EQU            $E
011824      RDCMD       EQU            $0
011825      WRTCMD      EQU            $1
011826      RPTCMD      EQU            $9
011827      STATCMD     EQU            $02      ; REQUEST STATUS OF BLOCK DEVICE. (BIT 0 = WRITE PROTECTED)
011828      STATSUB      EQU            $0
011829      PRETIME      EQU            $20      ; COMMAND NEEDS CURRENT DATE/TIME STAMP
011830      PREREF       EQU            $40      ; COMMAND REQUIRES FCB ADDRESS AND VERIFICATION
011831      PREPATH      EQU            $80      ; COMMAND HAS PATHNAME TO PREPROCESS
011832      SISTER       EQU            $1400
011833      *

```

```

011834 * VOLUME STATUS CONSTANTS (BITS)
011835 *
011836 DSWITCH      EQU      $40      ; FOR DISK SWITCHED ERROR RECOVERY.
011837 *
011838 * FILE STATUS CONSTANTS
011839 *
011840 DATALC        EQU      $1      ; DATA BLOCK NOT ALLOCATED.
011841 IDXALC        EQU      $2      ; INDEX NOT ALLOCATED
011842 TOPALC        EQU      $4      ; TOP INDEX NOT ALLOCATED
011843 STPMOD        EQU      $8      ; STORAGE TYPE MODIFIED
011844 USEMOD        EQU      $10     ; FILE USAGE MODIFIED
011845 EOFMOD        EQU      $20     ; END OF FILE MODIFIED
011846 DATMOD        EQU      $40     ; DATA BLOCK MODIFIED
011847 IDXMOD        EQU      $80     ; INDEX BLOCK MODIFIED
011848 FCBMOD        EQU      $80     ; HAS FCB/DIRECTORY BEEN MODIFIED? (FLUSH)
011849 *
011850 * FILE ATTRIBUTES CONSTANTS
011851 *
011852 READEN        EQU      $1      ; READ ENABLED
011853 WRITEN        EQU      $2      ; WRITE ENABLED
011854 NLINE        EQU      $10     ; NEW LINE ENABLED
011855 BKBITVAL      EQU      $20     ; FILE NEEDS BACKUP IF SET (BKBITFLG)
011856 RENAMEN        EQU      $40     ; RENAME OK WHEN ON.
011857 DSTROYEN      EQU      $80     ; DESTROY OK WHEN ON.
011858 PAGE
011859 * HEADER INDEX CONSTANTS
011860 *
011861 HNLEN         EQU      $0      ; HEADER NAME LENGTH (OFFSET INTO HEADER)
011862 *HNAME EQU $1 ; HEADER NAME
011863 HPENAB        EQU      $10     ; PASSWORD ENABLE BYTE
011864 HPASS        EQU      $11     ; ENCODED PASSWORD
011865 HCRDT        EQU      $18     ; HEADER CREATION DATE
011866 * HCRTM EQU $1A ; HEADER CREATION TIME
011867 HVER         EQU      $1C     ; SOS VERSION THAT CREATED DIRECTORY
011868 HCMP         EQU      $1D     ; BACKWARD COMPATIBLE WITH SOS VERSION
011869 HATTR        EQU      $1E     ; HEADER ATTRIBUTES- PROTECT ETC.
011870 * HENTLN EQU $1F ; LENGTH OF EACH ENTRY
011871 * HMENT EQU $20 ; MAXIMUM NUMBER OF ENTRIES/BLOCK
011872 HCENT        EQU      $21     ; CURRENT NUMBER OF FILES IN DIRECTORY
011873 HRBLK        EQU      $23     ; OWNER'S DIRECTORY ADDRESS
011874 HRENT        EQU      $25     ; OWNER'S DIRECTORY ENTRY NUMBER
011875 HRELN        EQU      $26     ; OWNER'S DIRECTORY ENTRY LENGTH
011876 VBMAP        EQU      HRBLK
011877 VTBLK        EQU      HRENT   ; (USED FOR ROOT DIRECTORY ONLY)
011878 *
011879 * VOLUME CONTROL BLOCK INDEX CONSTANTS
011880 *
011881 VCBSIZE        EQU      $20     ; CURRENT VCB IS 32 BYTES PER ENTRY (VER 0)
011882 VCBNML        EQU      0      ; VOLUME NAME LENGTH BYTE
011883 VCBNAM        EQU      1      ; VOLUME NAME

```

```

011884 VCBDEV      EQU      $10      ; VOLUME'S DEVICE
011885 VCBSTAT    EQU      $11      ; VOLUME STATUS. (80=FILES OPEN. 40=DISK SWITCHED.)
011886 VCBTBLK    EQU      $12      ; TOTAL BLOCKS ON THIS VOLUME
011887 VCBTFRE    EQU      $14      ; NUMBER OF UNUSED BLOCKS
011888 VCBROOT    EQU      $16      ; ROOT DIRECTORY (DISK) ADDRESS
011889 *VCBMORG EQU $18 ; MAP ORGANIZATION (NOT SUPPORTED BY V 0)
011890 *VCMBBUF EQU $19 ; BIT MAP BUF NUM
011891 VCBDMAP     EQU      $1A      ; FIRST (DISK) ADDRESS OF BITMAP(S)
011892 VCBCMAP     EQU      $1C      ; RELATIVE ADDRESS OF BIT MAP WITH SPACE (ADD TO VCBDMAP)
011893 *VCBMNUM EQU $1D ; RELATIVE BIT MAP CURRENTLY IN MEMORY
011894 VCBOPNC     EQU      $1E      ; CURRENT NUMBER OF OPEN FILES.
011895 VCBSWAP     EQU      $1F      ; $8X IF VOLUME SWAPPED; $00 IF UNSWAPPED WHERE X=LOW ORDER BYTE OF VCB
ADR/16
011896 *
011897 * FILE CONTROL BLOCK INDEX CONSTANTS
011898 *
011899 FCBREFN      EQU      0        ; FILE REFERENCE NUMBER (POSITION SENSITIVE)
011900 FCBDEVN      EQU      1        ; DEVICE (NUMBER) ON WHICH FILE RESIDES
011901 *FCBHEAD EQU 2 ; BLOCK ADDRESS OF FILE'S DIRECTORY HEADER
011902 *FCBDIRB EQU 4 ; BLOCK ADDRESS OF FILE'S DIRECTORY
011903 FCBENTN      EQU      6        ; ENTRY NUMBER WITHIN DIRECTORY BLOCK
011904 FCBSTYP      EQU      7        ; STORAGE TYPE - SEED, SAPLING, TREE, ETC.
011905 FCBSTAT      EQU      8        ; STATUS - INDEX/DATA/EOF/USAGE/TYP MODIFIED.
011906 FCBATTR      EQU      9        ; ATTRIBUTES - READ/WRITE ENABLE, NEWLINE ENABLE.
011907 FCBNEWL      EQU      $A      ; NEW LINE TERMINATOR (ALL 8 BITS SIGNIFICANT).
011908 FCBBUFN      EQU      $B      ; BUFFER NUMBER
011909 FCBFRST      EQU      $C      ; FIRST BLOCK OF FILE
011910 FCBIDX      EQU      $E      ; BLOCK ADDRESS OF INDEX (0 IF NO INDEX)
011911 FCBDATE      EQU      $10      ; BLOCK ADDRESS OF DATA
011912 FCBMARK      EQU      $12      ; CURRENT FILE MARKER.
011913 FCBEOF      EQU      $15      ; LOGICAL END OF FILE.
011914 FCBUSE      EQU      $18      ; ACTUAL NUMBER OF BLOCKS ALLOCATED TO THIS FILE.
011915 FCBWAP      EQU      $1A      ; $8N = SWAPPED, $00 = UNSWAPPED VOLUME ("N" = VCB ENTRY NUMBER)
011916 FCBLEVL      EQU      $1B      ; LEVEL AT WHICH THIS FILE WAS OPENED
011917 FCBDIRTY    EQU      $1C      ; FCB MARKED AS MODIFIED
011918
011919 *
011920 * ZERO PAGE STUFF
011921 *
011922 PAR          EQU      $A0
011923 COMMAND      EQU      PAR
011924 C.DNAMP        EQU      PAR+1
011925 C.PATH         EQU      PAR+1
011926 C.REFNUM      EQU      PAR+1
011927 C.ISNEWL      EQU      PAR+2
011928 C.OUTEOF      EQU      PAR+2
011929 C.BASE        EQU      PAR+2
011930 C.MRKPTR       EQU      PAR+2
011931 C.OUTBUF      EQU      PAR+2
011932 C.NWPATH      EQU      PAR+3

```

```

011933 C.FILIST      EQU      PAR+3
011934 C.NEWL       EQU      PAR+3
011935 C.OUTVOL     EQU      PAR+3
011936 C.OUTREF     EQU      PAR+3
011937 C.XLIST     EQU      PAR+3
011938 C.MAXPTH     EQU      PAR+3
011939 C.MARK       EQU      PAR+3
011940 C.NEWEOF     EQU      PAR+3
011941 C.BYTES     EQU      PAR+4
011942 C.FILSTLN  EQU      PAR+5
011943 C.OUTBLK    EQU      PAR+5
011944 C.OPLIST    EQU      PAR+5
011945 C.XLEN      EQU      PAR+5
011946 C.FILID    EQU      PAR+6
011947 C.OUTCNT  EQU      PAR+6
011948 C.OPLSTLN  EQU      PAR+7
011949 C.AUXID    EQU      PAR+7
011950 C.STOR     EQU      PAR+9
011951 C.EOFLL    EQU      PAR+$A
011952 C.EOFLH   EQU      PAR+$B
011953 C.EOFHL   EQU      PAR+$C
011954 DEBUPTR  EQU      PAR+$D      ; NOTE SAME AS BELOW
011955 C.EOFHH   EQU      PAR+$D
011956 * C.SPARE EQU PAR+$E
011957 *
011958 DEVICE    EQU      $C0
011959 DHPCMD    EQU      DEVICE
011960 UNITNUM   EQU      DEVICE+1
011961 DSTATREQ  EQU      DEVICE+2
011962 DBUFPL    EQU      DEVICE+2
011963 DBUFPH    EQU      DBUFPL+1
011964 DSTATBFL EQU      DEVICE+3      ; TO PASS BACK BUSY, WRITE PROTECT, READ PROTECT.
011965 DSTATBFH  EQU      DSTATBFL+1
011966 RQCNTL   EQU      DEVICE+4
011967 RQCNTH   EQU      RQCNTL+1
011968 BLOKNML  EQU      DEVICE+6
011969 BLOKNMH  EQU      BLOKNML+1
011970 BRDPTR   EQU      DEVICE+8      ; (AND 9)
011971 *
011972 DVNAMP    EQU      DEVICE+1      ; USED FOR 'VOLUME' TO CALL
011973 DVDNUM   EQU      DEVICE+3      ; 'GET.DNUM' IN DEVICE MANAGER.
011974 *
011975 SISBPH    EQU      SISTER+DBUFPH
011976 SISDSTAT EQU      SISTER+DSTATBFH
011977 SSBDRPH  EQU      SISTER+BRDPTR+1
011978 *
011979 *                PAGE
011980 *
011981 * ZERO PAGE TEMPORARIES
011982 *

```

```

011983 ZTEMPS      EQU      $B0
011984 PATHNML    EQU      ZTEMPS
011985 PATHNMH    EQU      PATHNML+1
011986 USRBUF     EQU      ZTEMPS
011987 TPATH     EQU      ZTEMPS+2
011988 WRKPATH    EQU      ZTEMPS+4
011989 TINDX      EQU      ZTEMPS+2
011990 DRBUFPL    EQU      ZTEMPS+4
011991 DRBUFPH    EQU      DRBUFPL+1
011992 VCBPTR     EQU      ZTEMPS+6
011993 BMADR      EQU      ZTEMPS+8
011994 FCBPTR     EQU      ZTEMPS+$A
011995 DATPTR     EQU      ZTEMPS+$C
011996 POSPTR     EQU      ZTEMPS+$E
011997 *
011998 MAXTEMPS    EQU      $F
011999 SISTEMPS   EQU      SISTER+ZTEMPS
012000 SSTIDXH     EQU      SISTER+TINDX+1
012001 SISPATH    EQU      SISTER+C.PATH+1
012002 SSNWPATH   EQU      SISTER+C.NWPATH+1
012003 SISUSRBF   EQU      SISTER+USRBUF+1
012004 SISOUTBF   EQU      SISTER+C.OUTBUF+1
012005 SISTPATH   EQU      SISTER+TPATH+1
012006 SISBMADR   EQU      SISTER+BMADR+1
012007 SISFCBP    EQU      SISTER+FCBPTR+1
012008 SISDATP    EQU      SISTER+DATPTR+1
012009 SISPOSP    EQU      SISTER+POSPTR+1
012010 *
012011 *
012012 * ADDRESSES
012013 *
012014 PATHBUF     EQU      $1000      ; NOTE: THIS IS $100 BYTES LONG.
012015 VCB         EQU      $1100
012016 GBUF       EQU      $1200      ; THRU $13FF
012017 *
012018 * INITIALIZATION EQUATES
012019 *
012020 BFMFCB1       EQU      $1C        ; FCB PAGE 1 ADDR
012021 BFMFCB2       EQU      $1D        ; FCB PAGE 2 ADDR
012022 BMAPAGE     EQU      <$B800    ; BIT MAP A ADDR
012023 BMBPAGE     EQU      <$BA00    ; BIT MAP B ADDR
012024 FCBZPP      EQU      FCBPTR
012025 *
012026 *
012027 *
012028             PAGE
012029             DSECT
012030             ORG      $0          ; (THE FOLLOWING DO NOT NEED TO BE ON ZERO PAGE. 7/16/80 JRH.)
012031 DATBLKL     DS      1
012032 DATBLKH     DS      1

```

```

012033  IDXADRL      DS      1      ; DISK ADDRESS OF INDEX BLOCK
012034  IDXADRH      DS      1
012035  REQI        DS      1
012036  REQH        DS      1
012037  INDXBLK    DS      1
012038  LEVELS     DS      1
012039  TOTENT     DS      1
012040  ENTCNTL    DS      1
012041  ENTCNTH    DS      1
012042  CNTENT     DS      1
012043  NOFREE     DS      1
012044  BMCNT      DS      1
012045  SAPTR      DS      1
012046  TREPTR     DS      1
012047  TLINK      DS      2
012048  FLINK      DS      2
012049  PATHCNT    DS      1
012050  PFXPTR     DS      2
012051  BMPTR      DS      1
012052  BASVAL     DS      1
012053  HALF       DS      1
012054  *
012055  *
012056  PAGE
012057  *
012058  * BIT MAP INFO TABLES (A & B)
012059  *
012060  BMTABSZ     EQU      $6
012061  BMTAB       DS      1
012062  BMBUFBNK    DS      1
012063  BMASTAT     DS      1
012064  BMADEV      DS      1
012065  BMAMADR      DS      1
012066  BMADADR      DS      2
012067  BMACMAP     DS      1      ; SIMILAR TO VCBCMAP
012068  BMBSTAT     DS      1
012069  BMBDEV      DS      1
012070  BMBMADR     DS      1
012071  DS          2      ; BMBDADR
012072  DS          1      ; BMBCMAP
012073  *
012074  FCBADDRH    DS      1      ; FILE CONTROL BLOCK'S BUFFER ADDRESS.
012075  FCBANKNM     DS      1      ; AND BANK (SISTER PAGE) BYTE.
012076  TPOSLL      DS      1
012077  TPOS LH      DS      1
012078  TPOSHI      DS      1
012079  RWREQL      DS      1
012080  RWREQH      DS      1
012081  BULKCNT     DS      1
012082  NLCHAR      DS      1

```

```

012083 NPATHDEV      DS          3          ; FOR NEW PATHNAME DEVICE AND DIRECTORY HEADER ADDRESS
012084 IOACCESS      DS          1          ; USED TO DETERMINE IF A CALL HAS BEEN MADE TO THE DISK DEVICE HANDLER
012085 DEVNUM         DS          1          ; CURRENT DEVICE TO BE ACCESSED.
012086 TOTDEVS        DS          1          ; USED FOR ACCESSING DRIVES IN NUMERIC ORDER
012087 CMDTEMP        DS          1          ; USED FOR TESTING REFNUM, TIME, AND DSKSWTCH (PRE)PROCESSING.
012088 DATELO          DS          1          ; DATE AND TIME MUST RESIDE ON ZERO PAGE.
012089 DATEHI          DS          1
012090 TIMELO          DS          1
012091 TIMEHI          DS          1
012092 *
012093 DUPLFLAG        DS          1          ; USED FOR DIFFERENCE BETWEEN VNFERR AND DUPVOL BY SYNPATH
012094 ZPGTEMP         DS          1          ; A ONE-BYTE UNSTABLE TEMPORARY
012095 VCBENTRY        DS          1          ; POINTER TO CURRENT VCB ENTRY
012096 *
012097                DEND
012098 *
012099                CHN          PATH,4,1
012100
012101 *****
012102 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: EQUATES
012103 *****
012104
012105

```

```

012106 =====
012107 DOCUMENT :SOS1.3.3of5.THREE:SOS.FNDFIL.TEXT
012108 =====
012109
012110 *****
012111 * APPLE /// SOS 1.3 SOURCE CODE FILE: FNDFIL
012112 *****
012113 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
012114
012115 PAGE
012116 *
012117 *
012118 FINDFILE JSR LOOKFILE ; SEE IF FILE EXISTS
012119 BCS NOFIND ; BRANCH IF AN ERROR WAS ENCOUNTERED
012120 MOVENTRY LDY H.ENTLN ; MOVE ENTIRE ENTRY INFO TO A SAFE AREA
012121 MOVENT1 LDA (DRBUFPL),Y
012122 STA DFIL+D.STOR,Y
012123 DEY
012124 BPL MOVENT1
012125 LDA #0 ; TO INDICATE ALL IS WELL
012126 NOFIND RTS ; RETURN CONDITION CODES.
012127 PAGE
012128 *
012129 *
012130 LOOKFILE JSR PREPROOT ; FIND VOLUME AND SET UP OTHER BORING STUFF
012131 BCS FNDERR ; PASS BACK ANY ERROR ENCOUNTERED
012132 LDY #0 ; TEST TO SEE IF ONLY ROOT WAS SPECIFIED.
012133 LDA (PATHNML),Y
012134 BNE LOOKFIL0 ; BRANCH IF MORE THAN ROOT.
012135 LDA #GBUF/256 ; OTHERWISE, REPORT A BADPATH ERROR
012136 STA DRBUFPH ; (BUT FIRST CREATE A PHANTOM ENTRY FOR OPEN)
012137 LDA #4
012138 STA DRBUFPL
012139 LDY #D.AUXID ; FIRST MOVE IN ID, AND DATE STUFF.
012140 PHANTM1 LDA (DRBUFPL),Y
012141 STA DFIL,Y
012142 DEY
012143 CPY #D.CREDT-1
012144 BNE PHANTM1
012145 PHANTM2 LDA ROOTSTUF-D.FILID,Y
012146 STA DFIL,Y
012147 DEY
012148 CPY #D.FILID-1
012149 BNE PHANTM2
012150 LDA #DIRTYP*$10 ; FAKE DIRECTORY FILE
012151 STA DFIL+D.STOR
012152 LDA #BADPATH ; (CARRY IS SET)
012153 RTS
012154 *

```



```

012155 ROOTSTUF      DFB      0,2,0,4
012156              DFB      0,0,8,0
012157 *
012158 LOOKFIL0      LDA      #0              ; RESET FREE ENTRY INDICATOR
012159              STA      NOFREE
012160              SEC              ; INDICATE THAT THE DIRECTORY TO BE SEARCHED HAS HEADER IN THIS BLOCK
012161 LOOKFIL1      LDA      #0              ; RESET ENTRY COUNTER
012162              STA      TOTENT
012163              JSR      LOOKNAM          ; LOOK FOR NAME POINTED TO BY 'PATHNML'
012164              BCC      NAMFOJMP        ; BRANCH IF NAME WAS FOUND.
012165              LDA      ENTCNTL         ; HAVE WE LOOKED AT ALL OF THE
012166              SBC      TOTENT         ; ENTRIES IN THIS DIRECTORY?
012167              BCC      DCRENTH        ; MAYBE, CHECK HI COUNT.
012168              BNE      LOOKFIL2       ; NO, READ NEXT DIRECTORY BLOCK
012169              CMP      ENTCNTH        ; HAS THE LAST ENTRY BEEN LOOKED AT (ACC=0)
012170              BEQ      ERRFNF        ; YES, GIVE 'FILE NOT FOUND' ERROR.
012171              BNE      LOOKFIL2       ; BRANCH ALWAYS.
012172 DCRENTH        DEC      ENTCNTH        ; SHOULD BE AT LEAST 1
012173              BPL      LOOKFIL2       ; (THIS SHOULD BE BRANCH ALWAYS...)
012174 ERRDIR         LDA      #DIRERR        ; REPORT DIRECTORY MESSED UP.
012175 FNDERR        SEC
012176              RTS
012177 NAMFOJMP      JMP      NAMFOUND       ; AVOID BRANCH OUT OF RANGE
012178 *
012179              PAGE
012180 LOOKFIL2      STA      ENTCNTL          ; KEEP RUNNING COUNT
012181              LDA      #GBUF/256        ; RESET INDIRECT POINTER
012182              STA      DRBUFPH
012183              LDA      GBUF+2          ; GET LINK TO NEXT DIRECTORY BLOCK
012184              BNE      NXTDIR0         ; (IF THERE IS ONE)
012185              CMP      GBUF+3          ; ARE BOTH ZERO, I.E. NO LINK?
012186              BEQ      ERRDIR         ; IF SO, THEN NOT ALL ENTRIES WERE ACCOUNTED FOR.
012187 NXTDIR0      STA      BLOKNML
012188              LDA      GBUF+3
012189              STA      BLOKNMH
012190              JSR      RDGBUF          ; GO READ THE NEXT LINKED DIRECTORY IN.
012191              BCC      LOOKFIL1        ; BRANCH IF NO ERROR.
012192              RTS                      ; RETURN ERROR (IN ACCUMULATOR).
012193 TELFREEX      JMP      TELFREE
012194 *
012195 FNFOX        JMP      FNF0              ; AVOID BRANCH OUT OF RANGE
012196 *
012197 CFLAG        DS      1              ; AM I CREATING?
012198 TTSAVE       DS      2              ; CURRENT BLOCK ADDR
012199 BLOKSAVE     DS      2              ; PARENT DIR ADDR
012200 *
012201 ERRFNF      LDA      NOFREE          ; WAS ANY FREE ENTRY FOUND?
012202              BNE      FNFOX
012203              LDA      GBUF+2          ; TEST LINK
012204              BNE      TELFREEX

```

```

012205          CMP      GBUF+3          ; IF BOTH ARE ZERO, THEN GIVE UP
012206          BNE      TELFREEEX      ; BRANCH IF NOT LAST DIR BLOCK
012207          LDA      CFLAG          ; DOING A CREATE?
012208          BEQ      FNFOX          ; NO, SIMPLY REPORT NOT FOUND
012209 *
012210 * EXTEND THE DIRECTORY BY A BLOCK
012211 *
012212          LDA      BLOKSAVE        ; BUT NOT
012213          ORA      BLOKSAVE+1      ; IF A ROOT DIRECTORY!
012214          BEQ      FNFOX          ; FORU BLOCKS HARD CODED
012215          LDA      TTLINK          ; FETCH CURRENT DIRECTORY
012216          STA      TLINK          ; ADDR (GBUF)
012217          LDA      TTLINK+1        ; AND ALLLOCATE A NEW
012218          STA      TLINK+1        ; BY LINKING TO CURRENT
012219          JSR      DIRWRT
012220          BCS      FNFO          ; RATS! NO SPACE SAY "DIRFULL"
012221 *
012222 * SAVE CURRENT BLOCK ADDR
012223 *
012224          LDA      TTLINK
012225          STA      TTSAVE
012226          LDA      TTLINK+1
012227          STA      TTSAVE+1
012228 *
012229 * FETCH DESCENDENT
012230 *
012231          LDA      GBUF+2
012232          STA      BLOKNML
012233          LDA      GBUF+3
012234          STA      BLOKNMH
012235          JSR      ZERGBUF          ; INIT THE NEW DIR BLOCK
012236 *
012237 * AND INSERT BACK POINTER
012238 * TO "CURRENT BLOCK"
012239 *
012240          LDA      TTSAVE
012241          STA      GBUF
012242          LDA      TTSAVE+1
012243          STA      GBUF+1
012244          JSR      WRTGBUF
012245          BCS      ERTS
012246 *
012247 * UPDATE DIR'S HEADER IN PARENT
012248 *
012249          LDA      BLOKSAVE
012250          STA      BLOKNML          ; PREPARE TO READ PARENT
012251          LDX      BLOKSAVE+1
012252          STX      BLOKNMH
012253          JSR      RDGBUF          ; FETCH PARENT
012254          LDY      #D.USAGE        ; BUMP BLOCKS USED BY HEADER

```

```

012255      LDA      (DEBUPTR),Y
012256      SEC
012257      ADC      #0          ; BY JUST ONE BLOCK
012258      STA      (DEBUPTR),Y
012259      INY
012260      LDA      (DEBUPTR),Y          ; TWO BYTE BLOCKS USED
012261      ADC      #0
012262      STA      (DEBUPTR),Y
012263      LDY      #D.EOF+1          ; INCREASE EOF BY $200
012264      LDA      (DEBUPTR),Y
012265      CLC
012266      ADC      #2
012267      STA      (DEBUPTR),Y
012268      INY
012269      LDA      (DEBUPTR),Y
012270      ADC      #0
012271      STA      (DEBUPTR),Y
012272      JSR      WRIGBUF          ; REWRITE PARENT DIR BLOCK
012273      LDA      TTSAVE+1          ; REFETCH CURRENT DIR BLOCK
012274      STA      BLOKNMH
012275      LDA      TTSAVE
012276      STA      BLOKNML
012277      JSR      RDGBUF          ; BACK FROM THE SHADOWS AGAIN
012278      JMP      ERRFNF          ; VOILA! WE HAVE EXTENDED THE DIRECTORY!
012279      *
012280  TELFREE      STA      D.ENTBLK
012281      LDA      GBUF+3
012282      STA      D.ENTBLK+1          ; ASSUME FIRST ENTRY OF NEXT BLOCK
012283      LDA      #1          ; IS FREE FOR USE.
012284      STA      D.ENTNUM
012285      STA      NOFREE          ; MARK D.ENTNUM AS VALID (FOR CREATE)
012286  FNFO        LDY      #0          ; TEST FOR 'FILE NOT FOUND' VERSUS 'PATH NOT FOUND'
012287      LDA      (PATHNML),Y
012288      TAY
012289      INY
012290      LDA      (PATHNML),Y          ; IF NON-ZERO THEN 'PATH NOT FOUND'
012291  ERRPATH1    SEC          ; IN EITHER CASE, INDICATE ERROR.
012292      BEQ      FNF1
012293      LDA      #PATHNOTFND          ; REPORT NO SUCH PATH.
012294  ERTS        RTS
012295  FNF1        LDA      #FNFERR          ; REPORT FILE NOT FOUND.
012296      RTS
012297      PAGE
012298      *
012299  NAMFOUND    LDA      (PATHNML),Y          ; (Y=0)
012300      SEC
012301      ADC      PATHNML          ; TEST FOR LAST NAME IN PATH
012302      TAY          ; IF ZERO, THEN THAT WAS LAST NAME
012303      CLC          ; TO INDICATE SUCCESS
012304      LDA      PATHBUF,Y

```

```

012305          BEQ          FILFOUND
012306 *NOW CHANGE THE PATHNAME POINTER TO POINT AT THE NEXT NAME IN THE PATH
012307          STY          PATHNML
012308          LDA          DRBUFPL          ; SAVE PARENTS
012309          STA          DEBUPTR          ; ENTRY POINTER
012310          LDA          DRBUFPH
012311          STA          DEBUPTR+1        ; IN CASE ENTRY ON PAGE 2
012312          LDA          BLOKNML          ; ADDRESS (DIR EXTEND)
012313          STA          BLOKSAVE
012314          LDA          BLOKNMH
012315          STA          BLOKSAVE+1
012316          LDY          #D.STOR          ; BE SURE THIS IS A DIRECTORY ENTRY
012317          LDA          (DRBUFPL),Y      ; HIGH NIBBLE WILL TELL
012318          AND          #$F0
012319          CMP          #DIRTYP*16        ; IS IT A SUB-DIRECTORY?
012320          BNE          ERRPATH1          ; REPORT THE USER'S MISTAKE
012321          LDY          #D.FRST           ; GET ADDRESS OF FIRST SUB-DIRECTORY BLOCK
012322          LDA          (DRBUFPL),Y
012323          STA          BLOKNML          ; (NO CHECKING IS DONE HERE FOR A VALID
012324          INY                               ; BLOCK NUMBER... )
012325          STA          D.HEAD           ; SAVE AS FILE'S HEADER BLOCK TOO.
012326          LDA          (DRBUFPL),Y
012327          STA          BLOKNMH
012328          STA          D.HEAD+1
012329          JSR          RDGBUF           ; READ SUB-DIRECTORY INTO GBUF
012330          BCS          FNDERR1          ; RETURN IMMEDIATELY ANY ERROR ENCOUNTERED.
012331          LDA          GBUF+HCENT+4      ; GET THE NUMBER OF FILES
012332          STA          ENTCNTL          ; CONTAINED IN THIS DIRECTORY
012333          LDA          GBUF+HCENT+5
012334          STA          ENTCNTH
012335          LDA          GBUF+HCMP+4        ; TEST BACKWARD COMPATIBILITY
012336          BEQ          MOVHEAD
012337 ERRCOMP   LDA          #CPTERR          ; TELL THEM THIS DIRECTORY IS NOT COMPATABLE
012338 NONAME     EQU          *
012339 FNDERR1    SEC
012340          RTS
012341 MOVHEAD   JSR          MOVHED0          ; MOVE INFO ABOUT THIS DIRECTORY
012342          JMP          LOOKFIL0          ; DO NEXT LOCAL PATHNAME
012343 *
012344 MOVHED0    LDX          #$A            ; MOVE INFO ABOUT THIS DIRECTORY
012345 MOVHED1    LDA          GBUF+HCRDT+4,X
012346          STA          H.CREDT,X
012347          DEX
012348          BPL          MOVHED1
012349          RTS
012350 *
012351          PAGE
012352 *
012353 *
012354 FILFOUND   EQU          *

```

```

012355 ENTADR      LDA      H.MAXENT      ; FIGURE OUT WHICH IS ENTRY NUMBER THIS IS.
012356           SEC
012357           SBC      CNTENT        ; MAX ENTRIES - COUNT ENTRIES + 1 = ENTRY NUMBER
012358           ADC      #0            ; (CARRY IS/WAS SET)
012359           STA      D.ENTNUM
012360           LDA      BLOKNML
012361           STA      D.ENTBLK
012362           LDA      BLOKNMH        ; AND INDICATE BLOCK NUMBER OF THIS DIRECTORY.
012363           STA      D.ENTBLK+1
012364           CLC
012365           RTS
012366 *
012367 LOOKNAM     LDA      H.MAXENT      ; RESET COUNT OF FILES PER BLOCK
012368           STA      CNTENT
012369           LDA      #GBUF/256
012370           STA      DRBUFPH
012371           LDA      #4
012372 LOKNAM1     STA      DRBUFPL        ; RESET INDIRECT POINTER TO GBUF
012373           BCS      LOKNAM2        ; BRANCH IF THIS BLOCK CONTAINS A HEADER
012374           LDY      #D.STOR
012375           LDA      (DRBUFPL),Y    ; GET LENGTH OF NAME IN DIRECTORY
012376           BNE      ISNAME        ; BRANCH IF THERE IS A NAME.
012377           LDA      NOFREE        ; TEST TO SEE IF A FREE ENTRY HAS BEEN DECLARED.
012378           BNE      LOKNAM2        ; YES BUMP TO NEXT ENTRY
012379           JSR      ENTADR        ; SET ADDRESS FOR CURRENT ENTRY
012380           INC      NOFREE        ; INDICATE A FREE SPOT HAS BEEN FOUND
012381           BNE      LOKNAM2        ; BRANCH ALWAYS.
012382 *
012383 ISNAME       AND      #$F         ; STRIP TYPE (THIS IS CHECKED BY 'FILFOUND')
012384           INC      TOTENT        ; (BUMP COUNT OF VALID FILES FOUND)
012385           CMP      (PATHNML),Y    ; ARE BOTH NAMES OF THE SAME LENGTH?
012386           BNE      LOKNAM2        ; NO, BUMP TO NEXT ENTRY
012387           TAY
012388 CMPNAME      LDA      (DRBUFPL),Y    ; COMPARE NAMES LETTER BY LETTER
012389           CMP      (PATHNML),Y
012390           BNE      LOKNAM2
012391           DEY                    ; HAVE ALL LETTERS BEEN COMPARED?
012392           BNE      CMPNAME        ; NO, CONTINUE..
012393           CLC
012394           RTS                    ; BY GOLLY, WE GOT US A MATCH!
012395 *
012396 LOKNAM2      DEC      CNTENT        ; HAVE WE CHECKED ALL POSSIBLE ENTRIES IN THIS BLOCK?
012397           BEQ      NONAME        ; YES, GIVE UP.
012398           LDA      H.ENTLN        ; ADD ENTRY LENGTH TO CURRENT POINTER
012399           CLC
012400           ADC      DRBUFPL
012401           BCC      LOKNAM1        ; BRANCH IF WE'RE STILL IN THE FIRST PAGE.
012402           INC      DRBUFPH        ; LOOK ON SECOND PAGE
012403           CLC
012404           BCC      LOKNAM1        ; CARRY SHOULD ALWAYS BE CLEAR BEFORE LOOKING AT NEXT.
                                           ; BRANCH ALWAYS...

```

```

012405          PAGE
012406 *
012407 *
012408 PREPROOT   JSR      FINDVOL          ; FIND CORRECT VOLUME AND DEVICE NUMBER
012409          BCC      ROOT1              ; BRANCH IF IT WAS FOUND.
012410 ROOT0      JSR      LOOKVOL          ; OTHERWISE LOOK ON ALL DEVICES.
012411          BCS      SRITZ              ; CAN'T FIND IT.
012412 ROOT1      LDA      #0                ; ZERO OUT DIRECTORY TEMPS
012413          LDY      #42                ; (DECIMAL)
012414 CLRDSP     STA      D.DEV,Y
012415          DEY
012416          BPL      CLRDSP
012417          LDY      #VCBDEV            ; SET UP DEVICE NUMBER
012418          LDA      (VCBPTR),Y
012419          STA      DEVNUM
012420          STA      D.DEV              ; FOR FUTURE REFERENCE
012421          INY
012422          LDA      (VCBPTR),Y        ; GET CURRENT STATUS OF THIS VOLUME
012423          STA      V.STATUS
012424          LDY      #VCBROOT          ; GET BLOCK ADDRESS OF ROOT DIRECTORY TOO.
012425          LDA      (VCBPTR),Y
012426          STA      BLOKNML
012427          STA      D.HEAD            ; PRESERVE AS HEADER
012428          INY
012429          LDA      (VCBPTR),Y
012430          STA      BLOKNMH
012431          STA      D.HEAD+1
012432          JSR      RDGBUF            ; GO READ IN ROOT
012433          BCC      ROOT2            ; BRANCH IF NO ERROR
012434          PHA
012435          LDY      #VCBSTAT          ; CHECK THIS BUGGER FOR AN OPEN FILE.
012436          LDA      (VCBPTR),Y
012437          ASL      A                  ; (SHIFT OPEN STATUS INTO CARRY)
012438          PLA
012439          BCS      ROOTERR           ; BRANCH IF ERROR NEEDS TO BE REPORTED
012440          BNE      ROOT0            ; OTHERWISE, LOOK ELSEWHERE (BRANCH ALWAYS).
012441 *
012442 ROOT2      JSR      CHKROOT          ; VERIFY ROOT NAME
012443          BEQ      ROOT3            ; BRANCH IF MATCHED.
012444          LDY      #VCBSTAT          ; TEST FOR OPEN FILES ON THIS VOLUME BEFORE
012445          LDA      (VCBPTR),Y        ; LOOKING FOR IT ELSEWHERE.
012446          BPL      ROOT0
012447          JSR      USRREQ            ; REQUEST USER MOUNT VOLUME
012448          BCC      ROOT1            ; USER SAID S/HE DID-- CHECK IT
012449          LDA      #VNFERR           ; REPORT VOLUME NOT FOUND ERR IF REFUSE TO INSERT
012450 SRITZ      RTS
012451 *
012452          PAGE
012453 ROOT3      LDY      #$F              ; (NOTE: X CONTAINS THE LENGTH OF THE ROOT NAME)
012454 ROOTINFO    LDA      GBUF+HCRDT+3,Y  ; SAVE HEADER INFO.

```

```

012455          STA          V.STATUS,Y
012456          DEY
012457          BNE          ROOTINFO          ; LOOP TIL ALL 15 BYTES MOVED
012458          LDA          H.FCNT
012459          STA          ENTCNTL
012460          LDA          H.FCNT+1
012461          STA          ENTCNTH
012462          TXA          ; NOW THAT ROOT IS IDENTIFIED, ADJUST
012463          SEC          ; PATH NAME POINTER TO NEXT NAME IN THE PATH
012464          ADC          PATHNML
012465          STA          PATHNML
012466          CLC          ; INDICATE NO ERROR
012467  ROOTERR      RTS
012468          *
012469          *
012470  CHKROOT      LDY          #0          ; GET LENGTH OF NAME
012471          LDA          (PATHNML),Y
012472          TAY
012473          TAX          ; SAVE IN X FOR LATTER ADJUSTMENT TO PATH POINTER
012474          EOR          GBUF+4
012475          AND          #$F          ; DOES PATHNAME HAVE SAME LENGTH AS DIRECTORY NAME?
012476          BNE          NOTROOT          ; BRANCH IF NOT
012477  CKROOT1      LDA          (PATHNML),Y          ; COMPARE CHARACTER BY CHARACTER
012478          CMP          GBUF+4,Y
012479          BNE          NOTROOT
012480          DEY
012481          BNE          CKROOT1          ; LOOP UNTIL ALL CHARACTERS MATCH
012482  NOTROOT      RTS
012483          *
012484          PAGE
012485  FINDVOL      LDA          #VCB/256          ; SEARCH VCB FOR VOLUME NAME
012486          STA          VCBPTR+1
012487          LDA          #0
012488          STA          D.DEV
012489          STA          VCBPTR
012490  FNDVOL1      PHA          ; SAVE LAST SEARCH POSITION
012491          TAX
012492          LDY          #0          ; (INDEX TO PATHNAME POINTER)
012493          LDA          VCB,X          ; GET LENGTH OF VOLUME NAME TO COMPARE
012494          BEQ          NXTVCB          ; BRANCH IF VCB ENTRY IS EMPTY
012495          CMP          (PATHNML),Y          ; ARE NAMES OF SAME LENGTH?
012496          BNE          NXTVCB          ; NO, INDEX NEXT VCB
012497          CLC          ; SCAN NAME BACKWARDS
012498          TAY
012499          TXA
012500          ADC          VCB,X
012501          TAX          ; NOW BOTH INDEXES POINT TO LAST CHARACTER OF THE NAMES TO COMPARE
012502  VOLNAM      LDA          (PATHNML),Y
012503          CMP          VCB,X
012504          BNE          NXTVCB

```

```

012505          DEX
012506          DEY
012507          BNE          VOLNAM          ; CHECK ALL CHARACTERS
012508          PLA          ; SINCE A MATCH IS FOUND
012509          STA          VCBPTR          ; SET UP INDEX TO VCB ENTRY
012510          TAX
012511          LDA          VCB+VCBSWAP,X   ; BRANCH IF
012512          BEQ          FOUNDEVOL       ; VOLUME NOT SWAPPED
012513          JSR          SWAPIN          ; IF USER REALLY WANTS IT, THEN BRING IN IF SWAPPED
012514          BCC          FOUNDEVOL       ; BRANCH IF SUCCESS
012515          LDA          #XIOERROR        ; USER REFUSES TO MOUNT
012516          RTS
012517          FOUNDEVOL          CLC          ; INDICATE VOLUME FOUND
012518          RTS
012519          *
012520          NXTVCB          PLA          ; GET CURRENT INDEX AGAIN.
012521          CLC
012522          ADC          #VCBSIZE         ; VCB ENTRY LENGTH.
012523          BCC          FNDVOL1         ; BRANCH IF THERE IS ANOTHER TO CHECK
012524          RTS          ; RETURN WITH CARRY SET TO SHOW FAILURE.
012525          PAGE
012526          *
012527          *
012528          LOOKVOL          LDX          #12          ; (1) COUNT+(12)DEVICE LIST
012529          LOOKVOL1          LDA          BLKDLST,X   ; EXTRN
012530          STA          SCRTRCH,X       ; MY CHANGEABLE COPY
012531          DEX
012532          BPL          LOOKVOL1         ; WORK BACKWARDS SO
012533          STA          TOTDEVS         ; ENTRY ZERO IS TOTAL DEVICES LISTED
012534          INX          ; MAKE XREG = ZERO
012535          LOKDEV1          INX
012536          STX          SCRTRCH
012537          LDA          SCRTRCH,X
012538          CMP          D.DEV
012539          BEQ          NXTDEV          ; DON'T LOOK AGAIN ON A DRIVE THAT HAS BEEN CHECKED
012540          STA          DEVNUM          ; CHECK FOR DEVICE ALREADY LOGGED IN A VCB
012541          JSR          DEVVCB          ; (CARRY CLEAR IF IT'S THERE)
012542          BCC          LOKVOL1
012543          LDA          #0          ; FIND A FREE VCB TO LOG THIS GUY IN
012544          ENTVCB          TAX          ; INDEX TO NEXT VCB ENTRY
012545          LDA          VCB,X
012546          BEQ          FREEVCB         ; FOUND A FREE SPOT.
012547          TXA          ; NOW INDEX TO NEXT, AND KEEP LOOKIN
012548          CLC
012549          ADC          #VCBSIZE         ; (EACH VCB ENTRY IS 32 BYTES)
012550          BCC          ENTVCB          ; BRANCH IF MORE TO FIND
012551          LDA          #0
012552          ENTVCB2          EQU          *          ; SEE IF WE CAN REPLACE A DEVICE
012553          TAX
012554          LDA          VCB+VCBSTAT,X   ; VCB HAS FILES OPEN?

```



```

012555          BEQ          FREEVCB          ; NO, USE IT!
012556          TXA
012557          CLC
012558          ADC          #VCBSIZE         ; SEARCH NEXT VCB ENTRY
012559          BCC          ENTVCB2
012560          RTS
012561          *
012562  CHKVLOG      LDY          #0           ; MAKE SURE VOLUME WAS ACTUALLY LOGGED IN
012563          LDA          (VCBPTR),Y
012564          BNE          FOUNDVOL        ; AH, MADE IT...
012565          LDA          #DUPVOL         ; WELL, NOT QUITE, THIS VOLUME CAN'T BE LOGGED
012566          SEC
012567          RTS
012568          PAGE
012569          *
012570  FREEVCB     STX          VCBPTR       ; NOW THIS IS THE POINTER TO A FREE VCB
012571          LDA          #2             ; ROOT DIRECTORIES ALWAYS AT BLOCK 2
012572          LDX          #0
012573          BEQ          GETROOT        ; BRANCH ALWAYS
012574  LOKVOL1     LDY          #VCBSTAT     ; MAKE SURE NO FILES ARE ACTIVE ON
012575          LDA          (VCBPTR),Y     ; THE VOLUME BEFORE LOGGING IT IN.
012576          BMI          SNSWIT        ; BRANCH IF FILES ACTIVE
012577          LDY          #VCBROOT+1     ; GET ADDRESS OF ROOT DIRECTORY
012578          LDA          (VCBPTR),Y     ; HIGH FIRST.
012579          TAX
012580          DEY
012581          LDA          (VCBPTR),Y     ; THEN LOW.
012582  GETROOT     JSR          GETROT0
012583          BCC          LOKVOL2        ; BRANCH IF SUCCESSFULLY READ.
012584          LDA          #0             ; OTHERWISE, TAKE THIS DEVICE OUT OF VCB
012585          TAY
012586          STA          (VCBPTR),Y     ; (VOLUME 'OFF LINE')
012587          BEQ          NXTDEV         ; BRANCH ALWAYS
012588          *
012589  LOKVOL2     JSR          LOGVCB       ; GO UPDATE VCB TO INCLUDE CURRENT VOLUME INFO
012590          BCS          NXTDEV         ; IF NOT A SOS DISKETTE, SKIP TO NEXT DEVICE
012591          JSR          CHKROOT        ; GO COMPARE TO SEE IF WE FOUND WHAT WE'RE
012592          BEQ          CHKVLOG        ; LOOKING FOR...
012593          *
012594  NXTDEV     LDX          SCRTECH       ; LOOK AT OTHER DEVICES?
012595          CPX          TOTDEVS
012596          BCC          LOKDEV1        ; YES.
012597          LDA          #VNFERR        ; REPORT VOLUME NOT FOUND.
012598          RTS
012599          *
012600  SNSWIT     EQU          *           ; SENSE DSWITCH
012601          LDY          #VCBDEV
012602          LDA          (VCBPTR),Y
012603          STA          DEVNUM         ; MAKE SURE DEVICE NUMBER IS CURRENT
012604          JSR          TWRPROT1       ; USES DEVNUM

```

```

012605      LDA      DSWGLOB      ; DISK SWITCH GLOBAL
012606      BEQ      NXTDEV      ; BRANCH IF NO DISK SWITCH
012607      JSR      VERFYVOL     ; COMPARES VCBPTR VS. DEVNUM CONTENTS
012608      BCC      NXTDEV      ; BRANCH IF DISK HAS NOT BEEN SWITCHED
012609      JSR      CHKROOT     ; COMPARES PATHNML VS. GBUF
012610      BNE      NXTDEV      ; IGNORE IF NOT WHAT WE ARE LOOKING FOR
012611      LDX      #0          ; LOOK FOR FREE
012612      JSR      SNSWIT1
012613      BCS      NXTDEV      ; ANY ERRORS LOGGING IN THE NEW VOLUME
012614      JMP      CHKVLOG     ; MAKE SURE THE NEW VOLUME IS LOGGED
012615 SNSWIT1  LDA      VCB,X      ; VCB ENTRY
012616      BEQ      SNSWIT2     ; BRANCH IF FOUND
012617      TXA
012618      CLC
012619      ADC      #VCBSIZE    ; LOOK AT NEXT VCB AREA
012620      TAX
012621      BCC      SNSWIT1
012622      RTS                ; CAN'T BE LOGGED IN!
012623 SNSWIT2  LDA      #0
012624      STA      DUPLFLAG    ; TURN OFF DUPLICATE VOLUME FLAG
012625      STX      VCBPTR
012626      JSR      LOGVCB1     ; PARTIALLY LOG IN THE NEW VOLUME
012627      BCS      NONSOS      ; CS MEANS NONSOS ERROR
012628      LDA      DUPLFLAG    ; WAS IT A DUPLICATE VOLUME?
012629      BNE      SNSWIT6     ; BRANCH IF YES
012630      LDY      #VCBSWAP    ; BY MAKING SWAP BYTE NON ZERO
012631      LDA      #1
012632      STA      (VCBPTR),Y   ; SO SWAPOUT WON'T AFFECT
012633      LDA      DEVNUM      ; A REG PASSES DEVNUM TO SWAPOUT
012634      JSR      SWAPOUT     ; OLD ACTIVE MOUNT MUST BE SWAPPED
012635      BCC      SNSWIT3
012636      LDA      #XIOERROR   ; USER REFUSED TO REPLACE OLD VOLUME
012637      RTS
012638 SNSWIT3  LDY      #VCBSWAP ; NOW LOG IN THE NEW ALL THE WAY
012639      LDA      #0
012640      STA      (VCBPTR),Y
012641 SNSWIT4  JSR      VERFYVOL ; DON'T BOTHER TO ASK IF NEW VOLUME IS ALREADY MOUNTED
012642      BCC      SNSWIT5     ; BRANCH IF NEW VOLUME ON LINE
012643      JSR      USRREQ      ; ASK USER TO REMOUNT NEW VOLUME
012644      BCC      SNSWIT4     ; USER SAYS THEY DID: CHECK IT OUT
012645      LDA      #VNFERR
012646 SNSWIT5  RTS
012647 SNSWIT6  LDA      #DUPVOL
012648      SEC
012649      RTS
012650      PAGE
012651      *
012652 NONSOS   LDA      #NOTSOS  ; TELL EM IT'S NOT A SOS DISK (COULD BE PASCAL)
012653      RTS                ; CARRY SHOULD ALREADY BE SET
012654      *

```

```

012655 *
012656 DEVVCB      LDA      #0          ; SCAN VCB FOR DEVICE SPECIFIED IN 'DEVNUM'
012657 DVCB1      TAX          ; FIRST TEST FOR VALID VCB.
012658          LDA      VCB,X
012659          BEQ      DVCB2
012660          LDA      VCB+VCBSWAP,X    ; SWAPPED VOLUMES DON'T COUNT
012661          BNE      DVCB2          ; AS LOGGED IN
012662          LDA      VCB+VCBDEV,X    ; GET DEVICE NUMBER
012663          CMP      DEVNUM         ; TEST AGAINST REQUESTED DEVICE
012664          BEQ      FOUNDEV       ; YES, SET UP A POINTER TO IT
012665 DVCB2      TXA          ; BUMP TO NEXT VCB
012666          CLC
012667          ADC      #VCBSIZE
012668          BCC      DVCB1          ; BRANCH IF MORE TO LOOK AT.
012669          RTS                    ; RETURN CARRY SET TO INDICATE NOT FOUND
012670 *
012671 TSTDUPVOL   LDX      VCBPTR      ; PRESERVE CURRENT ADDR OF FREE VCB
012672          LDA      #0          ; LOOK FOR A CURRENTLY LOGGED ON VOLUME OF THE SAME NAME.
012673 TSDUPV1    STA      VCBPTR
012674          JSR      CMPVCB
012675          BCS      TSDUPV2        ; BRANCH IF NO MATCH.
012676          LDY      #VCBSTAT      ; TEST FOR ANY OPEN FILES.
012677          LDA      (VCBPTR),Y
012678          BMI      FOUNDDUP       ; TELL THE SUCKER HE CAN'T LOOK AT THIS VOLUME!
012679          LDA      #0          ; TAKE DUPLICATE OFF LINE IF NO OPEN FILES.
012680          TAY
012681          STA      (VCBPTR),Y
012682          BEQ      NODUPVOL       ; RETURN THAT ALL IS OK TO LOG IN NEW.
012683 TSDUPV2    LDA      VCBPTR
012684          CLC
012685          ADC      #VCBSIZE        ; BUMP TO NEXT ENTRY.
012686          BCC      TSDUPV1
012687 NODUPVOL   EQU      *
012688 FOUNDEV     CLC
012689 FNDDUP1    STX      VCBPTR
012690          RTS
012691 *
012692 FOUNDDUP    STA      DUPLFLAG      ; A DUPLICATE HAS BEEN DETECTED.
012693          SEC                    ; INDICATE ERROR
012694          LDA      VCBPTR          ; SAVE ADDRESS OF DUPLICATE
012695          STA      VCBENTRY
012696          BCS      FNDDUP1        ; BRANCH ALWAYS TAKEN
012697          PAGE
012698 *
012699 *
012700 LOGVCB     LDY      #VCBNML      ; IS THIS A PREVIOUSLY LOGGED IN VOLUME
012701          LDA      (VCBPTR),Y    ; (ACC=0?)
012702          BEQ      LOGVCB1        ; NO, GO AHEAD AND PREPARE VCB.
012703          JSR      CMPVCB        ; DOES VCB MATCH VOLUME READ?
012704          BCC      VCBLOGD       ; YES, DON'T DISTURB IT.

```

```

012705 LOGVCB1      LDA      #0              ; ZERO OUT VCB ENTRY
012706             LDY      #VCBSIZE-1
012707 ZERVCB      STA      (VCBPTR),Y
012708             DEY
012709             BPL      ZERVCB
012710             JSR      TSTSOS          ; MAKE SURE IT'S A SOS DISKETTE.
012711             BCS      VCBLOGD      ; IF NOT, RETURN CARRY SET.
012712             JSR      TSTDUPVOL     ; FIND OUT IF A DUPLICATE WITH OPEN FILES ALREADY EXISTS
012713             BCS      NOTLOG0
012714             LDA      GBUF+4        ; MOVE VOLUME NAME TO VCB
012715             AND      #$F          ; STRIP ROOT MARKER
012716             TAY
012717             PHA
012718 MOVOLNM      LDA      GBUF+4,Y
012719             STA      (VCBPTR),Y
012720             DEY
012721             BNE      MOVOLNM
012722             PLA                    ; GET LENGTH AGAIN
012723             STA      (VCBPTR),Y    ; SAVE THAT TOO.
012724             LDY      #VCBDEV      ; SAVE DEVICE NUMBER ALSO.
012725             LDA      DEVNUM
012726             STA      (VCBPTR),Y
012727             JSR      CLEARBMS      ; MARKS THIS DEVICES OLD BITMAPS AS INVALID (A REG PASSED)
012728             LDA      GBUF+VTBLK+4 ; AND TOTOL NUMBER OF BLOCKS ON THIS UNIT,
012729             LDY      #VCBTBLK
012730             STA      (VCBPTR),Y
012731             LDA      GBUF+VTBLK+5
012732             INY
012733             STA      (VCBPTR),Y
012734             LDY      #VCBROOT
012735             LDA      BLOKNML      ; AND ADDRESS OF ROOT DIRECTORY
012736             STA      (VCBPTR),Y
012737             INY
012738             LDA      BLOKNMH
012739             STA      (VCBPTR),Y
012740             LDY      #VCBDMAP
012741             LDA      GBUF+VBMAP+4  ; AND LASTLY, THE ADDRESS
012742             STA      (VCBPTR),Y   ; OF THE FIRST BITMAP
012743             LDA      GBUF+VBMAP+5
012744             INY
012745             STA      (VCBPTR),Y
012746             CLC                    ; INDICATE THAT IT WAS LOGGED IF POSSIBLE.
012747 VCBLOGD      RTS
012748 NOTLOG0       JMP      NOTLOG1
012749             PAGE
012750 CMPVCB        LDA      GBUF+4        ; COMPARE VOLUME NAME IN VCB
012751             AND      #$F
012752             LDY      #VCBNML      ; WITH NAME IN DIRECTORY
012753             CMP      (VCBPTR),Y    ; ARE THEY SAME LENGTH
012754             BNE      NOTSAME

```

```

012755          TAY
012756 VCBCMP1  LDA      GBUF+4,Y
012757          CMP      (VCBPTR),Y
012758          BNE      NOTSAME
012759          DEY
012760          BNE      VCBCMP1
012761          CLC                      ; INDICATE MATCH.
012762          RTS
012763          *
012764 VERFYVOL  LDA      #0              ; READ IN ROOT DIRECTORY HEADER.
012765          LDA      #2
012766          JSR      GETROT0
012767          BCS      NOVRFY1          ; PASS BACK WHATEVER OTHER ERROR OCCURS.
012768          JSR      CMPVCB          ; TEST ROOT WITH VOLUME NAME IN VCB.
012769          BCC      NOVRFY          ; BRANCH IF ROOT MATCHES VCB
012770          LDA      #0              ; OTHERWISE, PASS BACK FOREIGN VOLUME ERROR (SOS OR UCSD)
012771 NOVRFY     RTS                      ; RETURN RESULTS IN CARRY.
012772 NOVRFY1    LDA      #VNFERR        ; NOTHING IN DRIVE
012773          RTS
012774          *
012775 GETROT0    STA      BLOKNML
012776          STX      BLOKNMH          ; STORE ADDRESS AND READ IN ROOT
012777          JSR      RDGBUF
012778          BCC      RETROT2          ; BRANCH IF SUCCESSFULLY READ.
012779 NOTSAME    EQU      *
012780          SEC                      ; INDICATE ERROR
012781 RETROT2    RTS
012782          *
012783 NOTLOG1    LDA      VCBPTR          ; LOAD THE VCB ADDRESS
012784          LDA      VCBENTRY          ; OF THE DUPLICATE VOLUME
012785          STA      VCBPTR
012786          STX      VCBENTRY          ; AND SAVE THE FREE VCB SPACE ADDR
012787          LDY      #VCBDEV          ; IS DUPLICATE ON SAME DEVICE?
012788          LDA      DEVNUM
012789          CMP      (VCBPTR),Y
012790          BNE      NOTLOG2          ; BRANCH IF NOT
012791          JSR      SWAPIN           ; SWAP IN IF NECESSARY
012792          LDA      #0
012793          STA      DUPLFLAG          ; NO MORE DUPLICATE VOLUME STATUS
012794          LDA      VCBPTR          ; MAKE CHKROOT WORK IN A MOMENT
012795          STA      PATHNML          ; THIS IS INCREDIBLY GROSS
012796          ; BUT IS A RESULT OF MAKING VOLUME A SPECIAL
012797          ; CASE OF SEARCHING ALL DEVICES FOR
012798          ; A KNOWN VOLUME
012799          CLC
012800          RTS
012801 NOTLOG2    LDA      VCBENTRY          ; REACH HERE IF REAL DUPLICATE VOLUME
012802          STA      VCBPTR          ; RESOTRE FREE VCB PTR
012803          CLC
012804          RTS                      ; DUPLICATE VOLUME PRETENDS TO BE NO ERROR

```

```

012805          PAGE
012806          *
012807  TSFRBLK   LDY      #VCBTFRE+1
012808          LDA      (VCBPTR),Y          ; FIND OUT IF ENOUGH FREE BLOCKS
012809          DEY                      ; ARE AVAILABLE TO ACCOMODATE REQUEST.
012810          ORA      (VCBPTR),Y          ; BUT FIRST FIND OUT IF WE GOT A PROPER COUNT FOR THIS VOLUME.
012811          BNE      CMPFREB           ; BRANCH IF COUNT IS NON-ZERO
012812          DEY                      ; IF ZERO, THEN COUNT MUST BE TAKEN
012813          LDA      (VCBPTR),Y          ; GET HIGH TOTAL BLKS
012814          TAX                      ; SAVE IT
012815          DEY                      ; GET LOW
012816          LDA      (VCBPTR),Y          ; TOTAL BLKS
012817          BNE      TSFR01
012818          DEX                      ; ADJUST FOR BITMAP BLOCK BOUNDARY
012819  TSFR01   TXA
012820          LSR      A                  ; DIVIDE BY 16. THE RESULT IS THE NUMBER
012821          LSR      A                  ; OF BIT MAPS TO BE SEARCHED.
012822          LSR      A
012823          LSR      A
012824          STA      BMCNT              ; SAVE IT.
012825          LDA      #0                 ; START COUNT AT ZERO.
012826          STA      SCRTCH
012827          STA      SCRTCH+1
012828          LDA      #$FF               ; MARK 'FIRST FREE' TEMP AS UNKNOWN
012829          STA      NOFREE
012830          LDY      #VCBDEV           ; MAKE SURE BIT MAP IS UP TO DATE
012831          LDA      (VCBPTR),Y        ; GET DEVICE NUMBER
012832          TAX                      ; PASS TO 'UPBMAP' IN X
012833          JSR      UPBMAP             ; (NOTHING HAPPENS IF IT DON'T HAFTA.)
012834          BCS      TFBERR            ; BRANCH IF WE GOT TROUBLE,
012835          LDY      #VCBDMAP           ; GET ADDRESS OF FIRST BIT MAP.
012836          LDA      (VCBPTR),Y
012837          STA      BLOKNML
012838          INY                      ; (FOR HIGH ADDRESS)
012839          LDA      (VCBPTR),Y
012840          STA      BLOKNMH
012841  BMAPRD   JSR      RDGBUF           ; USE G(ENERAL)BUFF(ER) FOR TEMPORARY
012842          BCS      TFBERR            ; SPACE TO COUNT FREE BLOCKS (BITS)
012843          JSR      COUNT              ; GO COUNT EM
012844          DEC      BMCNT              ; WAS THAT THE LAST BIT MAP?
012845          BMI      CHGVCB             ; IF SO, GO CHANGE FCB TO AVOID DOING THIS AGAIN!
012846          INC      BLOKNML           ; NOTE: THE ORGANIZATION OF THE BIT MAPS
012847          BNE      BMAPRD            ; ARE CONTIGUOUS FOR SOS VERSION 0
012848          INC      BLOKNMH           ; IF SOME OTHER ORGANIZATION IS IMPLEMENTED, THIS CODE
012849          JMP      BMAPRD            ; MUST BE CHANGED!
012850          PAGE
012851          *
012852  CHGVCB    LDY      #VCBCMAP         ; MARK WHICH BLOCK HAD FIRST FREE SPACE
012853          LDA      NOFREE
012854          BMI      DSKFULL             ; BRANCH IF NO FREE SPACE WAS FOUND.

```

```

012855      STA      (VCBPTR),Y
012856      LDY      #VCBTFRE+1      ; UPDATE THE FREE COUNT.
012857      LDA      SCRATCH+1      ; GET HIGH COUNT BYTE
012858      STA      (VCBPTR),Y      ; UPDATE VOLUME CONTROL BLOCK.
012859      DEY
012860      LDA      SCRATCH
012861      STA      (VCBPTR),Y      ; AND LOW BYTE TOO...
012862  CMPFREE  LDA      (VCBPTR),Y      ; COMPARE TOTAL AVAILABLE
012863      SEC
012864      SBC      REQL      ; FREE BLOCKS ON THIS VOLUME.
012865      INY
012866      LDA      (VCBPTR),Y
012867      SBC      REQH
012868      BCC      DSKFULL
012869      CLC
012870      RTS
012871  DSKFULL  LDA      #OVRERR
012872      SEC
012873  TFBERR   RTS
012874      PAGE
012875      *
012876  COUNT   LDY      #0      ; BEGIN AT THE BEGINNING.
012877  FRCONT  LDA      GBUF,Y      ; GET BIT PATTERN
012878      BEQ      FRCNT1      ; DON'T BOTHER COUNTING NOTHIN'
012879      JSR      CNTFREE
012880  FRCNT1  LDA      GBUF+$100,Y      ; DO BOTH PAGES WITH SAME LOOP
012881      BEQ      FRCNT2
012882      JSR      CNTFREE
012883  FRCNT2  INY
012884      BNE      FRCONT      ; LOOP TILL ALL 512 BYTES COUNTED
012885      BIT      NOFREE      ; HAS FIRST BLOCK WITH FREE SPACE BEEN FOUND YET?
012886      BPL      FRCNT3      ; BRANCH IF IT HAS.
012887      LDA      SCRATCH      ; TEST TO SEE IF ANY BLOCKS WERE COUNTED
012888      ORA      SCRATCH+1
012889      BEQ      FRCNT3      ; BRANCH IF NONE COUNTED.
012890      LDY      #VCBTBLK+1
012891      LDA      (VCBPTR),Y      ; SHOW THIS MAP IS FIRST WITH FREE SPACE
012892      SEC      ; CORRECT FOR EXACT MULTIPLES OF $1000
012893      SBC      #$01
012894      LSR      A
012895      LSR      A
012896      LSR      A
012897      LSR      A
012898      SEC      ; SUBTRACT COUNTDOWN FROM TOTAL BIT MAPS
012899      SBC      BMCNT
012900      STA      NOFREE
012901  FRCNT3  RTS
012902      *
012903  CNTFREE  ASL      A      ; COUNT THE NUMBER OF BITS IN THIS BYTE.
012904      BCC      CFREE1

```

```
012905          INC      SCRTCH
012906          BNE      CFREE1
012907          INC      SCRTCH+1
012908 CFREE1     TAX
012909          BNE      CNTFREE          ; LOOP UNTIL ALL BITS COUNTED.
012910          RTS
012911
012912          CHN      ALLOC,4,1
012913
012914 *****
012915 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: FNDFIL
012916 *****
012917
012918
012919
```



```

012920 =====
012921 DOCUMENT :SOS1.3.3of5.THREE:SOS.PATH.TEXT
012922 =====
012923
012924 *****
012925 * APPLE /// SOS 1.3 SOURCE CODE FILE: PATH
012926 *****
012927 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
012928
012929 PAGE
012930 *
012931 *
012932 *
012933 BFMGR      LDX      COMMAND      ; WHAT CALL?
012934 *
012935 *
012936 *
012937      LDA      DISPTCH,X      ; TRANSLATE INTO COMMAND ADDRESS
012938      ASL      A      ; (BIT 7 INDICATES IT'S GOT A PATHNAME TO PREPROCESS)
012939      STA      CMDTEMP
012940      AND      #$3F      ; (BIT 6 IS REFNUM PREPROCESS, 5 IS FOR TIME, SO STRIP EM.)
012941      TAX
012942      LDA      CMDTABLE,X      ; MOVE ADDRESS FOR INDIRECT JUMP.
012943      STA      CMDADR
012944      LDA      CMDTABLE+1,X      ; (HIGH BYTE)
012945      STA      CMDADR+1
012946      LDA      #<VCB
012947      STA      VCBPTR+1      ; INSURE DEFAULT HI ADDRESS TO VCB BEFORE CALLS
012948      LDA      #BKBITVAL      ; INIT "BACKUP BIT FLAG"
012949      STA      BKBITFLG      ; TO SAY "FILE MODIFIED"
012950      LDY      #MAXTEMPS      ; ZERO OUT SISTER PAGE FOR TEMPS
012951      LDA      #0
012952      STA      SERR      ; MAKE GLOBAL ERROR SAY "NONE"
012953      STA      DSWGLOB      ; "DISK SWITCH GLOBAL"
012954      STA      DUPLFLAG      ; "DUPLICATE VOLUME ON LINE"
012955      STA      CFLAG      ; SET "CREATE" TO NO
012956      STA      BLOKSAVE
012957      STA      BLOKSAVE+1      ; SET PARENT DIRECTORY TO NULL
012958 CLRSIS      STA      SISTEMPS,Y
012959      DEY
012960      BPL      CLRSIS      ; CARRY IS UNDISTURBED BY THIS LOOP
012961      BCC      NOPATH
012962      JSR      SETPATH      ; GO PROCESS PATHNAME BEFORE CALLING COMMAND
012963      BCS      ERRORSYS      ; BRANCH IF BAD NAME.
012964 NOPATH      ASL      CMDTEMP      ; TEST FOR REFNUM PREPROCESSING
012965      BCC      NOPREREF
012966      JSR      FINDFCB      ; GO SET UP POINTERS TO FCB AND VCB OF THIS FILE.
012967      BCS      ERRORSYS      ; BRANCH IF ANY ERRORS ARE ENCOUNTERED.
012968 NOPREREF      ASL      CMDTEMP      ; LASTLY CHECK FOR NECESSITY OF TIME STAMP.

```

```

012969          BCC      TSWVRFY
012970          LDX      #DATELO          ; PASS Z PAGE ADDRESS OF WHERE TO RETURN DATE/TIME
012971          JSR      DATETIME        ; (NO ERROR POSSIBLE)
012972 TSWVRFY   LDX      COMMAND        ; TEST FOR NECESSITY OF VOLUME VERIFICATION
012973          LDA      #PREPATH+PREREF+PRETIME ; TO ENSURE VCB IS SET
012974          AND      DISPTCH,X
012975          BEQ      EXECUTE
012976          LDY      #VCBSTAT
012977          LDA      (VCBPTR),Y
012978          AND      #DSWITCH        ; WAS THE VOLUME PREVIOUSLY SWITCHED?
012979          BEQ      EXECUTE
012980          DEY          ; GET DEVICE NUMBER
012981          LDA      (VCBPTR),Y
012982          STA      DEVNUM
012983 DVERIFY   JSR      VERIFYVOL      ; SEE IF PROPER VOLUME NOW ON LINE
012984          BCC      CLRDSWT        ; BRANCH IF YES
012985          JSR      USRREQ         ; OTHERWISE REQUEST IT BE PUT ON LINE
012986          BCC      DVERIFY        ; USER SEZ S/HE DID: CHECK IT OUT
012987          LDA      #VNFERR        ; VOLUME NOT FOUND IF USER REFUSES
012988          BNE      ERRORSYS      ; REPORT ERROR (BRANCH ALWAYS)
012989 CLRDSWT  LDY      #VCBSTAT        ; GET VOLUME
012990          LDA      (VCBPTR),Y    ; STATUS
012991          AND      #$FF-DSWITCH    ; TURN OFF DISK SWITCH
012992          STA      (VCBPTR),Y    ; SO WE WON'T VERIFY NEXT TIME
012993 EXECUTE   JSR      GOCMD         ; EXECUTE COMMAND
012994          BCC      GOODOP         ; BRANCH IF SUCCESSFUL
012995          CMP      #XDISKSW       ; DISK SWITCH?
012996          BNE      ERRORSYS      ; NO, REPORT SOME OTHER
012997          LDY      #VCBSTAT        ; MARK VCB WITH SWITCH
012998          LDA      (VCBPTR),Y
012999          AND      #$FF-DSWITCH    ; TO ENSURE VOLUME VERIFIED
013000          BPL      ERRCMD        ; NO FILES OPEN SO DSWITCH CANT APPLY
013001          ORA      #DSWITCH
013002 ERRCMD   STA      (VCBPTR),Y
013003          JMP      BFMGR         ; TRY THE COMMAND AGAIN
013004          *
013005 ERRORSYS  JSR      SYSERR
013006 GOODOP   RTS          ; GOOD RETURN
013007          *
013008 GOCMD    JMP      (CMDADR)
013009          *
013010          PAGE
013011          *
013012 CMDTABLE EQU      *
013013          DW      CREATE
013014          DW      DESTROY
013015          DW      RENAME
013016          DW      SETINFO
013017          DW      GETINFO
013018          DW      VOLUME

```

```

013019          DW      SETPREFIX
013020          DW      GETPREFIX
013021          DW      OPEN
013022          DW      NEWLINE
013023          DW      READ
013024          DW      WRITE
013025          DW      CLOSE
013026          DW      FLUSH
013027          DW      SETMARK
013028          DW      GETMARK
013029          DW      SETEOF
013030          DW      GETEOF
013031          *
013032  DISPTCH  EQU      *
013033          DFB      PREPATH+PRETIME+0      ; CREATE
013034          DFB      PREPATH+PRETIME+1      ; DESTROY
013035          DFB      PREPATH+PRETIME+2      ; RENAME
013036          DFB      PREPATH+PRETIME+3      ; SETINFO
013037          DFB      PREPATH+4              ; GETINFO
013038          DFB      5                      ; VOLUME
013039          DFB      6                      ; SETPREFIX, PATHNAME MOVED TO PREFIX BUFFER
013040          DFB      7                      ; GETPREFIX
013041          DFB      PREPATH+8              ; OPEN
013042          DFB      PREREF+$9              ; NEWLINE
013043          DFB      PREREF+$A              ; READ
013044          DFB      PREREF+$B              ; WRITE
013045          DFB      PRETIME+$C            ; CLOSE
013046          DFB      PRETIME+$D            ; FLUSH, REFNUM MAY BE ZERO TO FLUSH ALL.
013047          DFB      PREREF+$E              ; SETMARK
013048          DFB      PREREF+$F              ; GETMARK
013049          DFB      PREREF+$10             ; SET EOF
013050          DFB      PREREF+$11             ; GET EOF
013051          *
013052          PAGE
013053          *
013054  SETPATH   LDA      C.PATH                ; FOR A REGULAR PATHNAME,
013055          STA      TPATH                    ; SET UP TEMP POINTER TO PROCESS
013056          LDA      C.PATH+1                ; PATHNAME AND CHECK FOR SYNTAX ERRORS
013057          STA      TPATH+1
013058          LDA      SISPATH
013059          STA      SISTPATH                ; (LEAVE CALL PARAMETERS ALONE!)
013060          * DROP INTO 'SYNPATH'
013061          *
013062  SYNPATH   LDA      #>PATHBUF            ; SET UP DEFAULT ADDRESS FOR
013063          STA      PATHNML                  ; SYNTAXED PATHNAME -
013064          STA      WRKPATH                  ; LENGTH, NAME, LENGTH, NAME, ETC...
013065          LDA      #<PATHBUF
013066          STA      PATHNMH
013067          STA      WRKPATH+1                ; (ASSUMES FULL PATHNAME, NO PREFIX).
013068          LDX      #0                       ; USE INDEXED INDIRECT FOR SOURCE PATHNAME

```

```

013069          TXA                      ; SET BEGINNING OF PATH
013070          STA      (PATHNML,X)      ; TO ZERO TO INDICATE NOTHING PROCESSED.
013071          TAY
013072          LDA      (TPATH,X)        ; GET TOTAL LENGTH OF SOURCE PATHNAME
013073          BMI      ERRSYN
013074          BEQ      ERRSYN
013075          STA      PATHCNT          ; (THIS IS USED AS A 'COUNT-DOWN')
013076          JSR      INCTPTH         ; INCREMENT SOURCE POINTER
013077          LDA      (TPATH,X)        ; GET FIRST CHARACTER OF PATHNAME
013078          CMP      #DLIMIT         ; IS IT A FULL PATHNAME (NO PREFIX)?
013079          BEQ      BUMPATH         ; YES, WE'RE READY TO DO IT.
013080          CMP      #D$2E           ; IS IT A DRIVE NAME '.'?
013081          BNE      ADPREFIX        ; NO, ADD PREFIX TO BEGINNING
013082  DRIVENAM  LDA      (TPATH,X)        ; MOVE DRIVE NAME FOR VOLUME CALL
013083          CMP      #DLIMIT         ; HAVE WE MOVED ENTIRE NAME?
013084          BEQ      PREVOLM        ; YES, PROCESS IT.
013085          INY                      ; (IF THIS IS THE FIRST, MAKE ROOM FOR LENGTH OF NAME)
013086          STA      (WRKPATH),Y
013087          JSR      INCTPTH         ; BUMP POINTER TO GIVEN NAME.
013088          DEC      PATHCNT
013089          BNE      DRIVENAM
013090          BEQ      PREVOLM1
013091  *
013092          PAGE
013093  PREVOLM    JSR      INCTPTH         ; MAKE IT SO POINTING PAST DELIMITER.
013094          DEC      PATHCNT
013095  PREVOLM1  TYA                      ; SAVE LENGTH OF DRIVE NAME.
013096          STA      (WRKPATH,X)
013097          LDA      #>PATHBUF       ; POINT AT PATHNAME BUFFER FOR DEVICE ID CALL.
013098          STA      DVNAMP
013099          LDA      #<PATHBUF
013100          STA      DVNAMP+1
013101          LDA      #0              ; MAKE VIRTUAL POINT AT SWITCHED IN BANK.
013102          STA      SISTER+DVNAMP+1
013103          JSR      SRCHDEV         ; GO IDENTIFY WHICH VOLUME
013104          BCC      PREVOLM2        ; BRANCH IF NO ERROR
013105          CMP      #VNFERR        ; WAS IT REPORTED AS 'VOLUME NOT FOUND'?
013106          BNE      SPTHERR        ; NO SOME OTHER ERROR WAS ENCOUNTERED.
013107          LDX      DUPLFLAG       ; YES, WAS IT NOT FOUND BECAUSE SOME OTHER 'OPEN' VOLUME HAS SAME NAME?
013108          BEQ      SPTHERR        ; NO, IT SIMPLY WASN'T FOUND.
013109          LDA      #DUPVOL        ; (CARRY IS SET)
013110          RTS
013111  *
013112  PREVOLM2  LDY      #0            ; (X CONTAINS AN INDEX TO VCB)
013113          LDA      VCB,X          ; GET VOLUME NAME LENGTH.
013114          STA      PATHBUF,Y
013115  SPTH2     INX                      ; MOVE VOLUME NAME INTO PATH NAME BUFFER IN
013116          INY                      ; PLACE OF DISK DEVICE NAME ('.D1' OR SIMILAR)
013117          LDA      VCB,X
013118          STA      PATHBUF,Y

```

```

013119          CPY          PATHBUF          ; HAVE ALL CHARACTERS BEEN MOVED?
013120          BNE          SPATH2
013121          LDX          #0                ; RESET X FOR INDEXING
013122          STX          PATHNML
013123          LDA          #<PATHBUF
013124          STA          PATHNMH
013125          LDA          PATHCNT          ; IS THAT ALL THERE IS?
013126          BNE          SPATH3          ; NO, MORE TO COME...
013127          CLC
013128          JMP          ENDPATH
013129          *
013130 SPATH3     INY                ; BUMP TO END OF NAME+1
013131          STY          WRKPATH          ; RESET WORKPATH POINTER TO CURRENT.
013132          LDA          #0                ; RESET PATHNAME BUFFER POINTER.
013133          LDY          #<PATHBUF
013134          BNE          NOPREFIX          ; BRANCH ALWAYS...
013135          *
013136 ERRSYN     LDA          #BADPATH        ; RETURN SYNTAX ERROR
013137 SPTHEER    SEC
013138          RTS
013139          *
013140 ADPREFIX    LDA          PFXPTR          ; GET POINTER TO BEGINNING OF THE
013141          LDY          PFXPTR+1          ; PREFIX.
013142 NOPREFIX    STA          PATHNML
013143          STY          PATHNMH          ; IF NO PRESET PREFIX, THIS IS THE SAME AS
013144          BNE          FRSTCHAR          ; PATHBUF ADDRESS. (BRANCH ALWAYS TAKEN)
013145          *
013146          PAGE
013147          *
013148 BUMPATH     DEC          PATHCNT        ; FIRST ADJUST COUNT
013149          CLC                ; (JUST IN CASE OF LAST CHARACTER)
013150          BEQ          ENDPATH          ; (MUST OF HAD TRAILING SPACES)
013151          JSR          INCTPTH
013152 FRSTCHAR    LDY          #0                ; INIT COUNT FOR THIS PORTION OF THE
013153          TYA                ; PATHNAME. ALSO PRESET LENGTH TO ZERO IN
013154          STA          (WRKPATH,X)        ; CASE OF TRAILING SPACES.
013155          LDA          (TPATH,X)        ; GET CHARACTER.
013156          AND          #$7F            ; IGNORE HIGH BIT.
013157          CMP          #$20            ; IS IT A LEADING SPACE?
013158          BEQ          BUMPATH          ; IF SO, IGNORE IT.
013159          CMP          #$5B            ; IS IT GREATER THAN (UPPER CASE) A 'Z'?
013160          BCC          ALFA1            ; NO, MAKE SURE IT'S AN ALPHA CHARACTER
013161          AND          #$5F            ; YES, ASSUME IT'S LOWER CASE, AND UPSHIFT
013162          CMP          #$5B            ; WAS IT TRULY LOWER CASE?
013163          BCS          ERRSYN          ; NO, GIVE ERROR.
013164          *
013165 ALFA1       CMP          #$41            ; IS IT LESS THAN 'A'?
013166          BCC          ERRSYN          ; YES! IT'S CRAP...
013167          BCS          SAVPATH          ; NO, IT'S GOOD. SAVE IT.
013168          *

```

```

013169 NXTCHAR      LDA      (TPATH,X)          ; GET THE NEXT CHARACTER.
013170              AND      #$7F              ; THESE CHARACTERS MAY BE ALPHA, NUMERIC,
013171              CMP      #$5B              ; OR A PERIOD - ONLY THE FIRST HAD TO BE ALPHA
013172              BCC      ALFA2             ; BRANCH IF LESS THAN 'Z'
013173              AND      #$5F              ; UPSHIFT LOWER CASE.
013174              CMP      #$5B              ; NOW IS IT VALID?
013175              BCS      ERRSYN            ; NOPE.
013176 *
013177 ALFA2         CMP      #$41              ; IS IT GREATER THAN 'A'?
013178              BCS      SAVPATH            ; YUP, IT IS WORTH SAVIN.
013179              CMP      #$3A              ; >9?
013180              BCS      TSTDLM             ; YES
013181              CMP      #$30              ; NO, <0?
013182              BCS      SAVPATH            ; NO, IT'S VALID NUMERIC.
013183 TSTDLM        CMP      #DLIMIT           ; IS IT THE DELIMITER?
013184              BEQ      ENDPATH            ; YES. CARRY SET INDICATES MORE TO COME.
013185              CMP      #$2E              ; IS IT A '.' (PERIOD)?
013186              BNE      ERRSYN            ; NO, IT'S AN ERROR (#@&##@!)
013187 SAVPATH       CLC
013188              INY
013189              STA      (WRKPATH),Y        ; BUMP NAME LENGTH
013190              DEC      PATHCNT            ; IF ZERO, THAT WAS THE LAST CHARACTER
013191              BEQ      ENDPATH            ; (CARRY CLEAR INDICATES END OF PATH)
013192              INC      TPATH              ; BUMP POINTER TO SOURCE PATHNAME.
013193              BNE      NXTCHAR
013194              INC      TPATH+1             ; HIGH ORDER, WHEN NECESSARY.
013195              BNE      NXTCHAR            ; BRANCH ALWAYS.
013196 PAGE
013197 *
013198 ENDPATH         TYA
013199              STA      (WRKPATH,X)         ; GET CURRENT NAME LENGTH
013200              BCC      LSTNAME             ; AND PUT IT IN FRONT OF NAME
013201              CMP      #$10              ; BRANCH IF THAT WAS THE LAST OF PATH
013202              BCS      ERRSYN1            ; WAS THE NAME ILLEGALLY LONG?
013203              LDY      #0
013204              SEC
013205              ADC      WRKPATH             ; YES, REPORT IT.
013206              STA      WRKPATH            ; ADJUST WORK POINTER TO END OF PREVIOUS NAME.
013207              BCC      BUMPATH            ; REPLACE OLD POINTER.
013208              LDA      #TOOOLONG          ; DO NEXT NAME.
013209              JSR      SYSDEATH           ; THIS IS A NEVER ERROR!
013210 *
013211 LSTNAME        BEQ      TSTVALD          ; (NEVER RETURNS).
013212              CMP      #$10              ; MAKE SURE LAST ISN'T TOO LONG
013213              BCS      ERRSYN1
013214              INY
013215              LDA      #0
013216 TSTVALD        STA      (WRKPATH),Y        ; PUT A ZERO AT END OF PROCESSED PATHNAME
013217              LDA      (PATHNML,X)         ; SURE THERE IS A PATHNAME
013218              BEQ      ERRSYN1            ; IF NOT, REPORT ERROR.

```

```

013219          CLC                      ; INDICATE NO ERROR.
013220          RTS
013221      *
013222  ERRSYN1      JMP      ERRSYN
013223      *
013224  INCTPTH      INC      TPATH          ; POINT AT NEXT CHARACTER
013225          BNE      INCPH1
013226          INC      TPATH+1
013227  INCPH1      RTS
013228      *
013229          PAGE
013230  SETPPREFX     JSR      SETPATH        ; CALL IS MADE HERE SO A 'NUL' PATH MAY BE DETECTED.
013231          BCC      SETPRFX1          ; BRANCH IF PATHNAME OK
013232          TAX
013233          LDY      #0                ; SAVE ERROR CODE
013234          LDA      (C.PATH),Y        ; TEST FOR A NUL PATHNAME
013235          BEQ      RESETPFX          ; BRANCH IF PREFIX TO BE RESET.
013236          TXA
013237          RTS
013238  RESETPFX     STA      PFXPTR
013239          CLC
013240          RTS
013241  SETPRFX1      LDA      PATHNML        ; MAKE SURE NAME STARTED WITH A '/' DELIMITER.
013242          BNE      ERRSYN1          ; BRANCH IF IT DID.
013243          LDY      WRKPATH          ; FIND THE END OF THE INPUT PREFIX
013244          CLC
013245          LDA      (PATHNML),Y        ; ADD LAST LOCAL NAME LENGTH TO FIND TRUE END.
013246          BNE      SETPRFX3
013247          DEY
013248          TYA
013249          BNE      SETPRFX4
013250  SETPRFX3     ADC      WRKPATH
013251          TAY
013252  SETPRFX4     EOR      #$FF          ; GET COMPLIMENT TO FIND BEGINNING ADDRESS.
013253          STA      PFXPTR          ; OF NEW PREFIX IN THE PREFIX BUFFER
013254          STA      WRKPATH          ; (PREFIX ALWAYS ENDS AT THE LAST BYTE OF BUFFER)
013255  MOVPRFX       LDA      (PATHNML),Y
013256          STA      (WRKPATH),Y      ; MOVE IN NEW PREFIX
013257          DEY
013258          BPL      MOVPRFX
013259          CLC                      ; AND WE'RE FINISHED!
013260          RTS                      ; NO ERRORS POSIBLE FROM THIS ROUTINE.
013261      *
013262          PAGE
013263      *
013264  GETPPREFX     CLC                      ; CALCULATE HOW BIG A BUFFER IS NEEDED TO
013265          LDA      PFXPTR          ; PASS THE PREFIX BACK TO THE USER.
013266          EOR      #$FF          ; (EVEN IF NO PREFIX, 1 BYTE IS NEEDED TO SHOW 0 LENGTH)
013267          ADC      #2                ; ADD 2 FOR LEADING AND ENDING "/".
013268          CMP      C.MAXPTH        ; IS THERE ENOUGH SPACE IN USER'S BUFFER?

```

```

013269          BCC      SENDPRFX          ; BRANCH IF YES
013270          LDA      #BTSERR          ; TELL USER BUFFER IS TOO SMALL.
013271          RTS
013272          *
013273 SENDPRFX    LDY      #0              ; SAVE TOTAL LENGTH OF STRING TO BE RETURNED
013274          STA      (C.PATH),Y
013275          TAY
013276          DEY                          ; DISCOUNT TRAILING DELIMITER.
013277          BEQ      NULPREFIX          ; BRANCH IF PREFIX IS SET TO NUL.
013278          INY
013279          LDX      PFXPTR            ; GET BEGINNING ADDRESS OF PREFIX AGAIN
013280          DEX
013281          STX      WRKPATH
013282          LDA      #<PATHBUF
013283          STA      WRKPATH+1
013284 SNDLMIT    LDA      #DLIMIT        ; PLACE DELIMITER BEFORE, BETWEEN, AND AFTER LOCAL NAMES.
013285          STA      (C.PATH),Y
013286 SNDPRFX1   DEY
013287          BEQ      GOTPRFX            ; BRANCH IF ALL OF PREFIX IS TRANSFERED.
013288          LDA      (WRKPATH),Y
013289          STA      (C.PATH),Y          ; ASSUME IT'S A CHARACTER.
013290          AND      #$F0              ; NOW TEST TO SEE IF IT WAS A LOCAL LENGTH.
013291          BEQ      SNDLMIT            ; BRANCH IF IT WAS.
013292          BNE      SNDPRFX1          ; GO MOVE NEXT CHAR IF IT WASN'T (ALWAYS TAKEN).
013293 NULPREFIX   TYA                          ; RETURN NUL STRING.
013294          STA      (C.PATH),Y
013295 GOTPRFX     CLC                          ; INDICATE NO ERROR.
013296          RTS
013297          PAGE
013298          *
013299 FINDFCB     LDA      FCBADDRH        ; INITIALIZE INDIRECT POINTER TO
013300          STA      FCBPTR+1            ; FILE CONTROL BLOCK (ALLOCATED WHEN SYSTEM
013301          LDA      #0                  ; WAS FIRST BOOTED).
013302          STA      FCBPTR              ; NOTE: ALWAYS STARTS ON PAGE BOUNDARY.
013303          LDA      FCBANKNM           ; SET SISTE PAGE BYTE TOO...
013304          STA      SISFCBP
013305          LDY      C.REFNUM           ; GET REQUESTED REFERENCE
013306          BMI      ERRNOTBLK         ; BRANCH IF IT'S NOT A BLOCK DEVICE REFERENCE
013307          DEY                          ; (SHOULD BE IN THE RANGE OF 1-16 BEFORE DECREMENT)
013308          CPY      #$10              ; IS IT A VALID REFNUM?
013309          BCS      REEFER             ; NO, THE USER'S SMOKIN DOPE!
013310          TYA                          ; TO FIND ASSOCIATED FILE CONTROL STUFF,
013311          ASL      A                  ; MULTIPLY (REFNUM-1) BY 32.
013312          ASL      A
013313          ASL      A
013314          ASL      A
013315          ASL      A
013316          BCC      SVFCBLO           ; BRANCH IF IT'S WITHIN FIRST HALF OF FCB
013317          INC      FCBPTR+1           ; BUMP TO SECOND HAVE (REFNUM>8)
013318 SVFCBLO    STA      FCBPTR         ; SAVE LOW ADDRESS OF REFERENCED FCB

```



```

013319          LDA      C.REFNUM          ; NOW VERIFY THAT FILE IS OPEN.
013320          LDY      #FCBREFN
013321          CMP      (FCBPTR),Y        ; SHOULD BE EQUAL!
013322          BNE     ERRNOREF          ; BRANCH IF THEY'RE NOT
013323  FNDFCBUF  LDY      #FCBBUFN        ; IT'S A LEGAL FILE, NOW SET UP
013324          LDA      (FCBPTR),Y        ; INDIRECT POINTERS TO DATA
013325  GTBUFFRS  LDX      #DATPTR         ; (AND INDEX) BUFFER(S) IN ZERO PAGE
013326          JSR     GETBUFADR         ; GET BUFFER ADDRESS UNLESS
013327          BCS     REEFER1          ; BOB HAS BEEN SMOKIN DOPE...
013328          LDA      #2                ; (ASSUME AN INDEX BLOCK BUFFER IS ALSO PRESENT)
013329          ADC     DATPTR+1
013330          STA     TINDX+1
013331          LDA     DATPTR
013332          STA     TINDX
013333          LDA     SISDATP
013334          STA     SSTIDXH
013335          LDY     #FCBDEVN
013336          LDA     (FCBPTR),Y        ; MAKE SURE DEVICE
013337          STA     D.DEV              ; NUMBER TEMPS MATCH
013338          STA     DEVNUM            ; CURRENT FILE'S DEVICE
013339          LDA     #0                ; LOOK AT ALL VOLUMES LOGGED IN
013340  FNDFVOL   TAX
013341          LDA     VCB+VCBDEV,X      ; GET VOLUMES DEVICE NUMBER
013342          CMP     (FCBPTR),Y        ; HVE WE FOUND A MATCH.
013343          BNE     FNDFV1
013344          LDY     #FCBSWAP          ; SWAP BYTES
013345          LDA     VCB+VCBSWAP,X     ; MISMATCH
013346          CMP     (FCBPTR),Y        ; MEANS FILE BELONGS
013347          BNE     FNDFV.1          ; TO ANOTHER VOLUME
013348          LDA     VCB,X              ; IS THIS AN OPEN DEVICE?
013349          BEQ     FNDFV.1          ; NO, TRY ANOTHER VOLUME
013350          JSR     FVOLFOUND        ; YES, SAVE VCB ADDRESS
013351          LDA     VCB+VCBSWAP,X     ; SWAPPED?
013352          BEQ     REEFER1          ; NO, RETURN CALMLY TO USER
013353          JSR     SWAPIN           ; YES, SWAP ME IN
013354          BCC     REEFER1          ; RETURN WITHOUT ERROR
013355          LDA     #XIOERROR        ; USER REFUSED TO MOUNT PROPER VOLUME
013356          RTS
013357          *
013358  FNDFV.1   LDY     #FCBDEVN        ; RELOAD Y WITH DEVICE INDEX
013359  FNDFV1    TXA
013360          CLC
013361          ADC     #VCBSIZE
013362          BCC     FNDFVOL            ; LOOP UNTIL FOUND
013363          LDA     #VCBERR           ; OTHERWISE DIE A SYSTEM DEATH!
013364          JSR     SYSDEATH
013365          PAGE
013366          *
013367  ERRNOREF   LDA     #0                ; DROP A ZERO INTO THIS FCB TO
013368          STA     (FCBPTR),Y        ; SHOW FREE FCB

```

```

013369 *
013370 REEFER      LDA      #BADREFNUM      ; TELL USER THAT REQUESTED REFNUM
013371           SEC                      ; IS ILLEGAL (OUT OF RANGE) FOR THIS CALL.
013372 REEFER1     RTS
013373 *
013374 ERRNOTBLK  LDA      #NOTBLKDEV      ; TELL USER THAT SPECIFIED DEVICE IS NOT A BLOCK DEVICE
013375           SEC
013376           RTS
013377 *
013378 SVCBADR     EQU      *
013379 FVOLFOUND     STX      VCBPTR
013380           LDA      #VCB/256
013381           STA      VCBPTR+1
013382           CLC                      ; INDICATE LEGAL REFNUM
013383           RTS
013384           PAGE
013385 * NAME      : GETDNUM
013386 * FUNCTION:  GET DEVICE NUMBER
013387 * INPUT     : DVNAMP SETUP
013388 * OUTPUT    : DEVNUM IN 'SCRATCH'
013389 *          : 'BPL' IF NOT BLOCK DEV
013390 *          : 'BCS' IF NO DEVICE
013391 * VOLATILE:  ALL REGS
013392 *
013393 GETDNUM      EQU      *
013394           LDA      #>SCRATCH+1        ; SET UP POINTER TO SCRATCH AREA
013395           STA      DVDNUM            ; TO RECIEVE DEVICE NUMBER.
013396           LDA      #SCRHIGH
013397           STA      DVDNUM+1
013398           LDA      #0                ; PLACE A ZERO IN BANK BYTE SINCE
013399           STA      SISTER+DVDNUM+1  ; IT'S NOT IN A BANK.
013400           STA      VCBPTR+1
013401           LDA      #4                ; THE 'GET.DNUM' COMMAND.
013402           STA      DHPCMD
013403           JSR      RPEATIO0        ; CALL BOB FOR THE INFO.
013404           RTS                      ; RETURN WITH DEVMGR CC'S
013405           PAGE
013406 *
013407 * NAME      : SRCHDEV
013408 * FUNCTION:  SEARCH FOR A VOLUME
013409 *
013410 SRCHDEV     EQU      *
013411           JSR      GETDNUM           ; GET DEVNUM
013412           BCS     VOLERR1           ; BRANCH IF ANY ERROR OTHER THAN NOTBLOCKDEV
013413           BPL     ERRNOTBLK        ; BRANCH IF NOT A BLOCK DEVICE
013414           LDA      #0                ; NOW SEARCH FOR A VOL WITH THE
013415           STA      NFOPEN           ; INIT TEMP VCB POINTER
013416 VOLOOK      TAX                      ; SAME DEVNUM AS SCRATCH
013417           LDA      VCB+VCBSTAT,X    ; ANY FILES OPEN?
013418           BNE     VLOOK00          ; BRANCH IF SOME FILE OPEN

```

```

013419          STX          NFOPEN          ; ELSE SAVE THE VCB ENTRY PTR
013420 VLOOK00    EQU          *
013421          LDA          VCB+VCBSWAP,X    ; VOLUME SWAPPED OUT?
013422          BNE          VNOTEQ         ; YES, CANT BE THE ACTIVE VOL
013423          LDA          VCB+VCBDEV,X
013424          EOR          SCRTCH+1
013425          BEQ          VLOOK0         ; BRANCH IF MATCH.
013426 VNOTEQ    LDA          VCB,X         ; IS THIS A FREE VCB?
013427          BNE          VLOOK2         ; BRANCH IF NOT FREE, OTHEWISE TAKE NEXT BRANCH.
013428 VLOOK0    EOR          VCB,X         ; TEST FOR A VOLUME NAME LENGTH
013429          BEQ          VLOOK1         ; BRANCH IF VCB FREE
013430          JSR          SVCBADR        ; SAVE CURRENT ADDRESS OF VCB.
013431          LDA          VCB+VCBSTAT,X   ; TEST FOR ANY OPEN FILES.
013432          BPL          VLOOK3         ; LOG THE VOLUME IN JUST TO BE SURE
013433          LDA          SCRTCH+1       ; SET UP
013434          STA          DEVNUM        ; DEVICE NUMBER ARGUMENT
013435          TXA          ; SAVE PTR TO VCB
013436          PHA          ; ON STACK
013437          JSR          VERFYVOL       ; COMPARES VCBPTR TO DEVNUM CONTENTS
013438          BCC          VNOSWIT
013439          CMP          #VNFERR        ; SEE IF NOTHING IN DRIVE
013440          BEQ          VLOOK7         ; BRANCH IF NOTHING IN DRIVE
013441          JSR          TSTSOS         ; IS THE VOLUME AN UNRECOGNIZED SOS OR (UCSD OR DOS)?
013442          BCS          KNOTSOS       ; DEFINITELY NOT SOS FORMAT
013443          LDX          #0            ; START VCB SCAN AT BEGINNING
013444          JSR          SNSWIT1        ; FIND A FREE VCB AND LOG IN THE NEW GUY
013445          BCS          VNOSWIT1      ; CAN'T LOG IN NEW GUY--KEEP OLD
013446          PLA
013447          LDX          VCBPTR        ; PASS BACK X AS NEW VCB
013448          RTS
013449 *
013450 NFOPEN    DS          1            ; TEMP VCB PTR FOR VCB W/ NO FILES OPEN
013451 *
013452 VNOSWIT   CLC          ; RETURN IT TO USER
013453          PLA          ; REMEMBER OLD VCB PTR
013454          TAX          ; AND PASS BACK TO USER
013455          RTS
013456 ; RETURN TO CALLER X=POINTER TO VCB.
013457 *
013458 VOLERR1    SEC          ; RETURN SOME VOLUME ERROR
013459          RTS
013460 VNOSWIT1  CMP          #DUPVOL
013461          BNE          VLOOK7         ; REPORT OTHER ERROR FROM LOGGING IN NEW VOL AS VNF
013462          TAX
013463          PLA          ; MAKE STACK CORRECT
013464          TXA          ; RESTORE ERROR CODE
013465          SEC
013466          RTS          ; IF DUPLICATE VOLUME ERROR, RETURN FACT TO USER
013467 KNOTSOS   PLA          ; MAKE STACK CORRECT
013468          LDA          #NOTSOS        ; FOR THE PASCAL FOLK

```

```

013469          RTS          ; NOTSOS MEANS UCSD OR DOS OR BAD SOS VOLUME
013470      *
013471  VLOOK7          PLA          ; THROW AWAY OLD VCB PTR
013472          JMP          NOVOLM  ; AND REPORT VOLUME NOT FOUND
013473      *
013474  VLOOK1          JSR          SVCBADR  ; SAVE ADDRESS OF FREE VCB.
013475  VLOOK2          TXA          ; BUMP TO NEXT VOLUME ENTRY.
013476          CLC
013477          ADC          #VCBSIZE
013478          BCC          VOLOOK  ; BRANCH IF MORE TO CHECK.
013479          LDX          VCBPTR+1  ; FREE VCB YET FOUND?
013480          BNE          VLOOK3  ; BRANCH IF YES
013481          LDX          NFOPEN  ; SAVE POSSIBLE FREE VCB
013482          JSR          SVCBADR  ; AND SAVE PTR PERMANENTLY
013483  VLOOK3          LDA          VCBPTR+1  ; WAS A FREE VCB FOUND?
013484          BEQ          NOVOLM  ; BRANCH IF VOLUME CAN'T BE LOGGED IN.
013485          LDA          SCRTCH+1  ; GET DEVICE NUMBER
013486          STA          DEVNUM  ; SAVE DEVICE NUMBER.
013487          LDA          #1      ; FAKE OUT 'LOKVOL'
013488          STA          SCRTCH  ; TO THINK TO LOOK ONLY ONCE.
013489          STA          TOTDEVS
013490          LDA          #<VCB
013491          STA          VCBPTR+1
013492          STA          PATHNMH  ; (TO MAKE HARMLESS)
013493          LDA          #0
013494          STA          SISTER+PATHNMH
013495          LDX          VCBPTR
013496          STX          PATHNML
013497          STA          VCB,X    ; FORCE CURRENT VOLUME OFF LINE, THEN LOG WHATS THERE.
013498          JSR          FREEVCB  ; GO READ ROOT DIRECTORY.
013499          BCS          RTVOLNAM ; RETURN ANY ERRORS
013500          LDX          VCBPTR  ; MAKE SURE VOLUME WAS LOGGED IN
013501          LDA          VCB,X
013502          BEQ          NOVOLM  ; RETURN ERROR
013503          RTS          ; ELSE RETURN NORMALLY
013504  NOVOLM          LDA          #VNFERR  ; TELL USER 'NO VOLUME'
013505          SEC
013506  RTVOLNAM          TAX          ; SAVE REAL ERROR WHILE DUPLICATE IS CHECKED
013507          LDA          DUPLFLAG
013508          BEQ          RTV1     ; BRANCH IF NOT DUPLICATE
013509          LDX          #DUPVOL
013510  RTV1            TXA          ; RECALL ERROR
013511          RTS
013512
013513          CHN          VOLUME,4,1
013514
013515 *****
013516 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: PATH
013517 *****
013518

```

013519  
013520

```

013521 =====
013522 DOCUMENT :SOS1.3.3of5.THREE:SOS.PRINT.TEXT
013523 =====
013524
013525 *****
013526 * APPLE /// SOS 1.3 SOURCE CODE FILE: PRINT
013527 *****
013528 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
013529
013530          SBTL          'SOS 1.1 BLOCK FILE MANAGER' L
013531 * 01-FEB-82
013532          REL
013533          IBUFSIZ      1
013534          SBUFSIZ      40
013535          INCLUDE      SOSORG,6,1,254
013536          ORG          ORGBFM          ; BITMAPS $B800-$BBFF
013537 ZZORG          EQU          *
013538          REP          60
013539 *          (C) COPYRIGHT 1981 BY APPLE COMPUTER INC.
013540 *          ALL RIGHTS RESERVED
013541          REP          60
013542          MSB          OFF
013543          LST          VSYM
013544          CHN          EQUATES,4,1
013545          CHN          ALLOC
013546          INCLUDE      POSN/OPEN
013547          INCLUDE      READ/WRITE,2,,4
013548          INCLUDE      CLOSE/EOF,2,,4
013549          INCLUDE      DESTROY,2,,4
013550          INCLUDE      SWAPOUT/IN,2,,4
013551
013552 *****
013553 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: PRINT
013554 *****
013555
013556

```

```

013557 =====
013558 DOCUMENT :SOS1.3.3of5.THREE:SOS.UMGR.TEXT
013559 =====
013560
013561 *****
013562 * APPLE /// SOS 1.3 SOURCE CODE FILE: UMGR.SRC
013563 *****
013564 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
013565
013566             SBTL             "SOS 1.1 UTILITY MANAGER"
013567             REL
013568             INCLUDE         SOSORG,6,1,254
013569             ORG             ORGUMGR
013570 ZZORG             EQU             *
013571             MSB             OFF
013572             REP             60
013573 *             COPYRIGHT (C) APPLE COMPUTER INC. 1980
013574 *             ALL RIGHTS RESERVED
013575             REP             60
013576 * UTILITY MANAGER
013577 *
013578 * THIS MODULE HANDLES THE FOLLOWING SOS CALLS:
013579 *     SET.FENCE,     GET.FENCE
013580 *     SET.TIME,     GET.TIME
013581 *     JOYSTICK,     COLDSTRT
013582 *
013583 * IN ADDITION, IT CONTAINS THE ROUTINE DATETIME WHICH
013584 * PROVIDES THE DATE AND TIME FOR THE BLOCK FILE MANAGER.
013585 *
013586             REP             60
013587 *
013588             ENTRY          UMGR
013589             ENTRY          DATETIME
013590             ENTRY          BCDBIN
013591             ENTRY          COLDSTRT
013592 *
013593             ENTRY          PCLOCK
013594 *
013595             EXTRN          SYSBANK
013596             EXTRN          CEVPRI
013597             EXTRN          SYSERR
013598             EXTRN          BADSCNUM
013599             EXTRN          BADJMODE
013600             EXTRN          XNORESRC
013601             EXTRN          ALLOCSIR
013602             EXTRN          DEALCSIR
013603 *
013604 U.TPARMX             EQU             $C0
013605 U.REQCODE           EQU             U.TPARMX

```

```

013606 PRIORITY      EQU      U.TPARMX+1
013607 J.MODE        EQU      U.TPARMX+1
013608 J.VALUE       EQU      U.TPARMX+2
013609 TIME          EQU      U.TPARMX+1
013610 MEMORY        EQU      U.TPARMX+1
013611 *
013612 BITON2        EQU      $04
013613 BITON5        EQU      $20
013614 BITON6        EQU      $40
013615 BITON7        EQU      $80
013616 BITOFF5       EQU      $DF
013617 *
013618 Z.REG          EQU      $FFD0
013619 E.REG          EQU      $FFDF
013620 B.REG          EQU      $FFEF
013621                PAGE
013622                REP      35
013623 *
013624 * UTILITY SWITCH
013625 *
013626                REP      35
013627 *
013628 *
013629 UMGR           EQU      *
013630                LDA      E.REG      ;SELECT $C000 I/O SPACE
013631                ORA      #BITON6
013632                STA      E.REG
013633 *
013634                LDA      U.REQCODE
013635                CMP      #USWCNT
013636                BCS      UMGRERR
013637                ASL      A
013638                TAX
013639                LDA      USWTBL+1,X
013640                PHA
013641                LDA      USWTBL,X
013642                PHA
013643                RTS
013644 *
013645 UMGRERR           LDA      #>BADSCNUM
013646                JSR      SYSERR
013647 *
013648 * UTILITY SWITCH TABLE
013649 *
013650 USWTBL            EQU      *
013651                DW      SET.FENCE-1
013652                DW      GET.FENCE-1
013653                DW      SET.TIME-1
013654                DW      GET.TIME-1
013655                DW      JOYSTICK-1

```



```

013656          DW          COLDSTRT-1
013657 USWCNT          EQU          *-USWTBL/2
013658          PAGE
013659          REP          60
013660 *
013661 * SET.FENCE(IN.PRIORITY) SYSTEM CALL
013662 *
013663 * GET.FENCE(OUT.PRIORITY) SYSTEM CALL
013664 *
013665 * THESE TWO CALLS ALLOW THE CALLER TO EITHER RETRIEVE OR SET
013666 * THE CURRENT SYSTEM EVENT PRIORITY THRESHOLD.  BY RAISING
013667 * THE FENCE, A USER MAY INHIBIT THE EXECUTION OF EVENTS WHOSE
013668 * PRIORITY IS EQUAL TO OR LESS THAN THE VALUE OF THE SYSTEM
013669 * FENCE.
013670 *
013671          REP          60
013672 *
013673 *
013674 SET.FENCE          EQU          *
013675          LDA          PRIORITY
013676          STA          CEVPRI
013677          RTS                          ; NORMAL EXIT
013678 *
013679 *
013680 GET.FENCE          EQU          *
013681          LDA          CEVPRI
013682          LDY          #0
013683          STA          (PRIORITY),Y
013684          RTS                          ; NORMAL EXIT
013685          PAGE
013686          REP          60
013687 *
013688 * SET.TIME(IN.TIME)
013689 * GET.TIME(OUT.TIME)
013690 *
013691 * THESE SYSTEM CALLS ALLOW THE USER TO SET AND READ THE
013692 * SYSTEM'S CLOCK.  THE TIME IS EXPRESSED AS AN EIGHTEEN
013693 * DIGIT ASCII STRING IN THE FORM "YYYYMMDDWHHMMSSMMM" .
013694 *
013695 *      YYYY  YEAR      [1900-1999]
013696 *      MM   MONTH     [01-12]
013697 *      DD   DAY       [01-31]
013698 *      W    WEEKDAY    [1-7]  1 => SUNDAY
013699 *      HH   HOUR      [00-23]
013700 *      MM   MINUTE    [00-59]
013701 *      SS   SECOND    [00-59]
013702 *      MMM  MILLISECOND [000-999]
013703 *
013704 * THE CLOCK CHIP AUTOMATICALLY MAINTAINS THE TIME AND
013705 * DATE FROM MILLISECONDS TO MONTHS.  IT DOES NOT MAINTAIN

```

```

013706 * THE YEAR, HOWEVER, NOR DOES IT RECOGNIZE 29 FEBRUARY
013707 * IN LEAP YEARS. THE SOFTWARE SETS THE DAY AND MONTH
013708 * LATCHES TO THE DON'T CARE STATE AND USES THE REMAINING
013709 * EIGHT BITS TO HOLD A TWO DIGIT BCD YEAR. THE CLOCK
013710 * MUST BE RESET AT THE BEGINNING OF EACH YEAR AND ON
013711 * 29 FEBRUARY IN LEAP YEARS.
013712 *
013713 * SET.TIME ASSUMES THAT THE DATE IS VALID AND CORRECT.
013714 * THE CENTURY IS IGNORED AND MILLISECONDS ARE ALWAYS SET
013715 * TO ZERO. GET.TIME ALWAYS SETS THE CENTURY TO 19.
013716 *
013717 *          REP          60
013718 *
013719 *
013720 * TEMPORARY ZERO PAGE
013721 *
013722 PCLK          EQU          $D0          ;POINTER TO SAVED PCLOCK
013723 WKDAY        EQU          $D2
013724 CKSUM        EQU          $D3
013725 CLKTEMP     EQU          $18D4        ;THROUGH $18DD - ABSOLUTE
013726 *
013727 * CLOCK LOCAL DATA
013728 *
013729 PCLOCK       DS          $0A          ;PSEUDO CLOCK REGISTERS
013730 RETRY        DS          $01
013731 *
013732 * CLOCK HARDWARE ADDRESSES
013733 *
013734 CLOCK        EQU          $C070
013735 CSEC         EQU          $02
013736 CMIN         EQU          $03
013737 CMON        EQU          $07
013738 LDAY        EQU          $0E
013739 CRESET      EQU          $12
013740 STATUS      EQU          $14
013741 *
013742 WKMON        DFB          8,11,11,7,9,12
013743             DFB          7,10,13,8,11,13
013744 *
013745 *
013746 SET.TIME     EQU          *
013747             LDX          #$00
013748             LDY          #$12
013749             LDA          #'0'
013750             BNE          STIM011
013751 *
013752 STIM010      INX
013753             LDA          (TIME),Y      ;CONVERT TIME FROM
013754 STIM011      AND          #$0F        ; ASCII TO BCD AND
013755             STA          PCLOCK,X     ; TRANSFER TO PCLOCK

```

```

013756      DEY
013757      CPY      #$07
013758      BEQ      STIM010
013759      LDA      (TIME),Y
013760      ASL      A
013761      ASL      A
013762      ASL      A
013763      ASL      A
013764      ORA      PCLOCK,X
013765      STA      PCLOCK,X
013766      DEY
013767      BPL      STIM010
013768      *
013769      LDA      PCLOCK+7      ;CALCULATE WEEKDAY
013770      JSR      BCDBIN
013771      TAX
013772      LDA      PCLOCK+8
013773      JSR      BCDBIN
013774      TAY
013775      LSR      A
013776      LSR      A
013777      STA      WKDAY
013778      TYA
013779      AND      #$03
013780      BNE      STIM015
013781      CPX      #3
013782      BCS      STIM015      ; <SRS 82.162>
013783      DEY
013784      STIM015      CLC
013785      TYA
013786      ADC      WKDAY
013787      ADC      WKMON-1,X
013788      STA      WKDAY
013789      LDA      PCLOCK+6
013790      JSR      BCDBIN
013791      CLC
013792      ADC      WKDAY
013793      SEC
013794      STIM016      SBC      #7
013795      CMP      #8
013796      BCS      STIM016
013797      STA      PCLOCK+5
013798      *
013799      LDA      #$D0
013800      STA      PCLK      ;POINT (PCLK) TO 8F:FFD0
013801      LDA      #$FF
013802      STA      PCLK+1
013803      LDA      #$8F
013804      STA      $1401+PCLK
013805      LDA      #$A5

```

```

013806          STA      CKSUM          ;INITIALIZE CHECKSUM
013807          LDY      #$00
013808  *
013809  STIM020    LDA      PCLOCK,Y      ;SAVE PCLOCK
013810          STA      (PCLK),Y        ; BEHIND 6522
013811          EOR      CKSUM
013812          STA      CKSUM
013813          INY
013814          CPY      #$0A
013815          BCC      STIM020
013816          STA      (PCLK),Y        ;SAVE CHECKSUM
013817  *
013818          LDA      Z.REG
013819          PHA                      ;SAVE ZERO PAGE
013820          LDA      E.REG
013821          PHA                      ;SAVE ENVIRONMENT
013822          ORA      #BITON7         ; AND SET 1 MHZ
013823          STA      E.REG
013824  *
013825          LDY      #STATUS
013826          STY      Z.REG
013827          LDA      CLOCK          ;DOES CLOCK EXIST?
013828          BMI      STIM050         ; NO
013829  *
013830          LDX      #CRESET
013831          STX      Z.REG
013832          LDA      #$FF           ;RESET ALL COUNTERS
013833          STA      CLOCK
013834          STA      CLOCK
013835  *
013836          LDX      #CSEC-1
013837  STIM030    INX
013838          PHP
013839          SEI                      ;DISABLE INTERRUPTS
013840  STIM040    STX      Z.REG
013841          LDA      CLOCK          ;(DUMMY READ FOR STATUS)
013842          LDA      PCLOCK,X
013843          STA      CLOCK          ;SET CLOCK COUNTER
013844          LDA      CLOCK          ;(DUMMY READ FOR STATUS)
013845          STY      Z.REG
013846          LDA      CLOCK          ;CHECK STATUS BIT
013847          BNE      STIM040
013848          PLP                      ;RESTORE INTERRUPTS
013849          CPX      #CMON
013850          BCC      STIM030
013851  *
013852          LDX      #LDAY
013853          STX      Z.REG
013854          LDA      PCLOCK+8
013855          ORA      #5CC           ;STUFF YEAR INTO DAY

```

```

013856      STA      CLOCK      ; AND MONTH LATCHES
013857      INC      Z.REG
013858      LDA      PCLOCK+8
013859      LSR      A
013860      LSR      A
013861      ORA      #$CC
013862      STA      CLOCK
013863      *
013864      STIM050      PLA
013865      STA      E.REG      ;RESTORE ENVIRONMENT
013866      PLA
013867      STA      Z.REG      ; AND ZERO PAGE
013868      RTS
013869      PAGE
013870      GET.TIME      EQU      *
013871      LDA      Z.REG      ;SAVE ZERO PAGE
013872      PHA
013873      LDA      E.REG      ;SAVE ENVIRONMENT
013874      PHA
013875      ORA      #BITON7
013876      STA      E.REG      ;SET 1 MHZ
013877      *
013878      LDY      #STATUS
013879      STY      Z.REG
013880      LDA      CLOCK      ;DOES CLOCK EXIST?
013881      BMI      GTIM050      ; NO
013882      *
013883      LDA      #$10      ;ALLOW $10 RETRYs
013884      STA      RETRY
013885      GTIM010      LDX      #CMON+1
013886      PHP
013887      SEI      ;DISABLE INTERRUPTS
013888      *
013889      GTIM020      DEX
013890      BMI      GTIM030      ;ALL DONE
013891      STX      Z.REG
013892      LDA      CLOCK      ;COPY CLOCK COUNTERS
013893      STA      CLKTEMP,X      ; TO TEMP REGISTERS
013894      STY      Z.REG
013895      LDA      CLOCK      ;CHECK STATUS BIT
013896      BEQ      GTIM020
013897      *
013898      PLP      ;CLOCK READ ERROR
013899      DEC      RETRY
013900      BPL      GTIM010      ;TRY AGAIN
013901      BMI      GTIM050
013902      *
013903      GTIM030      PLP      ;RESTORE INTERRUPTS
013904      LDX      #LDAY+1
013905      STX      Z.REG

```

```

013906      LDA      CLOCK          ;READ YEAR FROM DAY
013907      SEC
013908      ROL      A              ; AND MONTH LATCHES
013909      ROL      A
013910      DEC     Z.REG
013911      AND     CLOCK
013912      STA     CLKTEMP+8
013913      *
013914      LDX     #$09
013915 GTIM040  LDA     CLKTEMP,X    ;COPY CLOCK DATA
013916      STA     PCLOCK,X        ; TO PSEUDO CLOCK
013917      DEX
013918      BPL     GTIM040
013919      *
013920 GTIM050  LDA     #$19
013921      STA     PCLOCK+9
013922      *
013923      PLA
013924      STA     E.REG          ;RESTORE ENVIRONMENT
013925      PLA
013926      STA     Z.REG          ; AND ZERO PAGE
013927      *
013928      LDY     #$11
013929      LDX     #$00
013930 GTIM060  LDA     PCLOCK,X    ;GET MOST SIGNIFICANT
013931      LSR     A              ; BCD DIGIT
013932      LSR     A
013933      LSR     A
013934      LSR     A
013935      ORA     #$30          ;CONVERT TO ASCII
013936      STA     (TIME),Y
013937      INX
013938      DEY
013939      BMI     GTIM080
013940 GTIM070  LDA     PCLOCK,X    ;GET LEAST SIGNIFICANT
013941      AND     #$0F          ; BCD DIGIT
013942      ORA     #$30          ;CONVERT TO ASCII
013943      STA     (TIME),Y
013944      DEY
013945      CPY     #$07
013946      BNE     GTIM060
013947      INX
013948      BNE     GTIM070
013949 GTIM080  RTS
013950      PAGE
013951      REP     60
013952      *
013953      * SUBROUTINE DATETIME
013954      *
013955      * THIS SUBROUTINE READS THE CLOCK AND WRITES A DATE/TIME

```

```

013956 * STAMP TO A FOUR BYTE BUFFER ON THE CALLER'S ZERO PAGE;
013957 * THE DATA FORMAT IS SHOWN BELOW.  ON ENTRY, X MUST POINT
013958 * TO THE BUFFER.  ON EXIT, ALL REGISTERS ARE CLOBBERED.
013959 * IF AN ERROR OCCURS, CARRY IS SET AND THE BUFFER IS
013960 * SET TO ZERO; OTHERWISE, CARRY IS CLEARED.
013961 *
013962 *   BITS: 7 6 5 4 3 2 1 0
013963 *   X+0  M M M D D D D D
013964 *   X+1  Y Y Y Y Y Y Y M
013965 *   X+2  -  MINUTE  -
013966 *   X+3  -  -  HOUR  -  -
013967 *
013968 *           REP           60
013969 *
013970 * TEMPORARY STORAGE
013971 *
013972 OFFSET           DFB           0
013973 ERRCNT           DFB           0
013974 CLKREGS         DS            5
013975 MIN             EQU           CLKREGS+0
013976 HOUR           EQU           CLKREGS+1
013977 DAY           EQU           CLKREGS+3
013978 MON           EQU           CLKREGS+4
013979 YEAR         EQU           CLKREGS+2
013980 *
013981 *
013982 DATETIME        EQU           *
013983                STX           OFFSET
013984                LDA           Z.REG
013985                PHA                ;SAVE ZERO PAGE
013986                LDA           E.REG
013987                PHA                ; AND ENVIRONMENT
013988                ORA           #BITON7+BITON6 ;SET 1 MHZ AND
013989                STA           E.REG ; ENABLE I/O SPACE
013990 *
013991                LDY           #STATUS
013992                STY           Z.REG
013993                LDA           CLOCK ;DOES CLOCK EXIST?
013994                BMI           DT030 ; NO
013995 *
013996                LDA           #8
013997                STA           ERRCNT ;ALLOW 8 RETRYS
013998 DT010          LDX           #CMON+1
013999                PHP
014000                SEI                ;DISABLE INTERRUPTS
014001 *
014002 DT020          DEX
014003                CPX           #CMIN
014004                BCC           DT050
014005                STX           Z.REG

```

```

014006      LDA      CLOCK      ;READ THE CLOCK
014007      STA      CLKREGS-CMIN,X
014008      STY      Z.REG
014009      LDA      CLOCK      ;CHECK STATUS
014010      BEQ      DT020
014011      *
014012      PLP
014013      DEC      ERRCNT
014014      BPL      DT010
014015      PLA
014016      STA      E.REG      ;RESTORE ENVIRONMENT
014017      PLA
014018      STA      Z.REG      ; AND ZERO PAGE
014019      LDX      #CMON-CMIN
014020      LDA      PCLOCK+CMIN,X
014021      STA      CLKREGS,X
014022      DEX
014023      BPL      DT040
014024      LDX      PCLOCK+8
014025      JMP      DT060
014026      *
014027      DT050      PLP      ;READ YEAR FROM LATCHES
014028      LDA      #LDAY+1
014029      STA      Z.REG
014030      LDA      CLOCK
014031      SEC
014032      ROL      A
014033      ROL      A
014034      DEC      Z.REG
014035      AND      CLOCK
014036      TAX
014037      *
014038      PLA
014039      STA      E.REG      ;RESTORE ENVIRONMENT
014040      PLA
014041      STA      Z.REG      ; AND ZERO PAGE
014042      *
014043      DT060      TXA
014044      JSR      BCDBIN      ;CONVERT YEAR TO BINARY
014045      STA      YEAR
014046      LDA      MON      ;CONVERT MONTH AND DAY
014047      JSR      BCDBIN      ; TO BINARY THEN
014048      ASL      A      ; COMBINE WITH YEAR
014049      ASL      A      ; TO FORM DATE STAMP
014050      ASL      A
014051      ASL      A
014052      ASL      A
014053      STA      MON
014054      ROL      YEAR
014055      LDA      DAY

```



```

014056          JSR      BCDBIN
014057          ORA      MON
014058          LDX      OFFSET
014059          STA      0,X
014060          LDA      YEAR
014061          STA      1,X
014062          LDA      MIN          ;CONVERT MINUTE
014063          JSR      BCDBIN
014064          STA      2,X
014065          LDA      HOUR        ;CONVERT HOUR
014066          JSR      BCDBIN
014067          STA      3,X
014068          CLC
014069          RTS
014070          PAGE
014071          REP      60
014072 *
014073 * SUBROUTINE BCDBIN
014074 *
014075 * THIS SUBROUTINE CONVERTS A BYTE FROM BCD TO BINARY.
014076 * THE BYTE IS PASSED AND RETURNED IN A.  THERE IS NO
014077 * ERROR CHECKING.  Y IS DESTROYED AND X IS UNCHANGED.
014078 *
014079          REP      60
014080 *
014081 BCDBIN      EQU      *
014082          PHA
014083          LSR      A          ;ISOLATE TENS DIGIT FOR
014084          LSR      A          ; INDEXING THE TABLE
014085          LSR      A
014086          LSR      A
014087          TAY
014088          PLA
014089          AND      #$0F          ;GET UNITS
014090          CLC
014091          ADC      TENS,Y        ;ADD IN TENS
014092          RTS
014093 *
014094 TENS          DFB      00,10,20,30,40,50,60,70,80,90
014095          PAGE
014096          REP      60
014097 *
014098 * SOS CALL $64 -- JOYSTICK INPUT
014099 * JOYSTICK(IN.J.MODE; OUT.J.VALUE)
014100 *
014101          REP      60
014102 *
014103 *
014104 AD.INPUT      EQU      $D0
014105 AD.TEMP      EQU      $D1

```

```

014106 *
014107 PA.SW0      EQU      $C061      ;PORT A, SWITCH 0
014108 PA.SW1      EQU      $C063      ;PORT A, SWITCH 1
014109 PB.SW0      EQU      $C062      ;PORT B, SWITCH 0
014110 PB.SW1      EQU      $C060      ;PORT B, SWITCH 1
014111 *
014112 AD.SEL0      EQU      $C058      ;A/D SELECT CONTROLS
014113 AD.SEL1      EQU      $C05E
014114 AD.SEL2      EQU      $C05A
014115 AD.CHRG      EQU      $C05C      ;A/D RAMP CHARGE /
014116 AD.STRT      EQU      $C05D      ; START TIMEOUT
014117 AD.FLAG      EQU      $C066      ;A/D TIMEOUT FLAG
014118 *
014119 TCHARGE      EQU      500        ;CHARGE TIME FOR A/D
014120 TOFFSET      EQU      360        ;OFFSET TIME TO A/D WINDOW
014121 *
014122 ANALOG        EQU      $F4A8      ;ROM ENTRY FOR ANALOG INPUT
014123 ANLOG1        EQU      $F4AB      ; INTERRUPT REENTRY
014124 D.T2          EQU      $FFD8      ;TIMER
014125 D.ACR         EQU      $FFDB      ;AUXILIARY CONTROL REGISTER
014126 D.IFR         EQU      $FFDD      ;INTERRUPT FLAG REGISTER
014127 *
014128 ENSEL         EQU      $C0DC
014129 ENSIO         EQU      $CODE
014130 *
014131 *
014132 JOYSTICK       EQU      *
014133             LDA      J.MODE        ;VALIDATE J.MODE
014134             CMP      #$08
014135             BCC      JS010
014136             LDA      #>BADJMODE
014137 JS.ERR         JSR      SYSERR
014138 *
014139 JS010           JSR      AD.SETUP    ;SET UP RESOURCES
014140             BCS      JS.ERR
014141             LDA      J.MODE        ;READ PORT B OR PORT A?
014142             AND      #BITON2
014143             BNE      JS020
014144             LDA      PB.SW0        ;PORT B
014145             LDX      PB.SW1
014146             LDY      #$01
014147             BNE      JS030
014148 JS020           LDA      PA.SW0        ;PORT A
014149             LDX      PA.SW1
014150             LDY      #$03
014151 JS030           STY      AD.INPUT    ;SAVE INPUT SELECT
014152             AND      #BITON7
014153             BEQ      JS040
014154             LDA      #$FF
014155 JS040           LDY      #$00

```

```

014156          STA      (J.VALUE),Y          ;RETURN SWITCH 0
014157          TXA
014158          AND      #BITON7
014159          BEQ      JS050
014160          LDA      #$FF
014161 JS050      INY
014162          STA      (J.VALUE),Y          ;RETURN SWITCH 1
014163          *
014164          LSR      J.MODE
014165          BCC      JS060
014166          LDA      AD.INPUT
014167          JSR      AD.READ          ;READ A/D
014168          LDY      #$02
014169          STA      (J.VALUE),Y          ;RETURN X AXIS
014170 JS060      INC      AD.INPUT
014171          LSR      J.MODE
014172          BCC      JS070
014173          LDA      AD.INPUT
014174          JSR      AD.READ          ;READ A/D
014175          LDY      #$03
014176          STA      (J.VALUE),Y          ;RETURN Y AXIS
014177          *
014178 JS070      JSR      AD.CLNUP          ;CLEAN UP
014179          RTS          ; AND EXIT
014180          PAGE
014181          REP      60
014182          *
014183          * SUBROUTINE AD.SETUP
014184          * THIS SUBROUTINE SETS UP THE ENVIRONMENT AND RESOURCES
014185          * FOR READING THE JOYSTICKS. IF AN ERROR OCCURS, CARRY
014186          * IS SET AND AN ERROR NUMBER IS RETURNED IN A.
014187          * OTHERWISE, CARRY IS CLEARED.
014188          *
014189          REP      60
014190 AD.SETUP    EQU      *
014191          LDA      #JOYSIRSIZ
014192          LDX      #>JOYSIRTBL
014193          LDY      #<JOYSIRTBL
014194          JSR      ALLOCSIR          ;ALLOCATE RESOURCES
014195          BCC      ADS010
014196          LDA      #>XNORESRC
014197          RTS
014198 ADS010     LDA      E.REG
014199          AND      #$7F          ;SET 2 MHZ,
014200          ORA      #$43          ; ENABLE ROM, & I/O SPACE
014201          STA      E.REG
014202          PHP
014203          SEI
014204          LDA      D.ACR
014205          AND      #BITOFF5          ;SET UP TIMER

```

```

014206          STA      D.ACR
014207          PLP
014208          BIT      ENSEL          ;DISABLE ENSEL
014209          BIT      ENSIO          ;SET ENSIO FOR INPUT
014210          RTS
014211 *
014212 JOYSIRTBL  EQU      *
014213          DFB      $0C,0,0,0,0      ;ENSIO
014214          DFB      $0D,0,0,0,0      ;ENSEL
014215          DFB      $0E,0,0,0,0      ;6522 D.T2
014216 JOYSIRSIZ  EQU      *-JOYSIRTBL
014217          REP      60
014218 *
014219 * SUBROUTINE AD.CLNUP
014220 * THIS SUBROUTINE RESTORES THE ENVIRONMENT AND RELEASES
014221 * THE RESOURCES AFTER READING THE JOYSTICKS.
014222 *
014223          REP      60
014224 AD.CLNUP     EQU      *
014225          LDA      E.REG
014226          AND      #$3C              ;RESTORE RAM AT $C000 & $F000
014227          STA      E.REG
014228          LDA      #JOYSIRSIZ
014229          LDX      #>JOYSIRTBL
014230          LDY      #<JOYSIRTBL
014231          JSR      DEALCSIR          ;DEALLOCATE RESOURCES
014232          RTS
014233          PAGE
014234          REP      60
014235 *
014236 * SUBROUTINE AD.READ
014237 * THIS SUBROUTINE READS A SPECIFIED A/D INPUT AND RETURNS
014238 * AN 8 BIT RESULT. IT ASSUMES THAT THE A/D RESOURCES HAVE
014239 * BEEN ALLOCATED, THE I/O SPACE AND $F000 ROM HAVE BEEN
014240 * SELECTED, AND THE SYSTEM IS RUNNING IN 2 MHZ MODE.
014241 *
014242 * PARAMETERS:
014243 *   A:  A/D INPUT PORT (0-7)
014244 *
014245 * RETURN VALUE:
014246 *   A:  RESULT (0 - 255)
014247 *   X, Y: UNDEFINED
014248 *
014249          REP      60
014250 *
014251 AD.READ     EQU      *
014252          LSR      A                  ;SELECT THE APPROPRIATE
014253          BIT      AD.SEL0            ; A/D INPUT
014254          BCC      ADR010
014255          BIT      AD.SEL0+1

```

```

014256 ADR010      LSR      A
014257           BIT      AD.SEL1
014258           BCC      ADR020
014259           BIT      AD.SEL1+1
014260 ADR020      LSR      A
014261           BIT      AD.SEL2
014262           BCC      ADR030
014263           BIT      AD.SEL2+1
014264 ADR030      PHP
014265 *
014266 ADR040      CLI
014267           BIT      AD.CHRG      ;CHARGE A/D CAPACITOR
014268           LDA      #>TCHARGE
014269           STA      D.T2
014270           LDA      #<TCHARGE
014271           STA      D.T2+1
014272           LDA      #BITON5
014273 ADR050      BIT      D.IFR
014274           BEQ      ADR050
014275 *
014276           SEI
014277           SEC
014278           LDA      #>TOFFSET
014279           STA      D.T2      ;SET UP TIMER
014280           LDA      #<TOFFSET
014281           BIT      AD.STRT      ;START A/D TIMEOUT
014282           JSR      ANALOG      ;MEASURE CONVERSION TIME
014283           BCC      ADR070
014284 *
014285 ADR060      CLI      ;PROCESS AN INTERRUPT
014286           SEI
014287           BIT      AD.FLAG      ;STILL TIMING?
014288           BPL      ADR040      ; NO -- START OVER
014289           JSR      ANLOG1      ; YES -- CONTINUE
014290           BCS      ADR060
014291 *
014292 ADR070      PLP
014293           EOR      #$FF      ;NORMALIZE RESULT
014294           BMI      ADR080      ;RESULT < 0
014295           STA      AD.TEMP
014296           TYA
014297           EOR      #$FF
014298           LSR      AD.TEMP
014299           ROR      A
014300           LSR      AD.TEMP
014301           ROR      A
014302           LSR      AD.TEMP
014303           BNE      ADR090      ;RESULT > 255
014304           ROR      A
014305           ADC      #0

```

```

014306          RTS
014307  ADR080   LDA      #0
014308          RTS
014309  ADR090   LDA      #$FF
014310          RTS
014311          PAGE
014312          REP      60
014313  *
014314  *  SYSTEM COLD START
014315  *
014316  *  THIS ROUTINE IS CALLED TO TELL THE USER TO REBOOT THE
014317  *  SYSTEM.  IT CLEARS THE SCREEN, DISPLAYS A MESSAGE,
014318  *  OVERWRITES BANKED MEMORY, AND HANGS UNTIL THE USER
014319  *  PERFORMS A HARD RESET.
014320  *
014321          REP      60
014322  *
014323  *
014324  COLDSTRT  EQU      *
014325          SEI
014326          LDA      #$40          ;SHUT DOWN INTERRUPTS
014327          STA      $FFCA        ; AND IGNORE NMI
014328          LDA      #$67
014329          STA      E.REG        ;DISABLE RESET
014330          LDA      #$00
014331          STA      Z.REG        ;USE PAGE ZERO
014332  *
014333          LDX      SYSBANK
014334          LDA      #$BF
014335          LDY      #$00
014336          STY      MEMORY
014337  CS010    STA      MEMORY+1
014338          STX      B.REG
014339          LDA      #$A0
014340  CS020    STA      (MEMORY),Y   ;SET MEMORY TO BLANKS
014341          DEY
014342          BNE      CS020
014343          DEC      MEMORY+1
014344          BNE      CS020
014345          DEX
014346          BPL      CS010
014347  *
014348          LDY      #6
014349  CS030    STA      $C050,Y      ;SELECT 40 COLUMN
014350          DEY                    ; BLACK & WHITE TEXT
014351          BPL      CS030
014352  *
014353          LDY      #BOOTLEN
014354  CS040    LDA      BOOTMSG-1,Y   ;PRINT BOOT MESSAGE
014355          STA      BOOTADR-1,Y

```

```

014356          DEY
014357          BNE          CS040
014358  *
014359          LDA          #$77
014360          STA          E.REG          ;ENABLE RESET
014361          JMP          *              ;HANG UNTIL RESET
014362          PAGE
014363          MSB          ON
014364  BOOTMSG    ASC          "INSERT SYSTEM DISKETTE & REBOOT"
014365  BOOTLEN    EQU          *-BOOTMSG
014366  BOOTADR    EQU          40-BOOTLEN/2+$628
014367          MSB          OFF
014368          LST          ON
014369  ZZEND     EQU          *
014370  ZZLEN     EQU          ZZEND-ZZORG
014371          IFNE        ZZLEN-LENUMGR
014372          FAIL        2,"SOSORG          FILE IS INCORRECT FOR UMBR"
014373          FIN
014374
014375 *****
014376 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: UMGR.SRC
014377 *****
014378
014379

```

```

014380 =====
014381 DOCUMENT :SOS1.3.3of5.THREE:SOS.VOLUME.TEXT
014382 =====
014383
014384 *****
014385 * APPLE /// SOS 1.3 SOURCE CODE FILE: VOLUME
014386 *****
014387 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
014388
014389             PAGE
014390             REP         40
014391 * NAME      : VOLUME
014392 * FUNCTION: RETURN VOLUME INFO
014393 * INPUT    : DEVICE NAME
014394 * OUTPUT   : THE INFO
014395 * VOLATILE: ALL REGS
014396             REP         40
014397 *
014398 VOLUME      EQU         *
014399             LDA         C.DNAMP           ; TRANSFER DEVICE NAME
014400             STA         DVNAMP           ; NAME FOR DMGR
014401             LDA         C.DNAMP+1
014402             STA         DVNAMP+1
014403             LDA         SISTER+C.DNAMP+1 ; AND XTND
014404             STA         SISTER+DVNAMP+1
014405             JSR         GETDNUM          ; GET DEVNUM
014406             BCC         VOL7             ; =>SOME KINDA ERROR
014407             RTS
014408 VOL7        BMI         VOL2             ; =>IT'S GOOD...
014409             LDA         #NOTBLKDEV       ; NOT BLOCKED
014410             JMP         VOLERR           ; =>RETURN THE ERROR
014411 *
014412 * UNCONDITIONALLY READ ROOT DIRECTORY:
014413 *
014414 VOL2        EQU         *
014415             LDA         SCRATCH+1
014416             STA         DEVNUM           ; SETUP DEV NUMBER
014417             LDA         #2               ; BLKNUM=2
014418             LDX         #0
014419             JSR         GETROT0          ; GET IT PLEASE
014420             LDA         #VNFERR          ; ERROR CODE
014421             BCC         VOL8             ; BRANCH IF NO ERROR ON READ
014422             RTS
014423 *
014424 VOL8        LDA         #>VCB           ; SET VCBPTR TO THE
014425             STA         VCBPTR           ; FIRST OF THEM
014426             LDA         #<VCB
014427             STA         VCBPTR+1
014428 *

```



```

014429 * IS THIS VOLUME SOS OR OTHER?
014430 *
014431          JSR      TSTSOS          ; WHICH KIND?
014432          BCC      VLOGGED        ; =>IT'S SOS
014433          JMP      VNOTSOS        ; =>NOT SOS
014434 *
014435 * IS THIS SOS VOLUME LOGGED IN?
014436 *
014437 VLOGGED      EQU      *
014438          JSR      CMPVCB          ; DOES VOLNAME MATCH?
014439          BCC      VFOUND          ; =>YES, WE KNOW ABOUT IT.
014440          JSR      VNXTVCB         ; BUMP TO NEXT
014441          BCC      VLOGGED        ; =>TRY 'EM ALL...
014442          BCS      VNEW           ; =>NOT FOUND, IT'S NEW (BRANCH ALWAYS)
014443 *
014444 *
014445 * IT'S BEEN LOGGED IN BEFORE:
014446 * IS IT SWAPPED IN OR OUT?
014447 *
014448 VFOUND      EQU      *
014449          LDY      #VCBSWAP        ; INDEX TO IT
014450          LDA      (VCBPTR),Y      ; SWAPPED?
014451          BPL      VFOUND1        ; =>IN. RETURN THE INFO
014452 *
014453 * SWAPPED OUT. BEFORE WE SWAP IT
014454 * IN, MAKE SURE IT BELONGS ON
014455 * THIS DEVICE!
014456 *
014457          LDY      #VCBDEV         ; INDEX TO IT
014458          LDA      (VCBPTR),Y      ; GET ITS DEVICE
014459          CMP      DEVNUM          ; CORRECT DEVICE?
014460          BEQ      VSWAPIN         ; =>YES
014461          LDA      #DUPVOL         ; IF FOR ANOTHER DEV,
014462          JMP      VOLERR          ; THEN IT'S AN ERROR!
014463 *
014464 * NOW SWAP-IN THIS VOLUME:
014465 *
014466 VSWAPIN      EQU      *
014467          JSR      SWAPIN          ; SWAP IT IN
014468          JMP      VINFO           ; AND RETURN THE INFO
014469 *
014470 VFOUND1      LDY      #VCBDEV
014471          LDA      (VCBPTR),Y      ; SAME DEVICES?
014472          CMP      DEVNUM
014473          BEQ      VINFO           ; YES; RETURN THE INFORMATION
014474          LDY      #VCBSTAT
014475          LDA      (VCBPTR),Y      ; OPEN FILES?
014476          BPL      VFOUND2        ; BRANCH IF NOT
014477          LDA      #DUPVOL
014478          BNE      VOLERR          ; ELSE REPORT DUPLICATE VOLUME ERROR (BRANCH ALWAYS)

```

```

014479 VFOUND2      LDY      #VCBNML      ; MOVE THE LOGIN TO THIS NEW DEVICE
014480              LDA      #0          ; BY UNLOGGING THE OLD
014481              STA      (VCBPTR),Y   ; AND LOGGING IN THE NEW (DROP INTO VNEW)
014482              REP      40
014483 *
014484 * IT'S A BRAND NEW VOLUME.
014485 * GUESS WE'LL HAVE TO LOG IT IN:
014486 *
014487 VNEW          EQU      *
014488              LDA      DEVNUM      ; PASS A REG TO SWAPOUT
014489              JSR      SWAPOUT     ; SWAP ANY ACTIVE VOL ON THIS DEVICE
014490              BCC      VNEW1      ; BRANCH ON NO ERROR
014491              LDA      #XIOERROR
014492              RTS
014493 VNEW1        LDA      #>VCB      ; FIND AN EMPTY VCB
014494              STA      VCBPTR
014495              LDA      #<VCB
014496              STA      VCBPTR+1
014497 VFREE        LDY      #VCBNML
014498              LDA      (VCBPTR),Y   ; EMPTY VCB?
014499              BEQ      VLOGIN     ; ITS FREE, USE IT
014500              LDY      #VCBDEV
014501              LDA      (VCBPTR),Y   ; OR ONE WITH SAME DEVICE
014502              CMP      DEVNUM
014503              BNE      VFREEX      ; BRANCH IF NO DEVICE MATCH
014504              LDY      #VCBSTAT
014505              LDA      (VCBPTR),Y   ; AND NO OPEN FILES
014506              BPL      VLOGIN     ; BRANCH IF OK TO REUSE THIS VCB
014507              LDA      DEVNUM
014508              JSR      SWAPOUT     ; THEN WE MUST SWAP OUT THIS VOLUME
014509              BCC      VFREEX
014510              LDA      #XIOERROR  ; SWAPOUT PROCEEDED OK
014511              RTS                ; ELSE REPORT ERROR
014512 VFREEX        JSR      VNXTVCB   ; TRY NEXT
014513              BCC      VFREE      ; MORE TO COME
014514 * RAN OUT OF MT'S ... FIND W/O FILES
014515 VNFIL         LDY      #VCBSTAT
014516              LDA      (VCBPTR),Y
014517              BPL      VLOGIN
014518              JSR      VNXTVCB
014519              BCC      VNFIL
014520 * ALL OPEN ... REPORT VCBFULL
014521              LDA      #FCBFULL
014522              BNE      VOLERR
014523 VLOGIN        EQU      *
014524              JSR      LOGVCB      ; AND LOGIN THIS ONE
014525              REP      40
014526 *
014527 * RETURN ALL THE NICE INFO:
014528 *

```

```

014529 VINFO      EQU      *
014530          LDA      #0
014531          LDY      #VCBTFRE      ; FETCH VOLUME FREE BLOCK COUNT
014532          STA      (VCBPTR),Y    ; FORCE RESCAN OF ALL
014533          INY
014534          STA      (VCBPTR),Y    ; TO MAKE SURE VCB INFO CURRENT
014535          STA      REQL           ; FREE BLOCKS
014536          STA      REQH
014537          JSR      TSFRBLK
014538 *
014539          LDX      VCBPTR         ; GET VCB INDEX
014540          LDY      #0
014541 VINFO1      EQU      *
014542          LDA      VCB+VCBTBLK,X  ; MOVE TOTAL
014543          STA      (C.OUTBLK),Y  ; BLOCKS AVAIL
014544          INX
014545          INY
014546          CPY      #4           ; AND FREE ONES TOO
014547          BNE      VINFO1
014548 *
014549          LDY      #0           ; NOW DO VOLNAME
014550          LDA      (VCBPTR),Y
014551          TAY
014552 VINFO2      EQU      *
014553          LDA      (VCBPTR),Y
014554          STA      (C.OUTVOL),Y
014555          DEY
014556          BPL      VINFO2
014557          CLC
014558          BCC      VOLRET        ; =>DONE
014559 *
014560 VOLERR        EQU      *
014561          SEC
014562 VOLRET        EQU      *
014563          RTS
014564          PAGE
014565          REP      40
014566 * THIS ISN'T A SOS VOLUME. MARK
014567 * THE ACTIVE VOL THIS DEVICE
014568 * SO THAT IT GETS CHECKED LATER:
014569 *
014570 VNOTSOS      EQU      *
014571          LDY      #VCBDEV         ; IS VCB FOR THIS
014572          LDA      (VCBPTR),Y     ; DEVICE?
014573          CMP      DEVNUM
014574          BNE      VNS2
014575          LDY      #VCBSTAT        ; INDEX TO IT
014576          LDA      (VCBPTR),Y     ; GET STATUS
014577          BPL      VNS2           ; =>NOT ACTIVE.
014578          ORA      #DSWITCH       ; SET 'SWITCHEROO'

```

```

014579          STA      (VCBPTR),Y          ; PUT IT BACK
014580 *
014581 VNS2      EQU      *
014582          JSR      VNXTVCB              ; GET NEXT VCB
014583          BCC      VNOTSOS              ; =>TRY 'EM ALL.
014584 *
014585          LDA      #NOTSOS                ; GIVE THE ERROR
014586          BNE      VOLERR                ; (BRANCH ALWAYS)
014587          SKP      5
014588 * NAME      : VNXTVCB
014589 * FUNCTION: BUMP VCBPTR TO NEXT VCB
014590 * INPUT      : NOTHING
014591 * OUTPUT     : VCBPTR UPDATED
014592 *           : 'BCC' IF MORE TO GO
014593 *           : 'BCS' IF DONE
014594 * VOLATILE: AC
014595 *
014596 VNXTVCB     EQU      *
014597          LDA      VCBPTR
014598          CLC
014599          ADC      #VCBSIZE                ; BUMP IT
014600          STA      VCBPTR
014601          RTS                                ; CARRY SET IF END OF PAGE
014602
014603          CHN      CREATE,4,1
014604
014605 *****
014606 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: VOLUME
014607 *****
014608
014609

```

```

014610 =====
014611 DOCUMENT :SOS1.3.4of5.FOUR:SOS.CLOSE.EOF.TEXT
014612 =====
014613
014614 *****
014615 * APPLE /// SOS 1.3 SOURCE CODE FILE: CLOSE.EOF
014616 *****
014617 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
014618
014619 PAGE
014620 *
014621 *
014622 CLOSE LDA C.REFNUM ; CLOSE ALL?
014623 BNE CLOSE1 ; NO, JUST ONE OF 'EM
014624 STA CFERR ; CLEAR GLOBAL CLOSE ERROR
014625 JSR GFCBADR ; SET UP POINTER TO FCB
014626 CLOSALL LDA #0 ; BEGIN AT THE BEGINNING.
014627 CLSALL1 STA FCBPTR ; SAVE CURRENT LOW BYTE OF POINTER
014628 LDY #FCBLEVL ; FETCH THE LEVEL AT WHICH
014629 LDA (FCBPTR),Y ; FILE WAS OPENED
014630 CMP LEVEL ; TEST AGAINST CURRENT GLOBAL LEVEL
014631 BCC NXTCLOS ; DONT CLOSE IF FILES LEVEL IS < GLOBAL LEVEL
014632 LDY #FCBREFN ; INDEX TO REFERENCE NUMBER
014633 LDA (FCBPTR),Y ; IS THIS REFERENCE FILE OPEN?
014634 BEQ NXTCLOS ; NO, TRY NEXT.
014635 JSR FLUSH2 ; CLEAN IT OUT...
014636 BCS CLOSEERR ; RETURN FLUSH ERRORS
014637 JSR CLOSE2 ; UPDATE FCB & VCB
014638 LDY C.REFNUM
014639 BEQ NXTCLOS ; NO ERR IF CLOSE ALL
014640 BCS CLOSEERR
014641 NXTCLOS LDA FCBPTR ; BUMP POINTER TO NEXT FILE CONTROL BLOCK.
014642 CLC
014643 ADC #$20
014644 BCC CLSALL1 ; BRANCH IF WITHIN SAME PAGE.
014645 LDA FCBPTR+1
014646 INC FCBPTR+1 ; BUMP TO NEXT PAGE.
014647 CMP FCBADDRH ; HAVE WE CHECKED BOTH PAGES?
014648 BEQ CLOSALL ; YES, RETURN NO ERROR.
014649 CLC
014650 LDA CFERR ; ON FINAL CLOSE OF CLOSE ALL REPORT LOGGED ERRORS
014651 BEQ C3 ; BRANCH IF NO ERRORS
014652 SEC
014653 C3 RTS
014654 *
014655 *
014656 CFERR DS 1 ; GLOBAL ERROR FLAG FOR FLUSH AND CLOSE ALL
014657 *
014658 *

```

```

014659 CLOSE1      JSR      FLUSH1      ; FLUSH FILE FIRST (INCLUDING UPDATING BIT MAP)
014660             BCS      CLOSEERR
014661 CLOSE2      LDY      #FCBBUFN
014662             LDA      (FCBPTR),Y
014663             JSR      RELBUF
014664             BCS      CLOSEERR
014665             LDA      #0
014666             LDY      #FCBREFN
014667             STA      (FCBPTR),Y
014668             INY                      ; BUMP TO 'FCBDEVN'
014669             LDA      (FCBPTR),Y
014670             STA      DEVNUM          ; GO LOOK FOR ASSOCIATED VCB.
014671             JSR      DEVVCB
014672             LDX      VCBPTR          ; GET VCBPTR
014673             DEC      VCB+VCBOPNC,X   ; INDICATE ONE LESS FILE OPEN.
014674             BNE      CLOSEND        ; BRANCH IF THAT WASN'T THE LAST...
014675             LDA      VCB+VCBSTAT,X
014676             AND      #$7F           ; STRIP 'FILES OPEN' BIT
014677             STA      VCB+VCBSTAT,X
014678 CLOSEND     CLC
014679             RTS
014680 CLOSERR     JMP      GLBERR          ; DON'T REPORT CLOSALL ERR NOW
014681 *
014682             PAGE
014683 *
014684 FLUSH        LDA      C.REFNUM        ; FLUSH ALL?
014685             BNE      FLUSH1          ; NO, JUST ONE OF 'EM
014686             STA      CFERR          ; CLEAR GLOBAL FLUSH ERROR
014687             JSR      GFCBADR        ; SET UP POINTER TO FCB
014688 FLSHALL     LDA      #0              ; BEGIN AT THE BEGINNING.
014689 FLSHAL1     STA      FCBPTR          ; SAVE CURRENT LOW BYTE OF POINTER
014690             LDY      #FCBREFN        ; INDEX TO REFERENCE NUMBER
014691             LDA      (FCBPTR),Y       ; IS THIS REFERENCE FILE OPEN?
014692             BEQ      NXFLUSH         ; NO, TRY NEXT.
014693             JSR      FLUSH2          ; CLEAN IT OUT...
014694             BCS      FLSHERR        ; RETURN ANY ERRORS
014695 *
014696             BCS      CLOSEERR
014697 NXFLUSH     LDA      FCBPTR          ; BUMP POINTER TO NEXT FILE CONTROL BLOCK.
014698             CLC
014699             ADC      #$20
014700             BCC      FLSHAL1         ; BRANCH IF WITHIN SAME PAGE.
014701             LDA      FCBPTR+1
014702             INC      FCBPTR+1        ; BUMP TO NEXT PAGE.
014703             CMP      FCBADDRH        ; HAVE WE CHECKED BOTH PAGES?
014704             BEQ      FLSHALL        ; YES, RETURN NO ERROR.
014705 FLUSHEND    CLC
014706             LDA      CFERR          ; ON LAST FLUSH OF A FLUSH(0)
014707             BEQ      F3              ; BRANCH IF NO LOGGED ERRORS
014708             SEC                      ; REPORT ERROR NOW

```

```

014709 F3          RTS
014710 FLSHERR     JMP          GLBERR          ; FLUSH ALL OR ONE?
014711 *
014712 FLUSH2     JSR          FNDFCBUF       ; MUST SET UP ASSOCIATED VCB AN BUFFER LOCATIONS FIRST.
014713           BCC          FLUSH2A        ; BRANCH IF NO ERROR ENCOUNTERED.
014714           JMP          GLBERR          ; CHECK FOR CLOSE OR FLUSH ALL
014715 *
014716 FLUSH1     LDA          #0             ; CLEAR
014717           STA          CFERR           ; GLOBAL ERROR FOR NORMAL REFNUM FLUSH
014718           JSR          FINDFCB        ; SET UP POINTER TO FCB USER REFERENCES
014719           BCS          FLSHERR        ; RETURN ANY ERRORS
014720 FLUSH2A     LDY          #FCBATTR       ; TEST TO SEE IF FILE IS
014721           LDA          (FCBPTR),Y      ; MODIFIED. FIRST TEST WRITE ENABLED.
014722           AND          #WRITEN
014723           BEQ          FLUSHEND       ; BRANCH IF 'READ ONLY'
014724           LDY          #FCBDIRTY      ; SEE IF EOF HAS BEEN MODIFIED
014725           LDA          (FCBPTR),Y
014726           BMI          FLUSH2B        ; BRANCH IF IT HAS
014727           LDY          #FCBSTAT        ; NOW TEST FOR DATA MODIFIED.
014728           LDA          (FCBPTR),Y      ; (IN OTHER WORDS: WAS FILE ACTUALLY
014729           AND          #USEMOD+EOFMOD+DATMOD ; WRITTEN TO WHILE IT'S BEEN OPEN?)
014730           BEQ          FLUSHEND       ; BRANCH IF FILE NOT MODIFIED.
014731 FLUSH2B     JSR          TWRPROT1      ; DISK SWITCH CHECKING
014732           LDA          DSWGLOB
014733           BEQ          FLUSH2C        ; BRANCH IF NO SWITCH
014734           LDA          #XDISKSW
014735           SEC
014736           RTS                          ; FORCES A VERIFIED RETRY
014737 FLUSH2C     LDY          #FCBSTAT        ; NOW TEST FOR DATA MODIFIED.
014738           LDA          (FCBPTR),Y
014739           AND          #DATMOD
014740           BEQ          FLUSH3         ; DOES CURRENT DATA BUFFER NEED TO BE
014741           JSR          WFCBDAT        ; WRITTEN? BRANCH IF NOT.
014742           BCS          FLSHERR        ; IF SO, GO WRITE IT STUPID!
014743 FLUSH3     LDY          #FCBSTAT        ; CHECK TO SEE IF THE INDEX BLOCK (TREE FILES ONLY)
014744           LDA          (FCBPTR),Y      ; NEEDS TO BE WRITTEN.
014745           AND          #IDXMOD
014746           BEQ          FLUSH4         ; BRANCH IF NOT...
014747           JSR          WFCBIDX
014748           BCS          FLSHERR        ; RETURN ANY ERRORS.
014749           PAGE
014750 *
014751 FLUSH4     LDY          #FCBENIN        ; NOW PREPARE TO UPDATE DIRECTORY
014752 OWNRMV     LDA          (FCBPTR),Y      ; NOTE: THIS CODE DEPENDS ON THE
014753           STA          D.DEV-FCBDEVN,Y ; DEFINED ORDER OF THE FILE CONTROL
014754           DEY                          ; BLOCK AND THE TEMPORARY DIRECTORY AREA IN 'WORKSPC'! *****
014755           CPY          #FCBDEVN-1
014756           BNE          OWNRMV
014757           LDA          D.HEAD          ; READ IN THE DIRECTORY HEADER FOR THIS FILE
014758           STA          BLOKNML

```

```

014759      LDA      D.HEAD+1
014760      STA      BLOKNMH
014761      LDA      D.DEV
014762      STA      DEVNUM
014763      JSR      RDGBUF          ; READ IT INTO THE GENERAL PURPOSE BUFFER
014764      BCS      FLSHERR        ; BRANCH IF ERROR.
014765      JSR      MOVHED0        ; MOVE HEADER INFO.
014766      LDA      D.ENTBLK       ; GET ADDRESS OF DIRECTORY BLOCK THAT
014767      LDY      D.ENTBLK+1     ; CONTAINS THE FILE ENTRY.
014768      CMP      D.HEAD        ; TEST TO SEE IF IT'S THE SAME BLOCK THAT
014769      BNE      FLSHEBLK      ; THE HEADER IS IN. BRANCH IF NOT.
014770      CPY      D.HEAD+1
014771      BEQ      FLUSH5        ; BRANCH IF HEADER BLOCK = ENTRY BLOCK.
014772      FLSHEBLK  STA      BLOKNML
014773      STY      BLOKNMH
014774      JSR      RDGBUF          ; GET BLOCK WITH FILE ENTRY IN GENERAL BUFFER.
014775      FLUSH5   JSR      ENTCALC      ; SET UP POINTER TO ENTRY
014776      JSR      MOVENTRY      ; MOVE ENTRY TO TEMP ENTRY BUFFER IN 'WORKSPC'
014777      LDY      #FCBUSE        ; UPDATE 'BLOCKS USED' COUNT.
014778      LDA      (FCBPTR),Y
014779      STA      DFIL+D.USAGE
014780      INY
014781      LDA      (FCBPTR),Y
014782      STA      DFIL+D.USAGE+1   ; HI BYTE TOO...
014783      LDY      #FCBEOF        ; AND MOVE IN END OF FILE MARK WHETHER
014784      EOFUPDTE LDA      (FCBPTR),Y   ; WE NEED TO OR NOT.
014785      STA      DFIL+D.EOF-FCBEOF,Y
014786      INY          ; MOVE ALL THREE BYTES.
014787      CPY      #FCBEOF+3
014788      BNE      EOFUPDTE
014789      LDY      #FCBFRST        ; ALSO MOVE IN THE ADDRESS OF
014790      LDA      (FCBPTR),Y     ; THE FILE'S FIRST BLOCK SINCE
014791      INY          ; IT MIGHT HAVE CHANGED SINCE THE FILE
014792      STA      DFIL+D.FRST     ; FIRST OPENED.
014793      LDA      (FCBPTR),Y
014794      STA      DFIL+D.FRST+1
014795      PAGE
014796      LDY      #FCBSTYP       ; AND THE LAST THING TO UPDATE IS
014797      LDA      (FCBPTR),Y     ; THE STORAGE TYPE.
014798      ASL      A              ; (SHIFT IT INTO THE HI NIBBLE)
014799      ASL      A
014800      ASL      A
014801      ASL      A
014802      STA      SCRTCH
014803      LDA      DFIL+D.STOR    ; GET OLD TYPE BYTE (IT MIGHT BE THE SAME)
014804      AND      #$F            ; STRIP OFF OLD TYPE
014805      ORA      SCRTCH        ; ADD IN THE NEW TYPE,
014806      STA      DFIL+D.STOR    ; AND PUT IT AWAY.
014807      JSR      DREVERSE      ; GO UPDATE DIRECTORY!
014808      BCS      FLUSHERR

```



```

014809          LDY          #FCBDIRTY          ; MARK
014810          LDA          (FCBPTR),Y        ; FCB/DIRECTORY
014811          AND          #$FF-FCBMOD       ; AS
014812          STA          (FCBPTR),Y        ; UNDIRTY
014813          LDX          #0                 ; NOW CHECK TO SEE IF A BIT MAP
014814          LDA          D.DEV             ; IS LYING AROUND THAT SHOULD BE WRITTEN.
014815          CMP          BMADEV           ; IS IT IN MAP BUFFER A?
014816          BEQ          BMAPUP           ; YES, PUT IT ON THE DISK IF NECESSARY.
014817          LDX          #BMTABSZ          ; SET INDEX TO BIT MAP TABLE 'B'
014818          CMP          BMBDEV           ; NO, WHAT ABOUT BIT MAP BUFFER B?
014819          BNE          FLSHEND1         ; NOPE, ALL DONE.
014820  BMAPUP          LDA          BMASTAT,X   ; TEST TO SEE IF IT'S BEEN MODIFIED.
014821          BPL          FLSHEND1         ; NOPE, ALL DONE AS I SAID.
014822          STX          BMTAB
014823          JSR          WRTBMAP          ; GO PUT IT AWAY.
014824          BCS          FLUSHERR
014825          LDX          BMTAB            ; MARK MAP AS UPDATED
014826          LDA          #0
014827          STA          BMASTAT,X
014828  FLSHEND1       CLC
014829          RTS
014830  FLUSHERR       EQU          *           ; DROP INTO GLBERR
014831  *
014832  GLBERR          EQU          *           ; REPORT ERROR IMMEDIATELY
014833  * ONLY IF NOT A CLOSE ALL OR FLUSH ALL
014834          LDX          C.REFNUM
014835          BNE          GLBERR1           ; NOT AN 'ALL' SO REPORT NOW
014836          CLC
014837          STA          CFERR            ; SAVE FOR LATER
014838  GLBERR1        RTS
014839  *
014840  *
014841  GFCBADR          LDA          FCBANKNM    ; GET BANK THAT FCB IS IN
014842          STA          SISFCBP
014843          LDA          FCBADDRH         ; AND HIGH BYTE ADDRESS OF FILE CONTORL BLOCK.
014844          STA          FCBPTR+1
014845          RTS                          ; SILLY THAT IT'S SO SHORT...
014846  *
014847  SETERR          LDA          #ACCSERR
014848          SEC
014849  EOFRETN         RTS
014850          PAGE
014851  *
014852  SETEOF           LDY          #FCBSTYP    ; ONLY KNOW HOW TO MOVE EOF OF TREE TYPE
014853          LDA          (FCBPTR),Y
014854          CMP          #TRETYP+1
014855          BCS          SETERR            ; BRANCH IF OTHER THAN TREE
014856          LDY          #FCBATTR
014857          LDA          (FCBPTR),Y
014858          AND          #WRITEN          ; CAN WE SET NEW EOF?

```

```

014859      BEQ      SETERR      ; NOPE, ACCESS ERROR.
014860      JSR      TSTWPROT    ; FIND OUT IF MOD IS POSSIBLE (HARDWARE WRITE PROTECT)
014861      BCS      SETERR
014862      LDY      #FCBEOF+2    ; SAVE OLD EOF
014863      LDX      #2           ; SO IT CAN BE SEEN
014864  SETSAVE  LDA      (FCBPTR),Y ; WHETHER BLOCKS NEED
014865      STA      OLDEOF,X     ; TO BE RELEASED
014866      DEY
014867      DEX
014868      BPL      SETSAVE      ; ALL THREE BYTES OF THE EOF
014869      JSR      ADJMARK      ; GET ADJUSTED END OF FILE ACCORDING TO 'C.BASE' INTO TPOS.
014870      BCS      EOFRETN     ; RETURN ANY ERROR IMMEDIATELY
014871      LDX      #2
014872  NEOFPOS  LDA      TPOSLL,X ; POSITION MARK TO NEW EOF
014873      STA      C.NEWEOF,X
014874      DEX
014875      BPL      NEOFPOS
014876      LDY      #FCBMARK+2  ; FIND OUT IF EOF < MARK.
014877      LDX      #2
014878  NEOFST   LDA      (FCBPTR),Y
014879      CMP      C.NEWEOF,X   ; COMPARE UNTIL NOT EQUAL OR CARRY CLEAR
014880      BCC      SETEOF1      ; BRANCH IF EOF>MARK
014881      BNE      SETEOF0      ; BRANCH IF EOF<MARK
014882      DEY
014883      DEX
014884      BPL      NEOFST       ; LOOP ON ALL THREE BYTES
014885  SETEOF0  JSR      RDPOSN    ; READ IN NEW POSITION.
014886      BCS      EOFRETN     ; RETURN ANY ERRORS.
014887  SETEOF1  LDX      #2
014888      LDY      #FCBEOF+2    ; MOVE NEW EOF TO FCB.
014889  SETEOF2  LDA      C.NEWEOF,X
014890      STA      (FCBPTR),Y
014891      DEY
014892      DEX
014893      BPL      SETEOF2      ; MOVE ALL THREE BYTES.
014894      JSR      FCBUSED      ; MARK FCB AS DIRTY (FOR FLUSH)
014895      *
014896      LDX      #2           ; POINT TO THIRD BYTE
014897  PURTEST  LDA      OLDEOF,X ; SEE IF EOF MOVED BACKWARDS
014898      CMP      C.NEWEOF,X   ; SO BLOCKS CAN
014899      BCC      PURTEST1     ; BE RELEASED (BRANCH IF NOT)
014900      BNE      PURGE        ; BRANCH IF BLOCKS TO BE RELEASED
014901      DEX
014902      BPL      PURTEST      ; ALL THREE BYTES
014903  PURTEST1 JMP      FLSHEND1         ; NEW EOF NOT SMALLER
014904  TRELEAS1 JMP      TRELEASE          ; OVERFLOW PREVENTER
014905      *
014906  PURGE    LDY      #FCBSTYP  ; FIND OUT WHAT TYPE OF TREE
014907      LDA      (FCBPTR),Y    ; TO PERFORM THE PROPER
014908      CMP      #SEEDTYP      ; STYLE OF BLOCK RELEASE

```

```

014909          BEQ          EOFOUT          ; SEED DON'T DEALLOCATE
014910          CMP          #TRETYP        ; FULL TREE?
014911          BEQ          TRELEAS1       ; BRANCH IF YES
014912 *
014913 * IF WE GET HERE, WE ARE RELEASING
014914 * BLOCKS AT THE END OF A SAPLING FILE: CALCULATE CORRECT POSITION
014915 * WITHIN THE INDEX BLOCK AND ALLOW SUBROUTINE
014916 * PURGE LATTER BLOCKS TO DEALLOCATE
014917 * ALL THE DATA BLOCKS THAT FOLLOW
014918 *
014919          JSR          FNDBMAP          ; REFRESH THE RIGHT MAP FOR THIS VOLUME
014920          LDX          TPOSHI         ; PRELOAD
014921          LDY          TPOSLH         ; THE THREE EOF
014922          LDA          TPOSLH         ; BYTES
014923          BNE          PUR1           ; BRANCH IF NO BOUNDARY ADJUSTMENT NEEDED
014924          CPY          #0
014925          BNE          PUR2           ; MIDDLE BYTE ZERO MEANS NO CARRY
014926          CPX          #0           ; ALL BYTES ZERO??
014927          BEQ          PUR1           ; BRANCH IF YES
014928          DEX
014929 *
014930 * THESE LINES IF CODE, SOMEWHAT CRYPTIC,
014931 * CALCULATE THE POINT AT WHICH THE
014932 * LAST BLOCK CONTAINING THE LAST BIT
014933 * OF DATUM
014934 *
014935 * THE FOLLOWING IS ROUGHLY A /512
014936 * ALGORITHM
014937 *
014938 PUR2          DEY
014939 PUR1          TXA
014940          LSR          A
014941          TYA
014942          ROR          A
014943 *
014944          JSR          PURLBLKS         ; MAKES A GOOD PTR TO DO THE RELEASING
014945          LDY          #FCBSTAT        ; MARK INDEX BLOCK
014946          LDA          (FCBPTR),Y     ; AS DIRTY
014947          ORA          #IDXMOD
014948          STA          (FCBPTR),Y
014949          LDA          PURUSE         ; INDICATE NEW NUMBER OF BLOCKS USED
014950          CLC
014951          ADC          #2             ; ACCOUNT FOR CARDINAL AND INDEX
014952          LDY          #FCBUSE
014953          STA          (FCBPTR),Y     ; FILE LOW BYTE
014954          INY
014955          LDA          #0             ; ANTICIPATE <257 BLOCKS
014956          BCC          PURHI
014957          LDA          #1             ; >256 BLOCKS IN FILE
014958 PURHI        STA          (FCBPTR),Y ; HIGH BYTE BLOCKS USED

```

```

014959 EOFOUT      CLC
014960              RTS                ; NO ERRORS POSSIBLE
014961 *
014962 PURLBLKS    EQU      *          ; PURGE LATTER BLOCKS
014963 * INPUT ARG: A REGISTER CONTAINING
014964 * POINTER TO CURRENT DATA BLOCK WITHIN THE
014965 * CURRENT INDEX BLOCK (TINDX)
014966 * DEALLOCATE ALL LEGAL BLOCKS AFTER
014967 * THE A REGISTER PTR. NO ERRORS POSSIBLE
014968 *
014969              TAY                ; MAKE PROPER INDEX
014970              STY      PURUSE      ; INDICATES NUMBER OF BLOCKS IN USE IN FILE
014971 PURLOOP      INY                ; POINT TO A PTR TO DATA BLK TO DEALLOCATE
014972              BEQ      PURLRSTS    ; NO MORE BLOCKS IN INDEX
014973              INC      TINDX+1     ; GET HIGH PART OF BLOCK ADDR
014974              LDA      (TINDX),Y
014975              TAX                ; X IS A PASSING PARM
014976              LDA      #0         ; TELL INDEX BLOCK THAT THE DATA
014977              STA      (TINDX),Y   ; BLOCK IS NOW FREE
014978              TXA
014979              DEC      TINDX+1     ; AND LOW PART
014980              ORA      (TINDX),Y
014981              BEQ      PURLOOP     ; INDICATED ADDR WAS ZERO-ZERO
014982              LDA      (TINDX),Y   ; A REG IS ANOTHER PASSING PARM
014983              PHA
014984              LDA      #0
014985              STA      (TINDX),Y   ; AND SET LOW DATA ADDR AS FREED
014986              PLA
014987              STY      PURPLACE    ; TEMP STORAGE
014988              JSR      DEALLOC     ; DEALLOCATE BLOCK (ADDR: A (LOW), X ( HIGH)
014989              LDY      #VCBTFRE
014990              CLC
014991              LDA      (VCBPTR),Y   ; ADJUST NUMBER OF FREE BLOCKS ON VOLUME
014992              ADC      #1
014993              STA      (VCBPTR),Y
014994              INY
014995              LDA      (VCBPTR),Y   ; HIGH BYTE OF TOTAL FREE
014996              ADC      #0
014997              STA      (VCBPTR),Y
014998              LDY      PURPLACE
014999              JMP      PURLOOP
015000 PURLRSTS    RTS
015001 PURUSE      DS      1          ; CURRENT NUMBER OF BLOCKS USED
015002 PURPLACE    DS      1          ; CURRENT PLACE IN RELEASE-BLOCK CYCLE
015003 TRELEASE    EQU      *
015004              JMP      EOFOUT     ; RELEASE TWO LEVEL TREE CODE GOES HERE
015005 *
015006 GETEOF      LDY      #FCBEOF    ; INDEX TO END OF FILE MARK
015007              LDX      #0         ; WE'VE GOT INDIRECT BOTH WAYS (IN & OUT)
015008 OUTFEOF      LDA      (FCBPTR),Y

```

```
015009          STA      (C.OUTEOF,X)
015010          INY
015011          CPY      #FCBEOF+3
015012          BEQ      OFFRTS          ; BRANCH IF ALL THREE BYTES TRANSFERED.
015013          INC      C.OUTEOF        ; BUMP USER'S POINTER.
015014          BNE      OUTEOF
015015          INC      C.OUTEOF+1
015016          BNE      OUTEOF          ; BRANCH ALWAYS
015017 *
015018          CHN      DESTROY,4,2
015019
015020 *****
015021 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: CLOSE.EOF
015022 *****
015023
015024
```

```

015025 =====
015026 DOCUMENT :SOS1.3.4of5.FOUR:SOS.DESTROY.TEXT
015027 =====
015028
015029 *****
015030 * APPLE /// SOS 1.3 SOURCE CODE FILE: DESTROY
015031 *****
015032 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
015033
015034 PAGE
015035 *
015036 NEWLINE LDY #FCBATTR ; ADJUST NEWLINE STATUS FOR OPEN FILE.
015037 LDA C.ISNEWL ; ON OR OFF?
015038 BPL OFFNEWL ; BRANCH IF NEW LINE IS TO BE CLEARED.
015039 LDA #NLINEN
015040 ORA (FCBPTR),Y ; SET NEW LINE BIT IN ATTRIBUTES
015041 STA (FCBPTR),Y
015042 LDY #FCBNEWL ; AND MOVE IN NEW 'NEW-LINE' BYTE.
015043 LDA C.NEWL
015044 STA (FCBPTR),Y
015045 CLC
015046 RTS ; NO ERROR POSSIBLE.
015047 *
015048 OFFNEWL LDA #$FF-NLINEN
015049 AND (FCBPTR),Y
015050 STA (FCBPTR),Y ; CLEAR NEW-LINE BIT.
015051 OFFRTS CLC ; THE NEW LINE CHARACTER DOES'T MATTER...
015052 RTS
015053 PAGE
015054 *
015055 GETINFO JSR FINDFILE ; LOOK FOR FILE THEY WANT OT KNOW ABOUT.
015056 BCC GTINFO1 ; BRANCH IF NO ERRORS.
015057 CMP #BADPATH ; WAS IT A ROOT DIRECTORY FILE?
015058 SEC ; (IN CASE OF NO MATCH)
015059 BNE GINFOERR
015060 LDA #$F0
015061 STA DFIL+D.STOR ; FOR GET INFO, REPORT PROPER STORAGE TYPE
015062 LDA #0 ; FORCE A COUNT OF FREE BLOCKS.
015063 STA REQL
015064 STA REQH
015065 JSR TSFRBLK ; (RETURNS IF IMMEDIATELY IF COUNT HAS PREVIOUSLY BEEN TAKEN)
015066 LDY #VCBTFRE+1
015067 LDA (VCBPTR),Y ; RETURN TOTAL BLOCKS AND TOTAL IN USE.
015068 STA REQH ; FIRST TRANSFER 'FREE' BLOCKS TO ZPAGE FOR LATER SUBTRACT
015069 DEY
015070 LDA (VCBPTR),Y ; TO DETERMINE THE 'USED' COUNT
015071 STA REQL
015072 DEY
015073 LDA (VCBPTR),Y ; TRANSFER TO 'D.' TABLE AS AUX I.D.

```

```

015074      STA      DFIL+D.AUXID+1      ; (TOTAL BLOCK COUNT IS CONSIDERED AUX I.D. FOR THE VOLUME)
015075      TAX
015076      DEY
015077      LDA      (VCBPTR),Y
015078      STA      DFIL+D.AUXID
015079      SEC
015080      SBC      REQL
015081      STA      DFIL+D.USAGE
015082      TXA
015083      SBC      REQH
015084      STA      DFIL+D.USAGE+1
015085  GTINFO1  LDY      #0
TRANSLATION
015086  GTINFO2  LDA      INFTABL,Y
015087      BPL      GTINFO3
015088      AND      #$7F
015089      BEQ      GTINFO4
015090      CMP      #D.STOR+1
015091      BNE      GINFOEND
015092      LDA      DFIL+D.STOR
015093      LSR      A
015094      LSR      A
015095      LSR      A
015096      LSR      A
015097      BPL      GTINFO4
015098      *
015099  GTINFO3  TAX
015100      LDA      DFIL,X
015101  GTINFO4  STA      (C.FILIST),Y
015102      INY
015103      CPY      C.FILSTLN
015104      BNE      GTINFO2
015105  GINFOEND  CLC
015106  GINFOERR  RTS
015107      *
015108      *
015109      PAGE
015110      *
015111  SETINFO  JSR      FINDFILE
015112      BCS      SINFOERR
015113      LDA      C.FILSTLN
015114      BEQ      SINFEND
015115      LDY      #0
015116      LDA      (C.FILIST),Y
015117      AND      #$1C
015118      BEQ      SETINF1
015119      LDA      #ACCSERR
015120      SEC
015121      RTS
015122  SETINF1  LDA      BACKMASK

```

```

015123 * BACKUP KNOWS HOW TO RESET THIS BIT. <SRS 82.162>
015124 STA BKBITFLG ; BIT (USED BY DREWISE)
015125 SETINF1X LDX INFNTABL,Y ; GET INDEX INTO CORESPONDING 'D.' TABLE
015126 BMI SETINF2 ; BRANCH IF WE'VE REACHED STORAGE TYPE PARAMETER
015127 LDA (C.FILIST),Y
015128 STA DFIL,X
015129 INY ; HAS USER'S REQUEST BEEN SATISFIED?
015130 CPY C.FILSTLN
015131 BNE SETINF1X ; NO, MOVE NEXT BYTE.
015132 SINFEND JMP DREWISE ; GO UPDATE DIRECTORY WITH CURRENT TIME.
015133 *
015134 SETINF2 LDY C.FILSTLN ; TEST TO SEE IF USER WANTS HIS TIME STAMP ADDED
015135 CPY #$F ; (LIST MUST BE AT LEAST $F BYTES LONG)
015136 BCC SINFEND ; NO PUT CURRENT TIME INSTEAD.
015137 LDY #$B ; MOVE IN THE NEXT GROUP OF BYTES
015138 SETINF3 LDX INFNTABL,Y
015139 BMI SINFEND1
015140 LDA (C.FILIST),Y
015141 STA DFIL,X
015142 INY
015143 CPY C.FILSTLN ; SATISFACTION YET?
015144 BNE SETINF3 ; NOPE, KEEP EM PUMPIN'
015145 SINFEND1 JMP DREWISE1
015146 *
015147 BKBITFLG DS 1 ; FOR TURNING OFF BACKUP BIT
015148 *
015149 *
015150 INFNTABL DFB D.ATTR,D.FILID,D.AUXID,D.AUXID+1
015151 DFB D.STOR+1+$80,D.EOF,D.EOF+1,D.EOF+2 ; (D.STOR=0 THUS D.STOR+1 WAS NECESSARY)
015152 DFB $80,D.USAGE,D.USAGE+1,D.MODDT ; (THE $80 IS FOR THE FOURTH BYTE OF EOF)
015153 DFB D.MODDT+1,D.MODTM,D.MODTM+1,$FF ; TABLE ALWAYS ENDS IN $FF
015154 PAGE
015155 *
015156 RENAME JSR LOOKFILE ; LOOK FOR SOURCE (ORIGINAL) FILE.
015157 BCC RNAME0 ; BRANCH IF FOUND.
015158 CMP #BADPATH ; TRYING TO RENAME A VOLUME?
015159 BNE RNAMEERR ; NO, RETURN OTHER ERROR.
015160 JSR RENPATH ; SYNTAX NEW NAME.
015161 BCS RNAMEERR
015162 LDA WRKPATH ; FIND OUT IF ONLY ROOTNAME FOR NEW NAME
015163 CMP PATHNML
015164 BNE RNBADPTH ; NOT SINGLE NAME, RETURN ERROR!
015165 LDY #VCBSTAT ; TEST FOR OPEN FILES BEFORE CHANGING
015166 LDA (VCBPTR),Y
015167 BPL RNAMEVOL ; BRANCH IF VOLUME NOT BUSY
015168 LDA #FILBUSY
015169 SINFOERR EQU *
015170 RTS ; (CARRY IS SET)
015171 RNAMEVOL LDY #0 ; GET NEWNAME'S LENGTH.
015172 LDA (WRKPATH),Y

```



```

015173          TAY
015174          ORA          #$F0          ; (ROOT FILE STORAGE TYPE)
015175          JSR          MVROTNAM      ; UPDATE ROOT DIRECTORY.
015176          BCS          RNAMEERR
015177          LDY          #0
015178          LDA          (WRKPATH),Y    ; UPDATE VCB ALSO.
015179          TAY
015180  RNMEVOL    LDA          (WRKPATH),Y
015181          STA          (VCBPTR),Y
015182          DEY
015183          BPL          RNMEVOL
015184          CLC
015185          RTS
015186          *
015187  RNAMEO    JSR          RENPATH      ; SET UP AND SYNTAX NEW NAME.
015188          BCS          RNAMEERR
015189          LDY          #0              ; VERIFY THAT BOTH NAMES HAVE SAME ROOT.
015190          LDA          (PATHNML),Y
015191          TAY
015192  TSTSMROT  LDA          (PATHNML),Y  ; COMPARE NEWNAME'S ROOT NAME WITH
015193          CMP          (VCBPTR),Y    ; OLD NAME'S VOLUME NAME.
015194          BNE          RNBADPTH      ; RETURN 'BADPATH' IF NOT SAME VOLUME.
015195          DEY
015196          BPL          TSTSMROT      ; (TEST SAME 'ROT')
015197          JSR          LOOKFILE      ; TEST FOR DUPLICATE FILE NAME.
015198          BCS          TSTFNF1      ; BRANCH IF ERROR TO TEST FOR FILE NOT FOUND.
015199          LDA          #DUPEERR    ; TELL USER THAT NEW NAME ALREADY EXISTS.
015200  RNAMEERR  SEC
015201          RTS
015202          PAGE
015203  TSTFNF1    CMP          #FNFERR      ; WAS IT A VALID FILE NOT FOUND?
015204          BNE          RNAMEERR      ; NO, RETURN OTHER ERROR CODE.
015205          LDX          #2            ; NOW MOVE NEW NAME'S OWNERSHIP (DIRECTORY HEADER) I.D.
015206  SVENEWID  LDA          D.DEV,X     ; THIS CONSISTS OF THE UNIT NUMBER,
015207          STA          NPATHDEV,X   ; AND THE ADDRESS OF THE DIRECTORY THE FILE
015208          DEX                      ; WASN'T FOUND IN. LOGIC BY NEGATION...
015209          BPL          SVENEWID
015210          JSR          SETPATH      ; NOW SYNTAX THE PATHNAME OF THE FILE TO BE CHANGED.
015211          BCS          RNAMEERR
015212          JSR          FINDFILE     ; GET ALL THE INFO ON THIS ONE.
015213          BCS          RNAMEERR
015214          JSR          TSTOPEN      ; DON'T ALLOW RENAME TO OCCUR IF FILE IS IN USE.
015215          LDA          #FILBUSY    ; ANTICIPATE ERROR
015216          BCS          RNAMEERR
015217          LDA          DFIL+D.ATTR  ; TEST BIT THAT SAYS IT'S OK TO RENAME
015218          AND          #RENAMEN
015219          BNE          RNAME1      ; BRANCH IF IT'S ALRIGHT TO RENAME.
015220          LDA          #ACCSERR     ; OTHERWISE REPORT ILLEGAL ACCESS.
015221          SEC
015222          RTS

```

```

015223 *
015224 RNAME1      LDX      #2          ; NOW TEST TO SEE IF NEW PATHNAME FITS IN THE
015225 SAMOWNR     LDA      D.DEV,X    ; SAME DIRECTORY FILE.
015226             CMP      NPATHDEV,X
015227             BEQ      RNAME2
015228 RNBADPTH     LDA      #BADPATH   ; TELL USER THAT PATHNAMES INCOMPATABLE.
015229             SEC
015230             RTS
015231 *
015232 RNAME2       DEX          ; TEST ALL THREE BYTES.
015233             BPL      SAMOWNR
015234             JSR      RENPATH     ; WELL... SINCE BOTH NAMES WOULD GO INTO THE
015235             BCS      RNAMEERR    ; DIRECTORY, RE-SYNTAX THE NEW NAME TO GET LOCAL NAME ADDRESS.
015236             TYA          ; (Y CONTAINS THE LOCAL NAME LENGTH+1)
015237             BEQ      RNBADPTH   ; REPORT ERROR IF LENGTH INFO NOT IMMEDIATELY AVAILABLE.
015238             DEY          ; (REMOVE THE +1)
015239 RNAME3       LDA      (WRKPATH),Y ; MOVE LOCAL NAME TO DIR ENTRY WORKSPACE.
015240             STA      DFIL+D.STOR,Y
015241             DEY
015242             BNE      RNAME3
015243             LDA      DFIL+D.STOR  ; PRESERVE FILE STORAGE TYPE.
015244             AND      #$F0       ; STRIP OFF OLD NAME LENGTH.
015245             TAX
015246             ORA      (WRKPATH),Y ; ADD IN NEW NAME'S LENGTH
015247             STA      DFIL+D.STOR
015248             CPX      #DIRTYP*16 ; THAT FILE MUST BE CHANGED ALSO.
015249             BNE      RNAMEDONE  ; BRANCH IF NOT DIRECTORY TYPE.
015250             PAGE
015251             LDA      DFIL+D.FRST  ; READ IN FIRST (HEADER) BLOCK OF SUB DIRECTORY
015252             STA      BLOKNML
015253             LDA      DFIL+D.FRST+1
015254             STA      BLOKNMH
015255             JSR      RDGBUF
015256             BCS      RNAMEERR    ; REPORT ERRORS
015257             LDY      #0          ; CHANGE THE HEADER'S NAME TO MATCH THE OWNER'S NEW NAME.
015258             LDA      (WRKPATH),Y ; GET LOCAL NAME LENGTH AGAIN
015259             TAY
015260             ORA      #HEDTYP*16  ; ASSUME IT'S A HEADER.
015261             JSR      MVROTNAM
015262             BCS      RNAMEERR
015263 RNAMEDONE     JMP      DREWISE1  ; END BY UPDATING ALL PATH DIRECTORIES
015264 *
015265 *
015266 MVROTNAM      STA      GBUF+4
015267 MVHEDNAM     LDA      (WRKPATH),Y
015268             STA      GBUF+4,Y
015269             DEY
015270             BNE      MVHEDNAM
015271             JMP      WRTGBUF     ; WRITE CHANGED HEADER BLOCK.
015272 *

```

```

015273 *
015274 RENPATH      LDA      C.NWPATH      ; GET ADDRESS TO NEW PATHNAME.
015275              STA      TPATH
015276              LDA      C.NWPATH+1    ; SET UP FOR SYNTAXING ROUTINE (SYNPATH).
015277              STA      TPATH+1
015278              LDA      SSNWPATH      ; (MOVE BYTE FOR SISTER PAGE, TOO.)
015279              STA      SISTPATH
015280              JMP      SYNPATH      ; GO SYNTAX IT. (RETURNS LAST LOCAL NAME LENGTH IN Y).
015281 *
015282 *
015283 DEALBLK      LDY      #0              ; BEGIN AT THE BEGINNING.
015284 DALBLK1      STY      SAPTR          ; SAVE CURRENT INDEX.
015285              LDA      GBUF,Y         ; GET ADDRESS (LOW) OF BLOCK TO BE DEALLOCATED.
015286              CMP      GBUF+$100,Y   ; TEST FOR NUL BLOCK.
015287              BNE      DALBLK2      ; BRANCH IF NOT NUL.
015288              CMP      #0
015289              BEQ      DALBLK3      ; SKIP IT IF NUL.
015290 DALBLK2      LDX      GBUF+$100,Y   ; GET THE REST OF THE BLOCK ADDRESS.
015291              JSR      DEALLOC        ; FREE IT UP ON VOLUME BIT MAP.
015292              BCS      DALBLKERR    ; RETURN ANY ERROR.
015293              LDY      SAPTR          ; GET INDEX TO SAPLING LEVEL INDEX BLOCK AGAIN.
015294 DALBLK3      INY
015295              BNE      DALBLK1      ; BRANCH IF MORE TO DEALLOCATE (OR TEST).
015296              CLC
015297              DALBLKERR  RTS          ; INDICATE NO ERROR.
015298 *
015299 *
015300              PAGE
015301 *
015302 DESTROY      JSR      FINDFILE      ; LOOK FOR FILE TO BE WIPED OUT.
015303              BCS      DESTERR      ; PASS BACK ANY ERROR.
015304              JSR      TSTOPEN      ; IS THIS FILE OPEN?
015305              LDA      TOTENT
015306              BEQ      DSTROY1      ; BRANCH IF FILE NOT OPEN.
015307              LDA      #FILBUSY
015308              SEC
015309              RTS          ; INFORM USER THAT FILE CAN'T BE DESTROYED AT THIS TIME.
015310 *
015311 DSTROY1      LDA      #0              ; FORCE PROPER FREE COUNT IN VOLUME.
015312              STA      REQL          ; (NO DISK ACCESS OCCURS IF ALREADY PROPER)
015313              STA      REQH
015314              JSR      TSFRBLK
015315              BCC      DSTROY2
015316              CMP      #OVRERR      ; WAS IT JUST A FULL DISK?
015317              SEC
015318              BNE      DESTERR      ; NOPE, REPORT ERROR.
015319 *
015320 DSTROY2      LDA      DFIL+D.ATTR    ; MAKE SURE IT'S OK TO DESTROY THIS FILE.
015321              AND      #DSTROYEN
015322              BNE      DSTROY3      ; BRANCH IF OK.

```

```

015323          LDA      #ACCSERR          ; TELL USER IT'S NOT KOSHER.
015324          JSR      SYSERR           ; (RETURNS TO CALLER OF DESTORY)
015325          *
015326 DSTROY3    JSR      TWRPROT1       ; BEFORE GOING THRU DEALLOCATION,
015327          BCS      DESTERR          ; TEST FOR WRITE PROTECTED HARDWARE.
015328          LDA      DFIL+D.STOR      ; FIND OUT WHICH STORAGE TYPE.
015329          AND      #$F0             ; STRIP OFF NAME LENGTH.
015330          CMP      #TRETYP+1*$10    ; IS IT A SEED, SAPLING, OR TREE?
015331          BCC      DSTREE           ; BRANCH IF IT IS.
015332          JMP      DSTDIR          ; OTHERWISE TEST FOR DIRECTORY DESTROY.
015333          *
015334 DSTREE     JSR      GTTINDX        ; GET A BIT MAP BUFFER AND TEMPORARY INDEX BUFFER.
015335          BCS      DESTERR          ;
015336          LDA      DFIL+D.STOR      ; GET STORAGE TYPE AGAIN
015337          AND      #$F0             ;
015338          CMP      #TRETYP*$10      ; IS THIS A TREE (FULL 2-LEVEL)?
015339          BNE      DSTSAP           ; NO, TEST FOR SAPLING.
015340          JSR      RDFRST          ; READ IN ROOT INDEX FOR THIS FILE.
015341          BCC      DSTRE2          ; BRANCH IF ALL IS WELL.
015342 DESTERR    RTS                  ; OTHERWISE RETURN ERROR.
015343          *
015344 DSTSAP      CMP      #SAPTYP*$10    ; IS IT A SAPLING
015345          BNE      DSTLAST          ; NO, JUST DEALLOCATE FIRST (AND ONLY) BLOCK.
015346          JSR      ZTMPIDX         ; CLEAR OUT TEMPORARY INDEX BUFFER.
015347          LDA      DFIL+D.FRST      ; MAKE THIS SAP LOOK LIKE A TREE...
015348          LDY      #0                ; THIS IS DONE BY PLACING THE FIRST BLOCK ADDRESS
015349          STA      (TINDX),Y        ; IN THE TEMP (TOP) INDEX BUFFER AS
015350          INC      TINDX+1
015351          LDA      DFIL+D.FRST+1    ; A SUB INDEX WOULD APPEAR.
015352          STA      (TINDX),Y
015353          DEC      TINDX+1
015354 DSTRE2     LDY      #0                ; BEGIN SCAN OF TOP LEVEL INDEX AT ZERO.
015355 DSTNX1      STY      TREPTR         ; SAVE POINTER TO TREE LEVEL.
015356          LDA      (TINDX),Y        ; GET BLOCK ADDRESS OF A SUB INDEX BLOCK
015357          INC      TINDX+1          ; (TEST FOR NUL BLOCK)
015358          CMP      (TINDX),Y
015359          BNE      DSTRE3          ; BRANCH IF WE'VE GOT AN BLOCK TO DEALLOCATE.
015360          CMP      #0                ; IS ENTIRE ADDRESS ZERO?
015361          BEQ      DSTRE4          ; YES, DO NEXT. (CARRY SET)
015362 DSTRE3     CLC                    ; INDICATE THERE IS A BLOCK OF INDEXES TO FREE UP.
015363          STA      BLOKNML
015364          LDA      (TINDX),Y        ; GET HI ADDRESS TOO.
015365          STA      BLOKNMH
015366 DSTRE4     DEC      TINDX+1        ; (RESTORE PROPER ADDRESS FOR BUFFER)
015367          BCS      DSTNX1          ; BRANCH IF NO SUB INDEX.
015368          JSR      RDGBUF          ; USE GENERAL BUFFER FOR SUB INDEX BUFFER.
015369          BCS      DESTERR          ;
015370          JSR      DEALBLK         ; GO FREE UP BLOCKS IN SUB INDEX
015371          BCS      DESTERR          ;
015372          LDY      TREPTR          ; AND FREE UP SUB INDEX BLOCK TOO.

```

```

015373          INC          TINDX+1
015374          LDA          (TINDX),Y
015375          TAX
015376          DEC          TINDX+1
015377          LDA          (TINDX),Y
015378          JSR          DEALLOC
015379          BCS          DESTERR
015380          LDY          TREPTR
015381 DSTNXT1      INY                ; HAVE ALL SUB INDEXES BEEN LOCATED?
015382          BNE          DSTNXT      ; NO, DO NEXT...
015383 DSTLAST      LDA          DFIL+D.FRST      ; DEALLOCATE FIRST BLOCOK OF FILE.
015384          LDX          DFIL+D.FRST+1
015385          JSR          DEALLOC
015386          BCS          DESTERR
015387          LDA          #0                ; UPDATE DIRECTORY TO FREE ENTRY SPACE.
015388          STA          DFIL+D.STOR
015389          CMP          H.FCNT          ; FILE ENTRY WRAP?
015390          BNE          DST1          ; BRANCH IF NO CARRY ADJUSTMENT
015391          DEC          H.FCNT+1        ; TAKE CARRY FROM HIGH BYTE OF FILE ENTRIES
015392 DST1        DEC          H.FCNT          ; MARK HEADER WITH ONE LESS FILE
015393          LDX          BMTAB          ; UPDATE (LAST) BITMAP.
015394          JSR          BMAPUP
015395          BCS          DESTERR
015396          LDY          #VCBTFRE
015397          LDA          DFIL+D.USAGE
015398          ADC          (VCBPTR),Y
015399          STA          (VCBPTR),Y      ; UPDATE CURRENT FREE BLOCK COUNT.
015400          INY
015401          LDA          DFIL+D.USAGE+1
015402          ADC          (VCBPTR),Y
015403          STA          (VCBPTR),Y
015404          LDA          #0                ; FORCE RESCAN FROM FIRST BITMAP
015405          LDY          #VCBCMAP
015406          STA          (VCBPTR),Y
015407          JMP          DREVERSE      ; UPDATE DIRECTORY LAST...
015408 *
015409          PAGE
015410 *
015411 DSTDIR        CMP          #DIRTYP*16      ; IS THIS A DIRECTORY FILE?
015412          BEQ          DSDIR1          ; YES, PROCEED.
015413          LDA          #CPTERR          ; FILE IS NOT COMPATABLE.
015414          JSR          SYSERR          ; GIVE UP.
015415 *
015416 DSDIR1        JSR          FNDBMAP      ; MAKE SURE A BUFFER IS AVAILABLE FOR THE BITMAP.
015417          BCS          DSDIRERR
015418          LDA          DFIL+D.FRST      ; READ IN FIRST BLOCK OF DIRECTORY INTO GBUF.
015419          STA          BLOKNML
015420          LDA          DFIL+D.FRST+1
015421          STA          BLOKNMH
015422          JSR          RDGBUF

```

```

015423          BCS      DSDIRERR
015424          LDA      GBUF+HCENT+4      ; FIND OUT IF ANY FILES EXIST ON THIS DIRECTORY.
015425          BNE      DSDIRACC          ; BRANCH IF ANY EXIST.
015426          LDA      GBUF+HCENT+5
015427          BEQ      DSDIR2
015428 DSDIRACC  LDA      #ACCSERR
015429          JSR      SYSERR
015430          *
015431 DSDIR2    LDA      GBUF+2            ; GET FORWARD LINK.
015432          CMP      GBUF+3            ; TEST FOR NO LINK.
015433          BNE      DSDIR3
015434          CMP      #0
015435          BEQ      DSTLAST          ; IF NO LINK, THEN FINISHED.
015436 DSDIR3    LDX      GBUF+3
015437          JSR      DEALLOC          ; FREE THIS BLOCK.
015438          BCS      DSDIRERR
015439          LDA      GBUF+2
015440          STA      BLOKNML
015441          LDA      GBUF+3
015442          STA      BLOKNMH          ; READ IN LINKED BLOCK.
015443          JSR      RDGBUF
015444          BCC      DSDIR2          ; LOOP UNTIL ALL ARE FREED.
015445 DSDIRERR  RTS
015446          *
015447          *
015448          PAGE
015449 WORKSPC   EQU      *
015450 V.STATUS   DS      1                ; VOLUME STATUS, INCLUDES 'ACTIVE' IN BIT 7
015451 H.CREDT   DS      2                ; DIRECTORY CREATION DATE
015452          DS      2                ; DIRECTORY CREATION TIME
015453          DS      1                ; VERSION UNDER WHICH THIS DIRECTORY WAS CREATED
015454          DS      1                ; EARLIEST VERSION THAT IT'S COMPATABLE WITH
015455 H.ATTR     DS      1                ; ATTRIBUTES (PROTECT BIT, ETC.)
015456 H.ENTLN    DS      1                ; LENGTH OF EACH ENTRY IN THIS DIRECTORY.
015457 H.MAXENT   DS      1                ; MAXIMUM NUMBER OF ENTRIES PER BLOCK
015458 H.FCNT     DS      2                ; CURRENT NUMBER OF FILES IN THIS DIRECTORY
015459          DS      2                ; ADDRESS OF FIRST ALLOCATION BIT MAP
015460          DS      2                ; TOTAL NUMBER OF BLOCKS ON THIS UNIT
015461          DS      5                ; (FOR FUTURE EXPANSION)
015462          *
015463 D.DEV       DS      1                ; DEVICE NUMBER OF THIS DIRECTORY ENTRY
015464 D.HEAD     DS      2                ; ADDRESS OF <SUB> DIRECTORY HEADER
015465 D.ENTBLK   DS      2                ; ADDRESS OF BLOCK WHICH CONTAINS THIS ENTRY
015466 D.ENTNUM   DS      1                ; ENTRY NUMBER WITHIN BLOCK.
015467 DFIL      EQU      *
015468 D.STOR     EQU      *-DFIL          ; STORAGE TYPE * 16 + FILE NAME LENGTH
015469          DS      1
015470 ; *-DFIL ; FILE NAME
015471          DS      15
015472 D.FILID    EQU      *-DFIL          ; USER'S IDENTIFICATION BYTE

```

```

015473          DS          1
015474 D.FRST      EQU      *-DFIL          ; FIRST BLOCK OF FILE
015475          DS          2
015476 D.USAGE     EQU      *-DFIL          ; NUMBER OF BLOCKS CURRENTLY ALLOCATED TO THIS FILE
015477          DS          2
015478 D.EOF        EQU      *-DFIL          ; CURRENT END OF FILE MARKER
015479          DS          3
015480 D.CREDIT     EQU      *-DFIL          ; DATE OF FILE'S CREATION
015481          DS          2
015482 ; *-DFIL ; TIME OF FILE'S CREATION
015483          DS          2
015484 ; EQU *-DFIL ; SOS VERSION THAT CREATED THIS FILE
015485          DS          1
015486 D.COMP       EQU      *-DFIL          ; BACKWARD VERSION COMPATABILITY
015487          DS          1
015488 D.ATTR       EQU      *-DFIL          ; 'PROTECT', READ/WRITE 'ENABLE' ETC.
015489          DS          1
015490 D.AUXID      EQU      *-DFIL          ; USER AUXILLARY IDENTIFACATION
015491          DS          2
015492 D.MODDT      EQU      *-DFIL          ; FILE'S LAST MODIFICATION DATE
015493          DS          2
015494 D.MODTM      EQU      *-DFIL          ; FILE'S LAST MODIFICATION TIME
015495          DS          2
015496 D.DHDR      EQU      *-DFIL          ; HEADER BLOCK ADDRESS OF FILE'S DIRECTORY
015497          DS          2
015498 *
015499 CMDADR      DS          2
015500 SCRATCH      DS          13           ; SCRATCH AREA FOR ALLOCATION ADDRESS CONVERSION
015501 OLDEOF      DS          3           ; TEMP USED IN W/R
015502 OLDMARK     DS          3           ; USED BY 'RDPOSN' AND 'WRITE'
015503 SCRHIGH     EQU      <SCRATCH       ; AND DEVICE NUMBERS FROM BOB'S CODE.
015504 *
015505          CHN          SWAPOUT/IN,4,2
015506
015507 *****
015508 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: DESTROY
015509 *****
015510
015511

```

```

015512 =====
015513 DOCUMENT :SOS1.3.4of5.FOUR:SOS.POSN.OPEN.TEXT
015514 =====
015515
015516 *****
015517 * APPLE /// SOS 1.3 SOURCE CODE FILE: POSN.OPEN
015518 *****
015519 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
015520
015521 PAGE
015522 GETMARK LDY #FCBMARK ; MOVE CURRENT POSITION MARKER TO
015523 GMARK1 LDA (FCBPTR),Y ; USER'S 4 BYTE BUFFER POINTED TO BY
015524 PHA ; C.MRKPTR IN SOS ZPAGE
015525 INY
015526 CPY #FCBMARK+3 ; USE STACK AS TEMPORARY STORAGE FOR THREE BYTE
015527 BNE GMARK1 ; POSITION VALUE.
015528 LDA #0 ; THE FOURTH (HIGHEST ORDER) BYTE IS ALWAYS ZERO.
015529 LDY #3
015530 PHA
015531 MOVMRK PLA
015532 STA (C.MRKPTR),Y ; MOVE TO USER'S SPACE
015533 DEY ; IS THERE ANOTHER TO PULL FROM STACK?
015534 BPL MOVMRK ; YES, GET NEXT LOWER BYTE FROM STACK.
015535 CLC ; INDICATE NO ERROR.
015536 RTS
015537 *
015538 SETMARK JSR ADJMARK ; MAKE ADJUSTMENTS TO REQUESTED MARK ACCORDING TO BASE.
015539 BCC SMARK1 ; BRANCH IF ADJUSTMENT WAS VALID.
015540 RTS
015541 SMARK1 LDX #2 ; NOW COMPARE END OF FILE WITH NEW
015542 LDY #FCBEOF+2 ; POSITION TO BE SURE IT'S WITHIN
015543 CMPEOF LDA TPOSLL,X ; THE BOUNDS OF CURRENTLY DEFINED
015544 CMP (FCBPTR),Y ; LIMITS.
015545 BCC CKSAMBLK ; BRANCH IF MARK<EOF
015546 BNE ERRMEOF ; RETURN ERROR IF MARK>= EOF
015547 DEY
015548 DEX
015549 BPL CMPEOF
015550 BMI CKSAMBLK ; BRANCH ALWAYS
015551 ERRMEOF LDA #POSNERR ; TELL USER MARK IS OUT OF RANGE.
015552 RTS ; (CARRY IS SET TO INDICATE ERROR)
015553 *
015554 ADJMARK LDA C.MARK+3 ; MAKE SURE FOURTH BYTE OF DISPLACE IS ZIP
015555 BNE ERRPOSN ; BRANCH TO ERR IF NOT
015556 LDX #$FD ; ANTICIPATE OTHER THAN BASE OF ZERO
015557 LDY #FCBMARK ; FURTHER ASSUME IT'S A BASE OFFSET FROM CURRENT POSITION
015558 LDA C.BASE ; NOW FIND OUT WHAT IT REALLY IS.
015559 LSR A ; (CARRY SET=SUBTRACT, NON ZERO REMAINDER= OFFSET FROM EOF)
015560 BCS SUBMARK

```



```

015561          BEQ          ADJMRK          ; BRANCH IF MARK IS FROM BEGINNING OF FILE
015562  ADDPOSN          LDA          (FCBPTR),Y          ; ADD USER QUANTITY TO CURRENT
015563          ADC          C.MARK+3,X          ; POSITION TO FORM NEW POSITION.
015564          STA          >TPOSL- $\$$ FD,X          ; (NOTE: ZERO PAGE REFERENCE WRAPS AROUND IN Z-PAGE)
015565          INY
015566          INX
015567          BNE          ADDPOSN          ; ADD ALL THREE BYTES
015568          BCS          ERRPOSN          ; BRANCH IF OVERFLOW
015569          BEQ          ADJMRK1          ; BRANCH ALWAYS
015570  *
015571          PAGE
015572  SUBMARK          BNE          SUBPOSN          ; BRANCH IF IT'S AN OFFSET FROM CURRENT POSITION
015573          LDY          #FCBEOF          ; OTHERWISE ASSUME OFFSET FROM END OF FILE.
015574  SUBPOSN          LDA          (FCBPTR),Y          ; SUBTRACT USER QUANTITY TO FORM
015575          SBC          C.MARK+3,X          ; NEW POSITION. IF FINAL
015576          STA          >TPOSL- $\$$ FD,X          ; RESULT IS L.T. ZERO, THEN REPORT
015577          INY          ; POSITION ERROR...
015578          INX
015579          BNE          SUBPOSN
015580          BCS          ADJMRK1          ; BRANCH IF LEGAL POSITION CALCULATED.
015581  ERRPOSN          LDA          #POSNERR
015582          SEC          ; INDICATE ERROR
015583          RTS
015584  *
015585  ADJMRK          LDX          #2          ; FIRST SET UP POSITION TEMPS USED
015586  ADJMRK0          LDA          C.MARK,X          ; BY BOTH POSITION ROUTINES
015587          STA          TPOSL,X
015588          DEX
015589          BPL          ADJMRK0
015590  ADJMRK1          CLC          ; NO ERRORS
015591          RTS
015592  *
015593  *
015594  RDPOSN          EQU          *
015595  CKSAMBLK          EQU          *
015596          LDY          #FCBMARK+1          ; FIRST TEST TO SEE IF NEW POSITION IS
015597          LDA          (FCBPTR),Y          ; WITHIN THE SAME (CURRENT) DATA BLOCK.
015598          AND          # $\$$ FE
015599          STA          SCRTCH
015600          INY          ; BUMP TO ACCESS HIGHEST ORDER ADDRESS BYTE
015601          LDA          TPOSLH          ; GET MIDDLE BYTE OF NEW POSITION
015602          SEC
015603          SBC          SCRTCH
015604          STA          SCRTCH
015605          BCC          TYPMARK          ; BRANCH IF POSSIBLY L.T. CURRENT POSITION
015606          CMP          #2          ; MUST BE WITHIN 512 BYTES OF BEGINNING OF CURRENT
015607          BCS          TYPMARK
015608          LDA          TPOSHI          ; NOW MAKE SURE WERE TALKIN ABOUT
015609          CMP          (FCBPTR),Y          ; THE SAME 64K CHUNK!
015610          BNE          TYPMARK          ; BRANCH IF WE AREN'T.

```

```

015611          JMP          SVMARK          ; IF WE IS, ADJUST FCB AND POSPTR AND RETURN.
015612          *
015613  TYPMARK          LDY          #FCBSTYP          ; NOW FIND OUT WHICH TYPE
015614          LDA          (FCBPTR),Y          ; OF FILE WE'RE POSITIONING ON.
015615          BEQ          FERRTYP          ; THERE IS NO SUCH TYPE AS ZERO, BRANCH NEVER!
015616          CMP          #4          ; IS IT A TREE CLASS FILE?
015617          BCC          CHKDSKSW          ; YES, GO POSITION
015618          JMP          DIRMARK          ; NO, TEST FOR DIRECTORY TYPE.
015619          *
015620  CHKDSKSW          EQU          *          ; MAKE SURE S/HE HASN'T MOVED THE VOLUME
015621          LDY          #FCBDEVN
015622          LDA          (FCBPTR),Y
015623          STA          DEVNUM          ; MAKE SURE DEVICE NUMBER PARM IS CURRENT
015624          JSR          TWRPROT1          ; PASSES DEVNUM (CHECK DISK SWITCH)
015625          LDA          DSWGLOB          ; DISK SWITCH GLOBAL
015626          BEQ          TREPOS          ; BRANCH IF NONE DETECTED
015627  CHKDSKS1          JSR          VERFYVOL          ; MATCHES VCBPTR VS. DEVNUM
015628          BCC          TREPOS          ; BRANCH IF DISK HASN'T SWITCHED
015629          JSR          USRREQ          ; POLITELY ASK USER TO MOUNT
015630          BCC          CHKDSKS1          ; SAID HE DID, CHECK AGAIN
015631          LDA          #VNFERR          ; REFUSES TO MOUNT
015632          RTS
015633          *
015634  FERRTYP          LDY          #FCBREFN          ; CLEAR ILLEGALLY TYPED FCB ENTRY
015635          STA          (FCBPTR),Y
015636          LDA          #BADREFNUM          ; TELL EM THERE IS NO SUCH FILE
015637          SEC
015638          RTS
015639          *
015640          PAGE
015641  TREPOS          LDY          #FCBSTYP          ; USE STORAGE TYPE AS NUMBER
015642          LDA          (FCBPTR),Y          ; OF LEVELS (SINCE 1=SEED, 2=SAPLING, AND 3=TREE)
015643          STA          LEVELS
015644          LDY          #FCBSTAT          ; SINCE IT'S A DIFFERENT DATA
015645          LDA          (FCBPTR),Y          ; BLOCK, MUST NOT FORGET PREVIOUS DATA.
015646          AND          #DATMOD          ; THEREFORE, SEE IF PREVIOUS DATA WAS MODIFIED
015647          BEQ          POSNEW1          ; THEN DISK MUST BE UPDATED.
015648          JSR          WFCBDAT          ; GO WRITE CURRENT DATA BLOCK.
015649          BCS          POSERR          ; RETURN ANY ERROR ENCOUNTERED.
015650          *
015651  POSNEW1          LDY          #FCBMARK+2          ; TEST TO SEE IF CURRENT
015652          LDA          (FCBPTR),Y          ; INDEX BLOCK IS GOING TO BE USABLE...
015653          AND          #$FE          ; OR IN OTHER WORDS-
015654          STA          SCRTCH          ; IS NEW POSITION WITHIN 128K OF THE BEGINNING
015655          LDA          TPOSHI          ; OF CURRENT SAPLING LEVEL CHUNK.
015656          SEC
015657          SBC          SCRTCH
015658          BCC          POSNEW2          ; BRANCH IF A NEW INDEX BLOCK IS ALSO NEEDED
015659          CMP          #2          ; NEW POSITION IS > THAN BEGINING OF OLD. IS IT WITHIN 128K?
015660          BCS          POSNEW2          ; BRANCH IF NOT.

```

```

015661          LDX          LEVELS          ; IS THE FILE WE'RE DEALING WITH A SEED?
015662          DEX
015663          BNE          DATLEVEL        ; NO, USE CURRENT INDEXES.
015664 TSTINY    LDA          TPOSLH        ; IS NEW POSITION UNDER 512?
015665          LSR          A
015666          ORA          TPOSHI
015667          BNE          NOIDXDAT       ; NO, MARK BOTH DATA AND INDEX BLOCK AS UN-ALLOCATED.
015668          LDY          #FCBFRST
015669          LDA          (FCBPTR),Y     ; FIRST BLOCK IS ONLY BLOCK AND IT'S DATA!
015670          STA          BLOKNML
015671          INY
015672          LDA          (FCBPTR),Y     ; (HIGH BLOCK ADDRESS)
015673          JMP          RNEWPOS        ; GO READ IN BLOCK AND SET APPROPRIATE STATUSES.
015674 *
015675          PAGE
015676 POSNEW2   LDY          #FCBSTAT       ; GOT A CHECK TO SEE IF PREVIOUS
015677          LDA          (FCBPTR),Y     ; INDEX BLOCK WAS MODIFIED.
015678          AND          #IDXMOD
015679          BEQ          POSNIDX         ; READ IN OVER IT IF CURRENT IS UP TO DATE.
015680          JSR          WFCBIDX        ; GO UPDATE INDEX ON DISK (BLOCK ADDR IN FCB)
015681          BCS          POSERR
015682 POSNIDX   LDX          LEVELS          ; BEFORE READING IN TOP INDEX, CHECK TO BE SURE
015683          CPX          #3              ; THAT THERE IS A TOP INDEX...
015684          BEQ          POSINDEX       ; BRANCH IF FILE IS FULL BLOWN TREE.
015685          LDA          TPOSHI         ; IS NEW POSITION WITHIN RANGE OF A
015686          LSR          A              ; SAPLING FILE (L.T. 128K)?
015687          PHP
015688          LDA          #TOPALC+IDXALC+DATA LC ; (TO INDICATE NO LEVEL IS ALLOCATED FOR NEW POSITION.)
015689          PLP
015690          BNE          NODATA         ; GO MARK 'EM ALL DUMMY.
015691          JSR          CLRSTATS        ; GO CLEAR STATUS BITS 0,1,2 (INDEX/DATA ALLOC STATUS).
015692          DEX
015693          BEQ          TSTINY         ; (UNAFFECTED SINCE LOADED ABOVE) CHECK FOR SEED
015694          JSR          RFCBFST        ; IF SEED, CHECK FOR POSITION L.T. 512...
015695          BCS          POSERR        ; GO GET ONLY INDEX BLOCK
015696          LDY          #FCBIDXB      ; BRANCH IF ERROR
015697          LDA          BLOKNML
015698          STA          (FCBPTR),Y     ; SAVE NEWLY LOADED INDEX BLOCK'S ADDRESS
015699          INY
015700          LDA          BLOKNMH
015701          STA          (FCBPTR),Y
015702          BCC          DATLEVEL        ; BRANCH ALWAYS...
015703 POSERR    SEC
015704          RTS
015705 *
015706 POSINDEX   JSR          CLRSTATS        ; CLEAR ALL ALLOCATION REQUIREMENTS FOR PREVIOUS POSITION
015707          JSR          RFCBFST        ; GET HIGHEST LEVEL INDEX BLOCK.
015708          BCS          POSERR
015709          LDA          TPOSHI         ; THEN TEST FOR A SAP LEVEL INDEX BLOCK
015710          LSR          A

```

```

015711          TAY
015712          LDA      (TINDX),Y
015713          INC      TINDX+1
015714          CMP      (TINDX),Y          ; (BOTH HI AND LO WILL BE ZERO IF NO INDEX EXISTS)
015715          BNE      SAPLEVEL
015716          CMP      #0          ; ARE BOTH BYTES ZERO?
015717          BNE      SAPLEVEL
015718          DEC      TINDX+1          ; DON'T LEAVE WRONG POINTERS LAYING AROUND!
015719  NOIDXDAT  LDA      #IDXALC+DATALC ; SHOW NEITHER INDEX OR DATA BLOCK ALLOCATED.
015720          JMP      NODATA
015721  *
015722          PAGE
015723  SAPLEVEL  STA      BLOKNML          ; READ IN NEXT LOWER INDEX BLOCK
015724          LDA      (TINDX),Y          ; (HI ADDRESS)
015725          STA      BLOKNMH
015726          DEC      TINDX+1
015727          JSR      RFCBIDX          ; READ IN SAPLING LEVEL
015728          BCS      POSERR
015729  DATLEVEL  LDA      TPOSHI          ; NOW GET BLOCK ADDRESS OF DATA BLOCK
015730          LSR      A
015731          LDA      TPOSLH          ; ( IF THERE IS ONE )
015732          ROR      A
015733          TAY
015734          LDA      (TINDX),Y          ; DATA BLOCK ADDRESS LOW
015735          INC      TINDX+1
015736          CMP      (TINDX),Y
015737          BNE      POSNEW3
015738          CMP      #0
015739          BNE      POSNEW3
015740          LDA      #DATALC          ; SHOW DATA BLOCK AS NEVER BEEN ALLOCATED
015741          DEC      TINDX+1
015742  *
015743  NODATA    LDY      #FCBSTAT
015744          ORA      (FCBPTR),Y          ; SET STATUS TO SHOW WHATS MISSIN'
015745          STA      (FCBPTR),Y
015746          LSR      A          ; THROW AWAY BIT THAT SAYS DATA BLOCK UN-ALLOCATED
015747          LSR      A          ; CUZ WE KNOW THAT. CARRY NOW INDICATES IF INDEX BLOCK
015748          JSR      ZIPDATA          ; ALSO IS INVALID AND NEEDS TO BE ZEROED (CARRY UNDISTURBED)
015749          BCC      SVMARK          ; BRANCH IF INDEX BLOCK DOESN'T NEED ZIPPIN.
015750  ZIPIIDX   STA      (TINDX),Y
015751          INY
015752          BNE      ZIPIIDX
015753          INC      TINDX+1
015754  ZPIDX1    STA      (TINDX),Y
015755          INY
015756          BNE      ZPIDX1
015757          DEC      TINDX+1          ; RESTORE PROPER ADDRESS
015758          JMP      SVMARK
015759  *
015760  ZIPDATA   LDA      #0          ; ALSO IS INVALID AND NEEDS TO BE ZEROED.

```

```

015761          TAY
015762  ZIPDAT0  STA      (DATPTR),Y      ; ZERO OUT DATA AREA
015763          INY
015764          BNE      ZIPDAT0
015765          INC      DATPTR+1
015766  ZPDAT1  STA      (DATPTR),Y
015767          INY
015768          BNE      ZPDAT1
015769          DEC      DATPTR+1
015770          RTS
015771  *
015772          PAGE
015773  *
015774  POSNEW3  STA      BLOKNML      ; GET DATA BLOCK OF NEW POSITION
015775          LDA      (TINDX),Y      ; (HI ADDRESS)
015776          DEC      TINDX+1
015777  RNEWPOS  STA      BLOKNMH
015778          JSR      RFCBDAT
015779          BCS      PRITZ          ; RETURN ANY ERROR
015780          JSR      CLRSTATS      ; SHOW WHOLE CHAIN IS ALLOCATED
015781  SVMARK   LDY      #FCBMARK+2    ; UPDATE POSITION IN FILE CONTROL BLOCK
015782          LDX      #2
015783  SVMRK1   LDA      (FCBPTR),Y      ; REMEMBER OLDMARK IN CASE
015784          STA      OLDMARK-FCBMARK,Y ; CALLING ROUTINE FAILS LATER
015785          LDA      TPOSLH,X
015786          STA      (FCBPTR),Y
015787          DEY
015788          DEX          ; MOVE 3 BYTE POSITION MARKER
015789          BPL      SVMRK1
015790  *
015791          CLC          ; LAST, BUT NOT LEAST, SET UP
015792          LDA      DATPTR          ; INDIRECT ADDRESS TO BUFFER PAGE POINTED
015793          STA      POSPTR          ; TO BY THE CURRENT POSITION MARKER.
015794          LDA      TPOSLH
015795          AND      #1
015796          ADC      DATPTR+1
015797          STA      POSPTR+1
015798          LDA      SISDATP
015799          STA      SISPOSP      ; SISTER PAGE BYTE ALSO.
015800          RTS          ; CARRY SHOULD ALWAYS BE CLEAR
015801  PRITZ    SEC          ; RANDOM ERROR
015802          RTS          ; RETURN
015803  *
015804  *
015805  CLRSTATS  LDY      #FCBSTAT      ; CLEAR ALLOCATION STATES FOR DATA BLOCK
015806          LDA      (FCBPTR),Y      ; AND BOTH LEVELS OF INDEXES.
015807          AND      #$FF-TOPALC-IDXALC-DATALC
015808          STA      (FCBPTR),Y      ; THIS SAYS THAT EITHER THEY EXIST CURRENTLY
015809          RTS          ; OR THAT THEY'RE UNNECESSARY FOR CURRENT POSITION.
015810  *

```

```

015811          PAGE
015812  *
015813  DIRMARK    CMP      #DIRTYP      ; IS IT A DIRECTORY?
015814          BEQ      DIRPOS      ; YES...
015815          LDA      #CPTERR      ; NO, THERE IS A COMPATABILITY PROBLEM-
015816          JSR      SYSERR      ; THE DAMN THING SHOULD OF NEVER BEEN OPENED!
015817  *
015818  DIRPOS      LDA      SCRTCH      ; RECOVER RESULTS OF PREVIOUS SUBTRACTION.
015819          LSR      A              ; USE DIFFERENCE AS COUNTER AS TO HOW MANY
015820          STA      CNTENT      ; BLOCKS MUST BE READ TO GET TO NEW POSITION.
015821          LDY      #FCBMARK+1      ; TEST FOR POSITION DIRECTION.
015822          LDA      (FCBPTR),Y
015823          CMP      TPOSLH      ; CARRY INDICATES DIRECTION...
015824          BCC      DIRFWRD      ; IF SET, POSITION FORWARD.
015825  DIRVRSE    LDY      #0          ; OTHERWISE, READ DIRECTORY FILE IN REVERSE ORDER.
015826          JSR      DIRPOS1      ; READ PREVIOUS BLOCK.
015827          BCS      DRPOSERR      ; BRANCH IF ANYTHING GOES WRONG.
015828          INC      CNTENT      ; COUNT UP TO 128
015829          BPL      DIRVRSE      ; LOOP IF THERE IS MORE BLOCKS TO PASS OVER.
015830          BMI      SVMARK      ; BRANCH ALWAYS.
015831  *
015832  DIRFWRD      LDY      #2          ; POSITION IS FORWARD FROM CURRENT POSITION.
015833          JSR      DIRPOS1      ; READ NEXT DIRECTORY BLOCK.
015834          BCS      DRPOSERR
015835          DEC      CNTENT
015836          BNE      DIRFWRD      ; LOOP IF POSITION NOT FOUND IN THIS BLOCK.
015837          BEQ      SVMARK      ; BRANCH ALWAYS.
015838  *
015839  DIRPOS1      LDA      (DATPTR),Y      ; GET LINK ADDRESS OF PREVIOUS OR
015840          STA      BLOKNML      ; NEXT DIRECTORY BLOCK.
015841          INY              ; BUT FIRST BE SURE THERE IS A LINK.
015842          CMP      (DATPTR),Y
015843          BNE      DIRPOS2      ; BRANCH IF CERTAIN LINK EXISTS
015844          CMP      #0          ; ARE BOTHE LINK BYTES 0?
015845          BNE      DIRPOS2      ; NOPE, JUST HAPPEN TO BE THE SAME VALUE.
015846          LDA      #EOFERR      ; SOMETHING IS WRONG WITH THIS DIRECTORY FILE!
015847  DRPOSERR    SEC              ; INDICATE ERROR
015848          RTS
015849  *
015850  DIRPOS2      LDA      (DATPTR),Y      ; (HIGH ORDER BLOCK ADDRESS)
015851          STA      BLOKNMH
015852  * DROP INTO 'RFCBDAT' (READ FILE'S DATA BLOCK)
015853  *
015854  * NOTE: FOR DIRECTORY POSITIONING NO OPTIMIZATION HAS BEEN
015855  * DONE SINCE DIRECTORY FILES WILL ALMOST ALWAYS BE LESS
015856  * THAN 6 BLOCKS. IF MORE SPEED IS REQUIRED OR DIRECTORY
015857  * TYPE FILES ARE TO BE USED FOR OTHER PURPOSES REQUIRING
015858  * MORE BLOCKS, THEN THE RECOMMENDED METHOD IS TO CALL
015859  * 'RFCBDAT' FOR THE FIRST BLOCK AND GO DIRECTLY TO
015860  * DEVICE (VIA JMP (IUNITL)) HANDLER FOR SUBSEQUENT

```

```

015861 * ACCESSES.
015862 * ALSO NOTE THAT NO CHECKING IS DONE FOR READ/WRITE
015863 * ENABLE SINCE A DIRECTORY FILE CAN ONLY BE OPENED
015864 * FOR READ ACCESS.
015865 *
015866 PAGE
015867 *
015868 RFCBDAT LDA #RDCMD ; SET READ COMMAND.
015869 STA DHPCMD
015870 LDX #DATPTR ; USE X TO POINT AT ADDRESS OF DATA BUFFER
015871 JSR FILEIO1 ; GO DO FILE INPUT.
015872 LDY #FCBDATB ; SAVE BLOCK NUMBER JUST READ IN FCB.
015873 BCC FCBLOKNM ; BRANCH IF NO ERRORS HAPPENED.
015874 RTS ; RETURN ERROR
015875 *
015876 RFCBIDX LDA #RDCMD ; PREPARE TO READ IN INDEX BLOCK.
015877 STA DHPCMD
015878 LDX #TINDX ; POINT AT ADDRESS OF CURRENT INDEX BUFFER
015879 JSR FILEIO1 ; GO READ INDEX BLOCK.
015880 BCS RDFCBERR ; REPORT ERROR
015881 LDY #FCBIDX B ; SAVE BLOCK ADDRESS OF THIS INDEX IN FCB.
015882 FCBLOKNM LDA BLOKNML
015883 STA (FCBPTR),Y
015884 INY
015885 LDA BLOKNMH
015886 STA (FCBPTR),Y
015887 CLC
015888 RDFCBERR RTS
015889 *
015890 RFCBFST LDX #TINDX ; POINT AT ADDRESS OF INDEX BUFFER
015891 LDY #FCBFRST ; AND BLOCK ADDRESS OF FIRST FILE BLOCK IN FCB
015892 LDA #RDCMD ; AND LASTLY, MAKE IT A READ!
015893 * DROP INTO DOFILEIO
015894 *
015895 DOFILEIO STA DHPCMD ; SAVE COMMAND.
015896 LDA (FCBPTR),Y ; GET DISK BLOCK ADDRESS FROM FCB.
015897 STA BLOKNML
015898 INY ; BLOCK ZERO NOT LEGAL.
015899 CMP (FCBPTR),Y
015900 BNE FILEIO
015901 CMP #0 ; ARE BOTH BYTES ZERO?
015902 BNE FILEIO ; NO, CONTINUE WITH REQUEST.
015903 LDA #ALCERR ; OTHERWISE REPORT ALLOCATION ERROR.
015904 JSR SYSDEATH ; NEVER RETURNS...
015905 *
015906 PAGE
015907 FILEIO LDA (FCBPTR),Y ; GET HIGH ADDRESS OF DISK BLOCK
015908 STA BLOKNMH
015909 FILEIO1 LDA 0,X ; GET MEMORY ADDRESS OF BUFFER FROM
015910 STA DBUFPL ; S.O.S. ZERO PAGE POINTED TO BY

```

```

015911      JSR      WRAPADJ      ;GO ADJUST FOR BANK CROSSING <SRS 82.162>
015912      LDA      1,X
015913      STA      DBUFPH      ; SET HI BYTE
015914      LDA      SISTER+1,X  ; AND BANK PAIR BYTE. <SRS 82.162>
015915      STA      SISBPB
015916      LDY      #FCBDEVN
015917      LDA      (FCBPTR),Y  ; OF COURSE HAVING THE DEVICE NUMBER
015918      STA      DEVNUM      ; WOULD MAKE THE WHOLE OPERATION MORE MEANINGFUL...
015919  FILEIO2  LDA      #2      ; ALSO, SET UP BYTE COUNT TO 512 AND
015920      STA      RQCNTH     ; SET 'BYTES READ' POINTER TO
015921      STA      IOACCESS    ; (INTERUPT! SET TO INDICATE REG CALL MADE TO DEV HANDLER. RETURN INTERUPT!)
015922      LDA      #>TRASH    ; A PLACE TO THROW BYTES READ AWAY
015923      STA      BRDPTR
015924      LDA      #<TRASH    ; LOCALLY DEFINED
015925      STA      BRDPTR+1
015926      LDA      #0        ; SO THAT IT DOESN'T MESS UP ANY OTHER DATA.
015927      STA      RQCNTL
015928      STA      SSBDRPH     ; ('BYTES READ' IS THROWN AWAY)
015929  RPEATIO1 LDA      DEVNUM  ; TRANSFER THE DEVICE NUMBER FOR DISPATCHER TO CONVERT TO UNIT NUMBER.
015930      STA      UNITNUM
015931  RPEATIO0 LDY      #$9     ; PREPARE TO SAVE DEVICE PARMS
015932  SAVPRMS  LDA      DEVICE,Y ; MOVE FROM Z PAGE
015933      STA      RPTBLOK,Y  ; TO MY OWN SPACE
015934      DEY                ; FROM $C9 THROUGH $CO
015935      BPL      SAVPRMS
015936  DMGRGO   EQU      *        ; CALL EXTERNAL DEVICE MANAGER
015937      LDA      #0
015938      STA      SERR        ; CLEAR GLOBAL ERROR VALUE
015939      JSR      DMGR       ; CALL THE DRIVER
015940      BCC      RRITZ      ; RTS IF NO ERRORS
015941      CMP      #XDISKSW   ; DISKSWITCH ITERATES
015942      BEQ      RPEATIO2  ; BRANCH IF DISK SWITCH AND REPEAT I/O REQUEST
015943      SEC                ; REPORT ERROR
015944  RRITZ    RTS
015945  RPEATIO2 LDY      #$9     ; LENGTH OF PARM BLOCK
015946  GETPRMS  LDA      RPTBLOK,Y
015947      STA      DEVICE,Y  ; RESTORE POSSIBLY DISTURBED PARM BLOCK
015948      DEY
015949      BPL      GETPRMS
015950      JMP      DMGRGO    ; AND TRY THE I/O AGAIN
015951      *
015952      *
015953  TRASH     DS      2        ; ONLY USED TO PUT BYTES READ TO SLEEP
015954  RPTBLOK   DS      10     ; DMGR PARM SAVE BLOCK
015955      *
015956      *
015957  WFCBFST   LDY      #FCBDEVN ; FETCH THE
015958      LDA      (FCBPTR),Y  ; DEVICE NUMBER
015959      TAX                ; AND UPDATE
015960      JSR      UPBMAP     ; ITS BITMAP

```



```

015961          LDX      #TINDX          ; POINT AT ADDRESS OF INDEX BLOCK
015962          LDY      #FCBFRST       ; AND THE DISK ADDRESS OF FILE'S FIRST BLOCK IN FCB
015963          LDA      #WRTCMD        ; LASTLY, MAKE IT A WRITE REQUEST.
015964          JMP      DOFILEIO       ; AND GO DO IT!
015965          *
015966 WFCBDAT    LDX      #DATPTR
015967          LDY      #FCBDATB        ; POINT AT MEMORY ADDRESS WITH X AND DISK ADDRESS WITH Y.
015968          LDA      #WRTCMD        ; WRITE DATA BLOCK.
015969          JSR      DOFILEIO
015970          BCS      FILIOERR        ; REPORT ANY ERRORS
015971          LDA      #$FF-DATMOD      ; MARK DATA STATUS AS CURRENT.
015972          JMP      FCBUPDAT
015973          *
015974 WFCBIDX    LDY      #FCBDEVN      ; MAKE SURE
015975          LDA      (FCBPTR),Y        ; THE BITMAP
015976          TAX
015977          JSR      UPBMAP          ; FOR THIS DEVICE ("X")
015978          LDX      #TINDX          ; IS UPDATED
015979          LDY      #FCBIDXB        ; POINT AT ADDRESS OF INDEX BUFFER
015980          LDA      #WRTCMD        ; AND BLOCK ADDRESS OF THAT INDEX BLOCK.
015981          JSR      DOFILEIO        ; GO WRITE OUT INDEX BLOCK.
015982          BCS      FILIOERR        ; REPORT ANY ERRORS
015983          LDA      #$FF-IDXMOD      ; MARK INDEX STATUS AS CURRENT.
015984 FCBUPDAT    LDY      #FCBSTAT      ; CHANGE STATUS BYTE TO
015985          AND      (FCBPTR),Y        ; REFLECT SUCCESSFUL DISK FILE UPDATE.
015986          STA      (FCBPTR),Y      ; (CARRY IS UNAFFECTED)
015987 FILIOERR    RTS
015988          *
015989          *
015990          PAGE
015991 OPEN        JSR      FINDFILE      ; FIRST OF ALL LOOK UP THE FILE...
015992          BCC      OPEN0
015993          CMP      #BADPATH        ; IS AN ATTEMPT TO OPEN A ROOT DIRECTORY?
015994          BNE      ERROPN         ; NO, PASS BACK ERROR
015995          *
015996 OPEN0       JSR      TSTOPEN      ; FIND OUT IF ANY OTHER FILES ARE WRITING
015997          BCC      OPEN1         ; TO THIS SAME FILE. (BRANCH IF NOT)
015998 ERRBUSY     LDA      #FILBUSY     ; REPORT SHARED ACCESS NOT ALLOWED.
015999 ERROPN     SEC
016000          RTS                    ; RETURN ERROR.
016001          *
016002 OPEN1      LDA      DATPTR        ; GET ADDRESS OF FIRST FREE FCB FOUND
016003          STA      FCBPTR         ; DURING TEST OPEN SEQUENCE AND USE
016004          LDA      DATPTR+1        ; IT AS FILE CONTROL AREA. IF HIGH BYTE OF
016005          STA      FCBPTR+1        ; POINTER IS ZERO, THEN NO FCB
016006          BNE      ASGNFCB        ; IS AVAILABLE FOR USE.
016007          LDA      #FCBFULL       ; REPORT FCB FULL ERROR.
016008          SEC
016009          RTS
016010          *

```

```

016011 ASGNFCB      LDY      #$1F          ; ASSIGN FCB, BUT FIRST
016012             LDA      #0            ; CLEAN OUT ANY OLD RUBBISH LEFT AROUND...
016013 CLRFCB      STA      (FCBPTR),Y
016014             DEY
016015             BPL      CLRFCB
016016             LDY      #FCBENTIN     ; NOW BEGIN CLAIM BY MOVING IN FILE
016017 FCBOWNR     LDA      D.DEV-1,Y     ; OWNERSHIP INFORMATION.
016018             STA      (FCBPTR),Y     ; NOTE: THIS CODE DEPENDS UPON THE DEFINED
016019             DEY                      ; ORDER OF BOTH THE FCB AND DIRECTORY ENTRY
016020             BNE      FCBOWNR       ; BUFFER (D.). BEWARE OF CHANGES!!! *****
016021             LDA      DFIL+D.STOR    ; GET STORAGE TYPE.
016022             LSR      A              ; STRIP OFF FILE NAME LENGTH.
016023             LSR      A
016024             LSR      A              ; (BY DIVIDING BY 16)
016025             LSR      A
016026             TAX                      ; SAVE IN X FOR LATER TYPE COMPARISON
016027             LDY      #FCBSTYP
016028             STA      (FCBPTR),Y     ; SAVE STORAGE TYPE.
016029             LDA      C.OPLSTLN     ; IS THERE AN OPEN LIST?
016030             BEQ      DEFOPEN       ; NO, USE DEFAULT REQUEST ACCESS...
016031             LDY      #0            ; YES, FIND OUT WHAT ACCESS IS REQUESTED.
016032             LDA      (C.OPLIST),Y  ; IF REQ-ACCESS IS ZERO, THEN
016033             BEQ      DEFOPEN       ; USE DEFAULTS...
016034             AND      DFIL+D.ATTR   ; CHECK REQUEST AGAINST ATTRIBUTES.
016035             CMP      (C.OPLIST),Y  ; WERE ALL ACCESS REQUESTS SATISFIED?
016036             BEQ      SVATTRB      ; YES, SAVE ATTRIBUTES.
016037             LDA      #ACCERR      ; REPORT ACCESS REQUEST CAN'T BE MET.
016038             SEC
016039             RTS
016040             PAGE
016041 DEFOPEN      LDA      DFIL+D.ATTR   ; GET FILES ATTRIBUTES AND
016042             AND      #READEN+WRITEN ; USE IT AS A DEFAULT ACCESS REQUEST.
016043 SVATTRB      LDY      #FCBATTR
016044             CPX      #DIRTYP       ; IF DIRECTORY, DON'T ALLOW WRITE ENABLE
016045             BNE      SVATTR1
016046             AND      #READEN
016047 SVATTR1     STA      (FCBPTR),Y
016048             AND      #WRITEN       ; CHECK FOR WRITE ENABLED REQUESTED.
016049             BEQ      OPEN2         ; BRANCH IF READ ONLY OPEN.
016050             LDA      TOTENT        ; OTHERWISE, BE SURE NO ONE ELSE IS READING SAME
016051             BNE      ERRBUSY       ; FILE (SET UP BY TSTOPEN).
016052 OPEN2       LDA      DFIL+D.COMP   ; OH, BY THE WAY... IS THIS FILE
016053             BEQ      OPEN3         ; COMPATABLE WITH VERSION 0000? *****
016054 ERRCOMPAT    LDA      #CPTERR     ; REPORT FILE IS INCOMPATABLE!
016055             SEC
016056             RTS
016057 *
016058 OPEN3       CPX      #TRETYP+1     ; IS IT A TREE TYPE FILE?
016059             BCC      OPEN4         ; TEST FOR FURTHER COMPATABILITY. IT MUST
016060             CPX      #DIRTYP       ; BE EITHER A TREE OR A DIRECTORY.

```

```

016061          BNE      ERRCMPAT          ; REPORT INCOMPATABLE.
016062 OPEN4     LDY      #FCBFRST        ; MOVE ADDRESS OF FIRST BLOCK OF FILE
016063          LDA      DFIL+D.FRST      ; INTO FCB. NO CHECKING IS DONE FOR VALIDITY.
016064          STA      (FCBPTR),Y
016065          STA      BLOKNML
016066          INY
016067          LDA      DFIL+D.FRST+1
016068          STA      (FCBPTR),Y        ; NOTE: THE FCB HAS NOT BEEN OFFICIALLY
016069          STA      BLOKNMH          ; CLAIMED YET. TO DO THIS, THE FIRST BYTE
016070          LDY      #FCBEOF          ; MUST CONTAIN A VALID REFERENCE NUMBER.
016071 EOFCBMV   LDA      DFIL+D.EOF-FCBEOF,Y ; MOVE CURRENT END OF FILE
016072          STA      (FCBPTR),Y        ; TO FCB.
016073          INY
016074          CPY      #FCBEOF+3
016075          BNE      EOFCBMV
016076          LDA      DFIL+D.USAGE
016077          STA      (FCBPTR),Y        ; AND CURRENT BLOCK COUNT OF FILE.
016078          INY
016079          LDA      DFIL+D.USAGE+1
016080          STA      (FCBPTR),Y
016081          LDA      C.OPLSTLN        ; NOW THAT WE'VE COME THIS FAR, FIND
016082          BEQ      DEFBUFR          ; OUT WHICH TYPE OF BUFFER AND ALLOCATE IT!
016083          CMP      #1                ; WAS IT ONLY TO SET ATTRIBUTES?
016084          BEQ      DEFBUFR
016085          CMP      #4                ; IS A FULL ADDRESS INCLUDED?
016086          BEQ      UBUFSPEC
016087          LDA      #BADLSTCNT
016088          SEC
016089          RTS
016090          *
016091          PAGE
016092 UBUFSPEC   LDY      #1                ; (INDEX TO 'PAGECNT' OF OPEN LIST)
016093          LDA      (C.OPLIST),Y        ; IS USER SPECIFYING THE BUFFER?
016094          BEQ      DEFBUFR          ; NO, USE DEFAULT BUFFER (DYNAMIC)
016095          CPX      #TRETYP+1        ; IF TREE TYPE FILE, THEN AT LEAS 4 PAGES ARE NEEDED.
016096          BCC      ONEKTST         ; BRANCH IF TREE TYPE.
016097          CMP      #2                ; DID USER GIVE AT LEAST 2 PAGES FOR DIRECTORY TYPE?
016098          BCS      FIXDBUF         ; YES, LOG IT WITH BUFFER MANAGER
016099 ERRBTS    LDA      #BTSERR        ; REPORT NOT ENOUGH BUFFER SPACE.
016100          SEC
016101          RTS
016102          *
016103 ONEKTST    CMP      #4                ; IS THERE AT LEAST ONE KILOBYTE BUFFER FOR TREES?
016104          BCC      ERRBTS           ; NO, THEN TO HELL WITH IT!.
016105 FIXDBUF    JSR      REQFXBUF       ; CALL BOB AND ASK FOR HIM TO FIX IT...
016106          BCC      FCBUFFER        ; GO SAVE BUFFER NUMBER.
016107 ERROPN1    RTS                    ; RETURN ANY ERROR ENCOUNTERED.
016108          *
016109 DEFBUFR     LDA      #4                ; ASSUME TREE FILE (4 PAGES REQUIRED)
016110          CPX      #TRETYP+1

```

```

016111      BCC      BUFREQST      ; BRANCH IF IT IS A TREE.
016112      LDA      #2              ; OTHERWISE, WE JUST NEED TWO PAGES.
016113  BUFREQST  JSR      REQBUF      ; CALL BOB TO ALLOCATE A DYNAMIC BUFFER.
016114      BCS      ERROPEN1      ; REPORT ANY ERRORS.
016115  FCBUFFER  LDY      #FCBBUFN    ; SAVE BUFFER NUMBER AND THEN
016116      STA      (FCBPTR),Y      ; FIND OUT WHERE IT IS.
016117      JSR      GTBUFFRS      ; HAVE BOB RETURN ADDRESS IN DATA & INDEX POINTERS.
016118      BCS      ERROPEN2      ; IF ERROR, FREE BUFFER BEFOR RETURNING.
016119      LDY      #FCBREFN      ; NOW CLAIM FCB FOR THIS FILE.
016120      LDA      CNTENT        ; THIS WAS SET UP BY 'TSTOPEN'.....
016121      STA      (FCBPTR),Y
016122      LDY      #FCBLEVL        ; MARK LEVEL
016123      LDA      LEVEL          ; AT WHICH
016124      STA      (FCBPTR),Y      ; FILE WAS OPENED
016125      LDY      #FCBSTYP        ; GET STORAGE TYPE AGAIN.
016126      LDA      (FCBPTR),Y      ; FILE MUST BE POSITIONED TO BEGINNING.
016127      CMP      #TRETYP+1      ; IS IT A TREE FILE?
016128      BCS      OPNDIR        ; NO, ASSUME IT'S A DIRECTORY.
016129      LDA      #$FF          ; FOOL THE POSITION ROUTINE INTO GIVING
016130      LDY      #FCBMARK        ; A VALID POSITION WITH PRELOADED DATA, ETC.
016131  OPNPOS    STA      (FCBPTR),Y
016132      INY
016133      CPY      #FCBMARK+3
016134      BNE      OPNPOS
016135      LDY      #2              ; SET DESIRED POSITION TO ZERO.
016136      LDA      #0
016137  OPNPOS1   STA      TPOSLL,Y
016138      DEY
016139      BPL      OPNPOS1
016140      JSR      RDPOSN          ; LET TREE POSITION ROUTINE DO THE REST.
016141      BCC      OPENDONE      ; BRANCH IF SUCCESSFUL.
016142      *
016143      PAGE
016144  ERROPEN2   PHA              ; SAVE ERROR CODE.
016145      LDY      #FCBBUFN      ; SINCE ERROR WAS ENCOUNTERED BEFORE FILE
016146      LDA      (FCBPTR),Y      ; WAS SUCCESSFULLY OPENED, THEN
016147      JSR      RELBUF        ; IT'S NECESSARY TO FREE THE BUFFER AND
016148      LDY      #FCBREFN      ; FILE CONTROL BLOCK.
016149      LDA      #0
016150      STA      (FCBPTR),Y
016151      PLA
016152      SEC
016153      RTS
016154      *
016155  OPNDIR     JSR      RFCBDAT    ; READ IN FIRST BLOCK OF DIRECTORY FILE.
016156      BCS      ERROPEN2      ; RETURN ANY ERROR AFTER FREEING BUFFER & FCB
016157  OPENDONE   LDY      #VCBOPNC  ; INCREMENT OPEN COUNT FOR THIS
016158      LDA      (VCBPTR),Y      ; VOLUME. ALSO MARK STATUS.
016159      CLC
016160      ADC      #1

```

```

016161      STA      (VCBPTR),Y
016162      LDY      #VCBSTAT          ; HI BIT INDICATES VOLUME BUSY
016163      LDA      (VCBPTR),Y
016164      ORA      #$80
016165      STA      (VCBPTR),Y          ; DOESN'T MATTER HOW MANY, JUST BE SURE IT'S SET.
016166      LDY      #FCBREFN          ; PASS USER HIS REFERENCE NUMBER
016167      LDA      (FCBPTR),Y
016168      LDY      #0
016169      STA      (C.OUTREF),Y
016170      CLC
016171      RTS
016172      *
016173      PAGE
016174      *
016175      TSTOPEN  LDA      FCBADDRH          ; TEST FOR SHARED ACCESS FILES WITH WRITE ENABLED.
016176      STA      FCBPTR+1
016177      LDA      FCBANKNM
016178      STA      SISFCBP
016179      LDA      #0
016180      STA      DATPTR+1          ; MARK AS NO FREE FOUND.
016181      STA      CNTENT
016182      STA      TOTENT          ; ALSO, INIT COUNT OF MATCHING FILES
016183      TSTOPN1  STA      FCBPTR          ; SAVE NEW LOW ORDER ADDRESS
016184      LDX      DATPTR+1          ; FIND OUT IF A FREE SPOT HAS BEEN FOUND YET.
016185      BNE      TSTOPN2          ; YES, DON'T INCREMENT REFNUM (CNTENT).
016186      INC      CNTENT          ; BUMP REFNUM
016187      TSTOPN2  LDY      #FCBREFN          ; TEST FOR IN USE FCB
016188      LDA      (FCBPTR),Y          ; (NON ZERO)
016189      BNE      CHKACTV          ; THIS FCB IS IN USE, COPARE OWNERSHIP.
016190      TXA
016191      BNE      TSNXFCB          ; BRANCH IF A FREE SPOT HAS ALREADY BEEN FOUND.
016192      LDA      FCBPTR          ; TRANSFER CURRENT POINTER SO IT MAY BE
016193      STA      DATPTR          ; USED AS A FREE FCB BY OPEN.
016194      LDA      FCBPTR+1          ; HIGH BYTE ALWAYS NON ZERO.
016195      STA      DATPTR+1
016196      JMP      TSNXFCB
016197      *
016198      CHKACTV  EQU      *          ; IF MATCHING FILE IS SWAPPED, IT DOESNT COUNT
016199      LDY      #FCBSWAP
016200      LDA      (FCBPTR),Y
016201      BNE      TSNXFCB          ; BRANCH IF SWAPPED
016202      LDY      #FCBENTIN          ; NOTE: THIS CODE DEPENDS ON THE
016203      WHOWNS   LDA      (FCBPTR),Y          ; DEFINED ORDER OF FCB AND DIRECTORY
016204      CMP      D.DEV-1,Y          ; *****
016205      BNE      TSNXFCB          ; BRANCH IF THIS ONE HAS A DIFFERENT OWNER.
016206      DEY
016207      BNE      WHOWNS
016208      INC      TOTENT          ; REPORT THIS ONE AS A CO-OWNER.
016209      LDY      #FCBATTR          ; NOW FIND OUT IF THIS ONE WANTS TO WRITE.
016210      LDA      (FCBPTR),Y

```

```
016211      AND      #WRITEN      ; IF WRITE IS NOT ENABLED THEN CONTINUE.
016212      BEQ      TSNXFCB
016213      SEC
016214      RTS
016215      *
016216 TSNXFCB      LDA      FCBPTR      ; CALCULATE NEXT FCB AREA (+$20)
016217      CLC
016218      ADC      #$20
016219      BCC      TSTOPN1      ; LOOP IF NO PAGE CROSS.
016220      LDX      FCBPTR+1
016221      INC      FCBPTR+1
016222      CPX      FCBADDRH      ; HAVE WE LOOKED AT BOTH PAGES?
016223      BEQ      TSTOPN1      ; NOPE, LOOK AT PAGE TWO.
016224      CLC
016225      RTS
016226      *
016227      CHN      READ/WRITE,4,2
016228
016229 *****
016230 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: POSN.OPEN
016231 *****
016232
016233
016234
```

```

016235 =====
016236 DOCUMENT :SOS1.3.4of5.FOUR:SOS.READ.WRITE.TEXT
016237 =====
016238
016239 *****
016240 * APPLE /// SOS 1.3 SOURCE CODE FILE: READ.WRITE
016241 *****
016242 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
016243
016244 PAGE
016245 READ      CLC                ; FIRST DETERMINE IF REQUESTED
016246          LDY      #FCBATTR    ; READ IS LEGAL
016247          LDA      (FCBPTR),Y
016248          AND      #READEN     ; IS READ ENABLED?
016249          BNE     READ1       ; YES, CONTINUE...
016250          LDA      #ACCSERR    ; REPORT ILLEGAL ACCESS.
016251          SEC
016252          RTS
016253 *
016254 READ1     LDY      #FCBMARK    ; GET CURRENT MARK INTO 'TPOS' AND
016255          LDA      (FCBPTR),Y  ; DETERMINE IF RESULTING POSITION
016256          STA      TPOSLL     ; EXCEEDS CURRENT END OF FILE.
016257          ADC      C.BYTES
016258          STA      SCRATCH
016259          INY
016260          LDA      (FCBPTR),Y
016261          STA      TPOSLLH
016262          ADC      C.BYTES+1   ; (THIS WAS DONE STRAIT-LINE SINCE
016263          STA      SCRATCH+1  ; WE'RE ADDING A TWO BYTE TO A THREE
016264          INY                ; BYTE QUANTITY)
016265          LDA      (FCBPTR),Y
016266          STA      TPOSLLH
016267          ADC      #0         ; ADD IN REMAINING CARRY.
016268          STA      SCRATCH+2
016269          LDY      #FCBEOF+2  ; NOW TEST EOF AGAINST POSITION GENERATED
016270 EOFTEST   LDA      SCRATCH-FCBEOF,Y
016271          CMP      (FCBPTR),Y ; IS NEW POSITION > EOF?
016272          BCC     READ2       ; NO, PROCEED.
016273          BNE     ADJUSTCNT   ; YES, ADJUST 'C.BYTES' REQUEST
016274          DEY
016275          CPY      #FCBEOF-1  ; HAVE WE COMPARED ALL TREE BYTES?
016276          BNE     EOFTEST     ; NO, TEST NEXT LOWEST.
016277 ADJUSTCNT EQU      *        ; ADJUST REQUEST TO READ UP TO (BUT
016278          LDY      #FCBEOF    ; NOT INCLUDING) END OF FILE.
016279          LDA      (FCBPTR),Y ; RESULT= (EOF-1)-POSITION
016280          SBC      TPOSLL
016281          STA      C.BYTES
016282          INY
016283          LDA      (FCBPTR),Y

```

```

016284          SBC          TPOSLH
016285          STA          C.BYTES+1
016286          ORA          C.BYTES          ; IF BOTH BYTES ARE ZERO, REPORT EOF ERROR.
016287          BNE          READ2
016288          LDA          #EOFERR
016289          JSR          SYSERR
016290  READ2      LDA          C.BYTES
016291          STA          RWREQH
016292          BNE          READ3          ; BRANCH IF READ REQUEST DEFINITELY NON-ZERO.
016293          CMP          C.BYTES+1
016294          BNE          READ3          ; BRANCH IF READ REQUEST<>ZERO
016295          STA          RWREQH
016296  GORDDNE   JMP          READONE          ; DO NOTHING.
016297          PAGE
016298          *
016299  READ3      LDA          C.BYTES+1
016300          STA          RWREQH
016301          LDA          C.OUTBUF          ; MOVE POINTER TO USERS BUFFER TO BFM
016302          STA          USRBUF          ; Z-PAGE AREA.
016303          LDX          #C.OUTBUF          ; <SRS 82.162>
016304          JSR          WRAPADJ          ; ADJUST FOR BANK CROSSING. <SRS 82.162>
016305          STA          USRBUF+1
016306          STY          SISUSRBF          ; SAVE VALID USER BUFFER ADDRESS (THAT WILL NOT CROSS BANKS)
016307          LDY          #FCBSTYP          ; NOW FIND OUT IF IT'S A TREE READ OR OTHER.
016308          LDA          (FCBPTR),Y
016309          CMP          #TRETYP+1
016310          BCC          TREAD          ; BRANCH IF A TREE FILE.
016311          JMP          DREAD          ; OTHERWISE ASSUME IT'S A DIRECTORY.
016312          *
016313  TREAD      JSR          RDPOSN          ; GET DATA POINTER SET UP.
016314          BCC          TREAD0          ; REPORT ANY ERRORS
016315          JMP          ERRFIX1
016316  TREAD0     JSR          PREPRW          ; TEST FOR NEWLINE, SETS UP FOR PARTIAL READ.
016317          JSR          READPART          ; MOVE CURRENT DATA BUFFER CONTENTS TO USER AREA
016318          BVS          GORDDNE          ; BRANCH IF REQUEST IS SATISFIED.
016319          BCS          TREAD          ; CARRY SET INDICATES NEWLINE IS SET.
016320          LDA          RWREQH          ; FIND OUT HOW MANY BLOCKS ARE TO BE READ
016321          LSR          A          ; IF LESS THAN TWO, THEN DO IT THE SLOW WAY.
016322          BEQ          TREAD
016323          STA          BULKCNT          ; SAVE BULK BLOCK COUNT.
016324          LDY          #FCBSTAT          ; MAKE SURE CURRENT DATA AREA
016325          LDA          (FCBPTR),Y          ; DOESN'T NEED TO BE WRITTEN BEFORE
016326          AND          #DATMOD          ; RESETTING POINTER TO READ DIRECTLY INTO
016327          BNE          TREAD          ; USER'S AREA. BRANCH IF DATA NEED TO BE WRITTEN
016328          STA          IOACCESS          ; TO FORCE FIRST CALL THRU ALL DEVICE HANDLER CHECKING.
016329          LDA          USRBUF          ; MAKE THE DATA BUFFER THE USER'S SPACE.
016330          STA          DATPTR
016331          LDA          USRBUF+1
016332          STA          DATPTR+1
016333          LDA          SISUSRBF

```



```

016334          STA          SISDATP
016335  *
016336          PAGE
016337  RDFAST      JSR          RDPOSN          ; GET NEXT BLOCK DIRECTLY INTO USER SPACE.
016338          BCS          ERRFIX           ; BRANCH ON ANY ERROR.
016339  RDFAST0     INC          DATPTR+1       ; BUMP ALL POINTERS BY 512 (ONE BLOCK)
016340          INC          DATPTR+1
016341          DEC          RWREQH
016342          DEC          RWREQH
016343          INC          TPOSLH
016344          INC          TPOSLH
016345          BNE          RDFAST1           ; BRANCH IF POSITION DOES NOT GET TO A 64K BOUNDARY.
016346          INC          TPOSHI           ; OTHERWISE, MUST CHECK FOR A 128K BOUNDARY
016347          LDA          TPOSHI           ; SET CARRY IF MOD 128K HAS BEEN REACHED
016348          EOR          #1
016349          LSR          A
016350  RDFAST1     DEC          BULKCNT        ; HAVE WE READ ALL WE CAN FAST?
016351          BNE          RDFAST2           ; BRANCH IF MORE TO READ.
016352          JSR          FXDATPTR          ; GO FIX UP DATA POINTER TO SOS BUFFER.
016353          LDA          RWREQH           ; TEST FOR END OF READ.
016354          ORA          RWREQH           ; ARE BOTH ZERO?
016355          BEQ          READONE
016356          BNE          TREAD            ; NO, READ LAST PARTIAL BLOCK.
016357  *
016358  RDFAST2     BCS          RDFAST
016359          LDA          TPOSHI           ; GET INDEX TO NEXT BLOCK ADDRESS
016360          LSR          A
016361          LDA          TPOSLH
016362          ROR          A
016363          TAY
016364          LDA          (TINDX),Y          ; INDEX TO ADDRESS IS INT(POS/512)
016365          STA          BLOKNML           ; GET LOW ADDRESS
016366          INC          TINDX+1
016367          CMP          (TINDX),Y          ; ARE BOTH HI AND LOW ADDRESS THE SAME?
016368          BNE          REALRD            ; NO, IT'S A REAL BLOCK ADDRESS.
016369          CMP          #0                ; ARE BOTH BYTES ZERO?
016370          BNE          REALRD            ; NOPE -- MUST BE REAL DATA
016371          STA          IOACCESS           ; DON'T DO REPEATIO JUST AFTER SPARSE
016372          BEQ          NOSTUF             ; BRANCH ALWAYS (CARRY SET)
016373  REALRD      LDA          (TINDX),Y          ; GET HIGH ADDRESS BYTE
016374          CLC
016375  NOSTUF      DEC          TINDX+1
016376          BCS          RDFAST            ; BRANCH IF NO BLOCK TO READ
016377          STA          BLOKNMH
016378          LDA          IOACCESS           ; HAS FIRST CALL GONE TO DEVICE YET?
016379          BEQ          RDFAST            ; NOPE, GO THRU NORMAL ROUTE...
016380          LDA          DATPTR+1           ; RESET HI BUFFER ADDRESS FOR DEVICE HANDLER
016381          STA          DBUFPH
016382          JSR          REPEATIO
016383          BCC          RDFAST0           ; BRANCH IF NO ERRORS.

```

```

016384          PAGE
016385  ERRFIX   PHA          ; SAVE ERROR CODE
016386          JSR      FXDATPTR ; GO RESTORE DATA POINTERS, ETC...
016387          PLA
016388  ERRFIX1  PHA          ; SAVE ERROR CODE
016389          JSR      READONE  ; PASS BACK NUMBER OF BYTES ACTUALLY READ.
016390          PLA
016391          SEC          ; REPORT ERROR
016392          RTS
016393  *
016394  READONE  LDY      #0          ; RETURN TOTAL NUMBER OF BYTES ACTUALLY READ
016395          SEC          ; THIS IS DERIVED FROM C.BYTES-RWREQ
016396          LDA      C.BYTES
016397          SBC      RWREQ
016398          STA      (C.OUTCNT),Y
016399          INY
016400          LDA      C.BYTES+1
016401          SBC      RWREQ
016402          STA      (C.OUTCNT),Y
016403          JMP      RDPOSN      ; LEAVE WITH VALID POSITION IN FCB.
016404  *
016405  PREPRW   SEC          ; ADJUST POINTER TO USER'S BUFFER TO
016406          LDA      USRBUF      ; MAKE THE TRANSFER
016407          SBC      TPOSLL
016408          STA      USRBUF
016409          BCS      PREPRW1      ; BRANCH IF NO ADJUSTMENT TO HI ADDR. NEEDED.
016410          DEC      USRBUF+1    ; NOTE: SARA ALLOWS INDIRECT FROM $101 UP
016411  PREPRW1 LDY      #FCBATTR    ; AS LONG AS ACTUAL RESULTING ADDRESS IS >=$200
016412          LDA      (FCBPTR),Y ; TEST FOR NEW LINE ENABLED
016413          AND      #NLINEN    ; SET CARRY IF IT IS.
016414          CLC
016415          BEQ      NONEWLN     ; BRANCH IF NEWLINE IS NOT ENABLED
016416          SEC
016417          LDY      #FCBNEWL
016418          LDA      (FCBPTR),Y ; MOVE NEWLINE CHARACTER TO MORE
016419          STA      NLCHAR      ; ACCESSABLE SPOT.
016420  NONEWLN LDY      TPOSLL    ; GET INDEX TO FIRST DATA
016421          LDA      DATPTR      ; RESET LOW ORDER OF POSPTR TO BEGINNING OF PAGE.
016422          STA      POSPTR
016423          LDX      RWREQ
016424          RTS          ; AND LASTLY GET LOW ORDER COUNT OF REQUESTED BYTES.
016425  *          ; RETURN STATUSES...
016426  READPART TXA
016427          BNE      RDPART0    ; BRANCH IF REQUEST IS NOT A EVEN PAGES
016428          LDA      RWREQ
016429          BEQ      SETRDNE    ; A CALL OF ZERO BYTES SHOULD NEVER GET HERE!
016430          DEC      RWREQ
016431  RDPART0   DEX
016432  RDPART   LDA      (POSPTR),Y ; MOVE DATA TO USER'S BUFFER
016433          STA      (USRBUF),Y ; ONE BYTE AT A TIME.

```

```

016434          TXA                      ; NOTE: THIS ROUTINE IS CODED TO BE
016435          BEQ          ENDRQCHK    ; FASTEST WHEN NEWLINE IS DISABLED.
016436 RDPART1   BCS          TSTNEWL   ; BRANCH IF NEW LINE NEEDS TO BE TESTED.
016437 RDPART2   DEX                      ;
016438          INY                      ; PAGE CROSSED?
016439          BNE          RDPART     ; NO. MOVE NEXT BYTE.
016440          LDA          POSPTR+1   ; TEST FOR END OF BUFFER
016441          INC          USRBUF+1   ; BUT FIRST ADJUST USER BUFFER POINTER
016442          INC          TPOSLH     ; AND POSITION.
016443          BNE          RDPART3
016444          INC          TPOSHI
016445 RDPART3   INC          POSPTR+1   ; AND SOS BUFFER HIGH ADDRESS.
016446          EOR          DATPTR+1   ; (CARRY HAS BEEN CLEVERLY UNDISTURBED.)
016447          BEQ          RDPART     ; BRANCH IF MORE TO READ IN BUFFER.
016448          CLV                      ; INDICATE NOT FINISHED.
016449          BVC          RDPRTDNE   ; BRANCH ALWAYS.
016450          *
016451 ENDRQCHK  LDA          RWREQH
016452          BEQ          RDRQDNE     ; BRANCH IF REQUEST SATISFIED.
016453          INY                      ; DONE WITH THIS BLOCK OF DATA?
016454          BNE          ENDRCHK1    ; NO, ADJUST HIGH BYTE OF REQUEST.
016455          LDA          POSPTR+1   ; MAYBE- CHECK FOR END OF BLOCK BUFFER.
016456          EOR          DATPTR+1   ; (DON'T DISTURB CARRY)
016457          BNE          ENDRCHK2   ; BRANCH IF HI COUNT CAN BE DEALT WITH NEXT TIME.
016458 ENDRCHK1 DEC          RWREQH
016459 ENDRCHK2 DEY                      ; RESTORE PROPER VALUE TO 'Y'
016460          JMP          RDPART1
016461          *
016462 TSTNEWL   LDA          (POSPTR),Y ; GET LAST BYTE TRANSFERED AGAIN.
016463          EOR          NLCHAR       ; HAVE WE MATCHED NEWLINE CHARACTER?
016464          BNE          RDPART2     ; NO, READ NEXT.
016465 RDRQDNE   INY                      ; ADJUST POSITION.
016466          BNE          SETRDNE
016467          INC          USRBUF+1    ; BUMP POINTERS.
016468          INC          TPOSLH
016469          BNE          SETRDNE
016470          INC          TPOSHI
016471 SETRDNE   BIT          SETVFLG   ; (SET V FLAG)
016472 RDPRTDNE  STY          TPOSLL     ; SAVE LOW POSITION
016473          BVS          RDONE1
016474          INX                      ; LEAVE REQUEST AS +1 FOR NEXT CALL
016475 RDONE1    STX          RWREQL     ; AND REMAINDER OF REQUEST COUNT.
016476          PHP                      ; SAVE STATUSES
016477          CLC                      ; ADJUST USER'S LOW BUFFER ADDRESS
016478          TYA
016479          ADC          USRBUF
016480          STA          USRBUF
016481          BCC          RDPART4
016482          INC          USRBUF+1    ; ADJUST HI ADDRESS AS NEEDED.
016483 RDPART4   PLP                      ; RESTORE RETURN STATUSES

```

```

016484 SETVFLG      RTS                ; (THIS BYTE <$60> IS USED TO SET V FLAG)
016485 *
016486 FXDATPTR     LDA      DATPTR      ; PUT CURRENT USER BUFFER
016487             STA      USRBUF      ; ADDRESS BACK TO NORMAL
016488             LDA      DATPTR+1
016489             STA      USRBUF+1    ; BANK PAIR BYTE SHOULD BE MOVED ALSO.
016490             LDA      SISDATP
016491             STA      SISUSRBF
016492             LDY      #FCBBUFN    ; RESTORE BUFFER ADDRESS
016493             LDA      (FCBPTR),Y
016494             LDX      #DATPTR
016495             JMP      GETBUFADR    ; END VIA CALL TO BOB'S CODE.
016496 *
016497             PAGE
016498 *
016499 * READ DIRECTORY FILE...
016500 *
016501 DREAD         JSR      RDPOSN
016502             BCS      ERRDRD      ; PASS BACK ANY ERRORS
016503             JSR      PREPRW      ; PREPARE FOR TRANSFER.
016504             JSR      READPART    ; MOVE DATA TO USER'S BUFFER
016505             BVC      DREAD       ; REPEAT UNTIL REQUEST IS SATISFIED.
016506             JSR      READONE     ; UPDATE FCB AS TO NEW POSITION.
016507             BCC      DREDONE    ; BRANCH IF ALL IS WELL.
016508             CMP      #EOFERR     ; WAS LAST READ TO END OF FILE?
016509             SEC
016510             BNE      DREDEERR    ; BRANCH IF NOT EOF ERROR.
016511             JSR      SVMARK
016512             JSR      ZIPDATA     ; CLEAR OUT DATA BLOCK.
016513             LDY      #FCBDATB+1  ; PROVIDE DUMMY BACK POINTER FOR FUTURE RE-POSITION
016514             LDA      (FCBPTR),Y  ; GET HI BYTE OF LAST BLOCK.
016515             PHA
016516             DEY
016517             LDA      (FCBPTR),Y  ; AND LOW BYTE.
016518             PHA
016519             LDA      #0          ; NOW MARK CURRENT BLOCK AS IMPOSSIBLE.
016520             STA      (FCBPTR),Y
016521             INY
016522             STA      (FCBPTR),Y
016523             TAY                ; NOW MOVE LAST BLOCK ADDRESS TO DATA BUFFER AS BACK POINTER.
016524             PLA
016525             STA      (DATPTR),Y
016526             PLA
016527             INY
016528             STA      (DATPTR),Y
016529 DREDONE      CLC                ; INDICATE NO ERROR
016530 DREDEERR     RTS
016531 *
016532 ERRDRD        JMP      ERRFIX1   ; REPORT HOW MUCH WE COULD TRANSFER BEFORE ERROR.
016533 *

```

```

016534          PAGE
016535 WRITE    CLC          ; FIRST DETERMINE IF REQUESTED
016536          LDY          #FCBATTR      ; WRITE IS LEGAL
016537          LDA          (FCBPTR),Y
016538          AND          #WRITEN      ; IS WRITE ENABLED?
016539          BNE          WRITE1        ; YES, CONTINUE...
016540 ERRACCS   LDA          #ACCSEERR    ; REPORT ILLEGAL ACCESS.
016541          SEC
016542 WPEROR    RTS
016543 *
016544 WRITE1    JSR          TSTWPROT     ; OTHERWISE, MAKE SURE DEVICE IS NOT WRITE PROTECTED.
016545          BCS          WPEROR        ; REPORT WRITE PROTECTED AND ABORT OPERATION.
016546 *
016547          LDY          #FCBMARK      ; GET CURRENT MARK INTO 'TPOS' AND
016548          LDA          (FCBPTR),Y     ; DETERMINE IF RESULTING POSITION
016549          STA          TPOSLL        ; EXCEEDS CURRENT END OF FILE.
016550          ADC          C.BYTES
016551          STA          SCRATCH
016552          INY
016553          LDA          (FCBPTR),Y
016554          STA          TPOSLL
016555          ADC          C.BYTES+1      ; (THIS WAS DONE STRAIGHT-LINE SINCE
016556          STA          SCRATCH+1     ; WE'RE ADDING A TWO BYTE TO A THREE
016557          INY                      ; BYTE QUANTITY)
016558          LDA          (FCBPTR),Y
016559          STA          TPOSHI
016560          ADC          #0            ; ADD IN REMAINING CARRY.
016561          STA          SCRATCH+2
016562          LDY          #FCBEOF+2     ; NOW TEST EOF AGAINST POSITION GENERATED
016563 WEOFTEST   LDA          SCRATCH-FCBEOF,Y
016564          CMP          (FCBPTR),Y   ; IS NEW POSITION > EOF?
016565          BCC          WRITE2        ; NO, PROCEED.
016566          BNE          WADJEOF      ; YES, ADJUST END OF FILE
016567          DEY
016568          CPY          #FCBEOF-1    ; HAVE WE COMPARED ALL TREE BYTES?
016569          BNE          WEOFTEST     ; NO, TEST NEXT LOWEST.
016570 WADJEOF   CLC
016571          LDY          #FCBEOF      ; ADJUST REQUEST TO WRITE UP TO (BUT
016572 WRTADJEOF  LDA          (FCBPTR),Y  ; NOT INCLUDING) END OF FILE.
016573          STA          OLDEOF-FCBEOF,Y ; SAVE OLD EOF IN CASE OF LATER ERROR
016574          LDA          SCRATCH-FCBEOF,Y ; RESULT=EOF
016575 *
016576          STA          (FCBPTR),Y
016577          INY
016578          CPY          #FCBEOF+3
016579          BNE          WRTADJEOF
016580 WRITE2    LDA          C.BYTES
016581          STA          RWREQ
016582          BNE          WRITE3        ; BRANCH IF WRITE REQUEST DEFINITELY NON-ZERO.
016583          CMP          C.BYTES+1

```

```

016584          BNE      WRITE3          ; BRANCH IF WRITE REQUEST<>ZERO
016585          STA      RWREQH
016586          JMP      WRITDONE        ; DO NOTHING.
016587 *
016588          PAGE
016589 WRITE3     LDA      C.BYTES+1
016590          STA      RWREQH
016591          LDA      C.OUTBUF          ; MOVE POINTER TO USERS BUFFER TO BFM
016592          STA      USRBUF          ; Z-PAGE AREA.
016593          LDA      C.OUTBUF+1
016594          STA      USRBUF+1        ; (SO IT MAY BE ADJUSTED WITHOUT LOOSING
016595          LDA      SISOUTBF          ; ORIGINAL ADDRESS.)
016596          STA      SISUSRBF
016597          LDY      #FCBSTYP          ; NOW FIND OUT IF IT'S A TREE WRITE OR OTHER.
016598          LDA      (FCBPTR),Y
016599          CMP      #TRETYP+1
016600          BCC      TWRITE           ; BRANCH IF A TREE FILE.
016601          JMP      ERRACCS          ; OTHERWISE RETURN AN ACCESS ERROR!
016602 TWRITE     JSR      RDPOSN          ; READ BLOCK WE'RE
016603          BCS      WRITERROR
016604          LDY      #FCBSTAT
016605          LDA      (FCBPTR),Y
016606          AND      #DATALC+IDXALC+TOPALC
016607          BEQ      TREWRT1
016608          LDY      #0                ; FIND OUT IF ENOUGH DISK SPACE IS AVAILABLE FOR
016609 TWRTALC   INY                    ; INDEXES AND DATA BLOCK
016610          LSR      A
016611          BNE      TWRTALC
016612          STY      REQL
016613          STA      REQH
016614          JSR      TFRBLK
016615          BCS      WRITERROR        ; PASS BACK ANY ERRORS.
016616          LDY      #FCBSTAT
016617          LDA      (FCBPTR),Y        ; NOW GET MORE SPECIFIC.
016618          AND      #TOPALC          ; ARE WE LACKING A TREE TOP?
016619          BEQ      TSTSAPWR        ; NO, TEST FOR LACK OF SAPLING LEVEL INDEX.
016620          JSR      TOPDOWN          ; GO ALLOCATE TREE TOP AND ADJUST FILE TYPE.
016621          BCC      DBLOKALC        ; CONTINUE WITH ALLOCATION OF DATA BLOCK.
016622 WRITERROR  PHA                    ; SAVE ERROR
016623          LDY      #FCBEOF
016624 WRITERR01   LDA      OLDEOF-FCBEOF,Y
016625          STA      (FCBPTR),Y        ; RESTORE OLD EOF UPON ERR
016626          INY
016627          CPY      #FCBEOF+3
016628          BNE      WRITERR01
016629          LDY      #FCBMARK
016630 WRITERR02   LDA      OLDMARK-FCBMARK,Y
016631          STA      (FCBPTR),Y        ; AND RESTORE OLD MARK!
016632          INY
016633          CPY      #FCBMARK+3

```

```

016634          BNE          WRITERR02
016635          PLA
016636          SEC
016637          RTS                ; ERROR RETURN
016638          *
016639  TWRITEGO          BVC          TWRITE                ; A PIGGY-BACK BACKWARD BRANCH
016640          *
016641          PAGE
016642  TSTSAPWR          LDA          (FCBPTR),Y                ; GET STATUS BYTE AGAIN.
016643          AND          #IDXALC                ; DO WE NEED A SAPLING LEVEL INDEX BLOCK?
016644          BEQ          DBLOKALC                ; NO, ASSUME IT'S JUST A DATA BLOCK NEEDED.
016645          JSR          SAPDOWN                ; GO ALLOCATE AN INDEX BLOCK AND UPDATE TREE TOP.
016646          BCS          WRITERROR                ; RETURN ANY ERRORS.
016647  DBLOKALC          JSR          ALCWBLK                ; GO ALLOCATE FOR DATA BLOCK.
016648          BCS          WRITERROR
016649          LDA          TPOSHI                ; CALCULATE POSITION WITHIN INDEX BLOCK.
016650          LSR          A
016651          LDA          TPOSLH
016652          ROR          A
016653          TAY                ; NOW PUT BLOCK ADDRESS INTO INDEX BLOCK
016654          INC          TINDX+1                ; HIGH BYTE FIRST.
016655          LDA          SCRTCH+1
016656          TAX
016657          STA          (TINDX),Y
016658          DEC          TINDX+1                ; (RESTORE POINTER TO LOWER PAGE OF INDEX BLOCK)
016659          LDA          SCRTCH                ; GET LOW BLOCK ADDRESS
016660          STA          (TINDX),Y                ; NOW STORE LOW ADDRESS.
016661          LDY          #FCBDATB                ; ALSO UPDATE FILE CONTROL BLOCK TO INDICATE
016662          STA          (FCBPTR),Y                ; THAT THIS BLOCK IS ALLOCATED.
016663          INY
016664          TXA                ; GET HIGH ADDRESS AGAIN.
016665          STA          (FCBPTR),Y
016666          LDY          #FCBSTAT
016667          LDA          (FCBPTR),Y
016668          ORA          #IDXMOD
016669          AND          #$FFF-DATALC-IDXALC-TOPALC ; CLEAR ALLOCATION REQUIREMENT BITS.
016670          STA          (FCBPTR),Y
016671  TREWRT1          LDX          #USRBUF                ; LOCATE POINTER TO ADJUST <SRS 82.162>
016672          JSR          WRAPADJ                ; ADJUST FOR BANK CROSSING <SRS 82.162>
016673          JSR          PREPRW                ; WRITE ON
016674          JSR          WRTPART
016675          BVC          TWRITEGO
016676  WRITDONE          JMP          RDPOSN                ; UPDATE FCB WITH NEW POSITION.
016677          *
016678          PAGE
016679  WRTPART          TXA
016680          BNE          WRPART                ; BRANCH IF REQUEST IS NOT A EVEN PAGES
016681          LDA          RWREQH                ; A CALL OF ZERO BYTES SHOULD NEVER GET HERE!
016682          BEQ          SETWRDNE                ; DO NOTHING!
016683          *

```

```

016684          DEC          RWREQH
016685  WRPART          DEX
016686          LDA          (USRBUF),Y          ; MOVE DATA FROM USER'S BUFFER
016687          STA          (POSPTR),Y          ; ONE BYTE AT A TIME.
016688          TXA
016689          BEQ          ENDWQCHK
016690  WRPART2          INY          ; PAGE CROSSED?
016691          BNE          WRPART          ; NO. MOVE NEXT BYTE.
016692          LDA          POSPTR+1          ; TEST FOR END OF BUFFER
016693          INC          USRBUF+1          ; BUT FIRST ADJUST USER BUFFER POINTER
016694          INC          TPOSLH          ; AND POSITION.
016695          BNE          WRPART3
016696          INC          TPOSHI
016697  WRPART3          INC          POSPTR+1          ; AND SOS BUFFER HIGH ADDRESS.
016698          EOR          DATPTR+1          ; (CARRY HAS BEEN CLEVERLY UNDISTURBED.)
016699          BEQ          WRPART          ; BRANCH IF MORE TO WRITE TO BUFFER.
016700          CLV
016701          BVC          WRPRTDNE          ; INDICATE NOT FINISHED.
016702          *
016703  ENDWQCHK          LDA          RWREQH          ; BRANCH ALWAYS.
016704          BEQ          WRTRQDNE
016705          INY          ; BRANCH IF REQUEST SATISFIED.
016706          BNE          ENDWCHK1          ; ARE WE DONE WITH THIS BLOCK OF DATA?
016707          LDA          POSPTR+1          ; BRANCH IF NOT.
016708          EOR          DATPTR+1          ; WHILE THIS IS REDUNDANT, IT'S NECESSARY FOR
016709          BNE          ENDWCHK2          ; PROPER ADJUSTMENT OF REQUEST COUNT.
016710  ENDWCHK1          DEC          RWREQH          ; (NOT FINISHED- OK TO ADJUST HI BYTE.)
016711  ENDWCHK2          DEY          ; RESET MODIFIED Y
016712          JMP          WRPART2
016713          *
016714  WRTRQDNE          INY          ; AND POSITION.
016715          BNE          SETWRDNE
016716          INC          USRBUF+1          ; BUMP POINTERS.
016717          INC          TPOSLH
016718          BNE          SETWRDNE
016719          INC          TPOSHI
016720  SETWRDNE          BIT          SETVFLG          ; (SET V FLAG)
016721  WRPRTDNE          STY          TPOSLH          ; SAVE LOW POSITION
016722          STX          RWREQH          ; AND REMAINDER OF REQUEST COUNT.
016723          PHP          ; SAVE STATUSES
016724          LDY          #FCBSTAT
016725          LDA          (FCBPTR),Y
016726          ORA          #DATMOD+USEMOD
016727          STA          (FCBPTR),Y
016728          CLC          ; ADJUST USER'S LOW BUFFER ADDRESS
016729          LDA          TPOSLH
016730          ADC          USRBUF
016731          STA          USRBUF
016732          BCC          WRPART4
016733          INC          USRBUF+1          ; ADJUST HI ADDRESS AS NEEDED.

```



```

016734 WRPART4      JSR      FCBUSED      ; SET DIRECTORY FLUSH BIT
016735              PLP              ; RESTORE RETURN STATUSES
016736              RTS
016737              PAGE
016738 TOPDOWN      JSR      SWAPDOWN      ; FIRST MAKE CURRENT 1ST BLOCK AN ENTRY IN NEW TOP.
016739              BCS      TPDWNERR      ; RETURN ANY ERRORS
016740              LDY      #FCBSTYP      ; FIND OUT IF STORAGE TYPE HAS BEEN CHANGED TO 'TREE'.
016741              LDA      (FCBPTR),Y    ; (IF NOT, ASSUME IT WAS ORIGINALLY A SEED AND
016742              CMP      #TRETYP      ; BOTH LEVELS NEED TO BE BUILT.
016743              BEQ      TOPDWN1      ; OTHERWISE, ONLY AN INDEX NEED BE ALLOCATED)
016744              JSR      SWAPDOWN      ; MAKE PREVIOUS SWAP A SAP LEVEL INDEX BLOCK.
016745              BCS      TPDWNERR
016746 TOPDWN1      JSR      ALCWBLK      ; GET ANOTHER BLOCK ADDRESS FOR THE SAP LEVEL INDEX.
016747              BCS      TPDWNERR
016748              LDA      TPOSHI      ; CALCULATE POSITION OF NEW INDEX BLOCK
016749              LSR      A              ; IN THE TOP OF THE TREE.
016750              TAY
016751              LDA      SCRTCH      ; GET ADDRESS OF NEWLY ALOCATED INDEX BLOCK AGAIN
016752              TAX
016753              STA      (TINDX),Y
016754              INC      TINDX+1
016755              LDA      SCRTCH+1
016756              STA      (TINDX),Y    ; SAVE HI ADDRESS
016757              DEC      TINDX+1
016758              LDY      #FCBIDX+1    ; MAKE NEWLY ALLOCATED BLOCK THE CURRENT INDEX BLOCK.
016759              STA      (FCBPTR),Y
016760              TXA
016761              DEY
016762              STA      (FCBPTR),Y
016763              JSR      WFCBFST      ; SAVE NEW TOP OF TREE.
016764              BCS      TPDWNERR
016765              JMP      ZTMPIDX      ; END BY RE-CLEARING CURRENT (NEW) INDEX BLOCK.
016766 *
016767 SAPDOWN      LDY      #FCBSTYP      ; FIND OUT IF WE'RE DEALING WITH A TREE
016768              LDA      (FCBPTR),Y    ; OR A SIMPLE SEED.
016769              CMP      #SEEDTYP      ; IF SEED THEN AN ADJUSTMENT TO FILE TYPE IS NECESSARY.
016770              BEQ      SAPDWN1      ; BRANCH IF SEED.
016771              JSR      RFCBFST      ; OTHERWISE READ IN TOP OF TREE.
016772              BCC      TOPDWN1      ; BRANCH IF NO ERROR.
016773 TPDWNERR     RTS              ; RETURN ERRORS
016774 *
016775              PAGE
016776 SAPDWN1      EQU      *              ; MAKE CURRENT SEED INTO A SAPLING
016777 *
016778 SWAPDOWN      JSR      ALCWBLK      ; ALLOCATE A BLOCK BEFORE SWAP
016779              BCS      SWAPER      ; RETURN ERRORS IMMEDIATELY.
016780              LDY      #FCBFRST      ; GET PREVIOUS FIRST BLOCK
016781              LDA      (FCBPTR),Y    ; ADDRESS INTO INDEX BLOCK.
016782              PHA              ; SAVE TEMPORARLY WHILE SWAPPING IN NEW TOP INDEX
016783              LDA      SCRTCH      ; GET NEW BLOCK ADDRESS (LOW)

```

```

016784          TAX
016785          STA      (FCBPTR),Y
016786          INY
016787          LDA      (FCBPTR),Y
016788          PHA
016789          LDA      SCRTCH+1          ; AND HIGH ADDRESS TOO.
016790          STA      (FCBPTR),Y
016791          LDY      #FCBIDX+1          ; MAKE NEW TOP ALSO THE CURRENT INDEX IN MEMORY.
016792          STA      (FCBPTR),Y
016793          TXA          ; GET LOW ADDRESS AGAIN
016794          DEY
016795          STA      (FCBPTR),Y
016796          LDY      #0          ; MAKE PREVIOUS THE FIRST ENTRY IN SUB INDEX
016797          INC      TINDX+1
016798          PLA
016799          STA      (TINDX),Y
016800          DEC      TINDX+1
016801          PLA
016802          STA      (TINDX),Y
016803          JSR      WFCBFST          ; SAVE NEW FILE TOP.
016804          BCS      SWAPERR
016805          LDY      #FCBSTYP          ; NOW ADJUST STORAGE TYPE
016806          LDA      #1          ; BY ADDING 1 (THUS SEED BECOMES SAPLING BECOMES TREE)
016807          ADC      (FCBPTR),Y
016808          STA      (FCBPTR),Y
016809          LDY      #FCBSTAT
016810          LDA      (FCBPTR),Y          ; MARK STORAGE TYPE MODIFIED.
016811          ORA      #STPMOD
016812          STA      (FCBPTR),Y
016813          CLC          ; RETURN 'NO ERROR' STATUS.
016814  SWAPERR  RTS
016815  *
016816          PAGE
016817  ALCWBLK  JSR      ALC1BLK
016818          BCS      ALUSERR
016819          LDY      #FCBUSE
016820          LDA      (FCBPTR),Y          ; BUMP CURRENT USAGE COUNT BY 1.
016821          CLC
016822          ADC      #1
016823          STA      (FCBPTR),Y
016824          BCC      INCUSG1
016825          INY
016826          LDA      (FCBPTR),Y
016827          ADC      #0
016828          STA      (FCBPTR),Y
016829  INCUSG1  LDY      #FCBSTAT          ; MARK USAGE AS MODIFIED.
016830          LDA      (FCBPTR),Y
016831          ORA      #USEMOD
016832          STA      (FCBPTR),Y
016833          CLC          ; INDICATE NO ERROR

```

```

016834 ALUSERR      RTS                ; ALL DONE
016835 *
016836 TSTWPROT     LDY          #FCBSTAT      ; CHECK FOR A 'NEVER BEEN MODIFIED' CONDITION
016837             LDA          (FCBPTR),Y      ; GET STATUS BYTE
016838             AND          #USEMOD+DATMOD+IDXM0D+EOFMOD
016839             CLC                ; ANTICIPATE WRITE OK
016840             BNE          ALUSERR         ; ORDINARY RTS
016841             LDY          #FCBDEVN       ; GET FILE'S DEVICE NUMBER
016842             LDA          (FCBPTR),Y
016843             STA          DEVNUM        ; GET CURRENT STATUS OF BLOCK DEVICE
016844 TWRPROT1     LDA          #STATCMD
016845             STA          DHPCMD
016846             LDA          #STATSUB     ; STORE SUB COMMAND OF STATUS CALL
016847             STA          DSTATREQ
016848             LDA          #>TWRCODE
016849             STA          DSTATBFL     ; FETCH RETURN CODE IN SCRATCH AREA
016850             LDA          #<TWRCODE
016851             STA          DSTATBFH
016852             LDA          #0           ; MAKE SURE REGULAR RAM IS SELECTED (NO BANKS)
016853             STA          SISDSTAT
016854             STA          SERR         ; CLEAR GLOBAL ERROR FLAG
016855             LDA          DEVNUM       ; SET UP LAST PARM
016856             STA          UNITNUM      ; FOR DEVICE CALL
016857             JSR          DMGR        ; MAKE THE EXTERNAL CALL
016858             BCS          WPROTRET    ; RETURN ANY SPECIFIC ERRORS
016859             LDA          TWRCODE     ; GET STATUS BYTE
016860             LSR          A           ; SHIFT WRITE PROTECT STATE INTO CARRY
016861             LSR          A
016862             LDA          #XNOWRITE    ; ANTICIPATE WRITE PROTECTED.
016863             RTS                ; CARRY IS INDETERMINATE
016864 WPROTRET    EQU          *
016865             CMP          #XDISKSW     ; IF EXPLICITLY DISK SWITCH
016866             BNE          WPROT1     ; BRANCH IF XNODRIVE OR XNOWRITE
016867             STA          DSWGLOB     ; IF DISKSW, FLAG UNTIL ENTIRE OPERATION IS COMPLETE
016868             CLC
016869             RTS                ; DISKSWITCH DOESNT SET CARRY
016870 WPROT1     SEC
016871             RTS
016872 DSWGLOB      DS          1           ; DISK SWITCH GLOBAL
016873 TWRCODE      DS          1           ; A RARE EMBEDDED TEMP STORE
016874 *
016875             PAGE
016876 *
016877 * MEMORY 'WRAP-AROUND' ADJUST ROUTINE. THIS ROUTINE ADJUSTS
016878 * ADDRESSES THAT CROSS BANK PAIR BOUNDARIES. ON ENTRY, X CONTAINS
016879 * THE OFFSET OF THE ZERO PAGE EXTENDED POINTER TO BE ADJUSTED.
016880 * ON EXIT, THE POINTER WILL HAVE BEEN ADJUSTED, IF NECESSARY,
016881 * AND THE ASSOCIATED X-BYTE WILL ALSO HAVE BEEN ADJUSTED.
016882 * ONLY ADDRESSES IN THE RANGE $8200-$8E00 WILL BE ADJUSTED.
016883 *

```

```

016884 * UPON EXIT, A CONTAINS HIGH BYTE OF ADDRESS & Y CONTAINS UPDATED X-BYTE.
016885 * THIS ROUTINE LEAVES X UNCHANGED.
016886 *
016887 WRAPADJ      LDA      1,X          ; GET HIGH ADDRESS BYTE <SRS 82.162>
016888           LDY      SISTER+1,X      ; CHECK X-BYTE <SRS 82.162>
016889           BPL      WRAPDNE         ; NOT AN EXTENDED ADDRESS. <SRS 82.162>
016890           CMP      #$82           ; DOES IT NEED UPDATING? <SRS 82.162>
016891           BCC      WRAPDNE         ; NO <SRS 82.162>
016892           CPY      #$8F           ; SPECIAL BANK? <SRS 82.162>
016893           BCS      WRAPDNE         ; NO <SRS 82.162>
016894           AND      #$7F           ; ADJUST THE ADDRESS <SRS 82.162>
016895           STA      1,X          ; UPDATE <SRS 82.162>
016896           INC      SISTER+1,X     ; INCREMENT X-BYTE <SRS 82.162>
016897           INY      Y             ; UPDATE Y ALSO <SRS 82.162>
016898 *
016899 WRAPDNE      RTS                    ; RETURN VALID HIGH ADDRESS AND BANK BYTE.
016900
016901           CHN      CLOSE/EOF,4,2
016902
016903 *****
016904 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: READ.WRITE
016905 *****
016906
016907
016908

```

```

016909 =====
016910 DOCUMENT :SOS1.3.4of5.FOUR:SOS.SWAPOUT.IN.TEXT
016911 =====
016912
016913 *****
016914 * APPLE /// SOS 1.3 SOURCE CODE FILE: SWAPOUT.IN
016915 *****
016916 * ASSEMBLER: APPLE |[ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
016917
016918 SWAPOUT      EQU      *
016919 *
016920 * SWAP OUT A VOLUME LOGGED ON A DEVICE
016921 * INPUT ARGUMENT: DEVICE NUMBER "A"
016922 * (STORED AS "DEVNUM")
016923 * OUTPUT ARGUMENT: NONE
016924 * CONDITION CODE: CARRY SET USER DID NOT COMPLY WITH REQUEST
016925 *
016926 * SAVE VCBPTR, FCBPTR, DEVNUM ON STACK
016927 * 1) FIND UNSWAPPED VOLUME IN VCB
016928 * 2) IF DIRTY BIT MAP FOR THIS VOLUME THEN DO
016929 *   IF NOT ONLINE, REQUEST USER TO INSERT
016930 *   IF REQUEST DENIED, UNCONDITIONALLY CLOSE ALL FILES ON THIS VOLUME AND RTS
016931 *   IF ONLINE, UPDATE AND RELEASE BIT MAP
016932 *   DOEND
016933 * 3) SWAP IT (MARK VCBSWAP FIELD $80, MARK ALL FILES ON THIS VOLUME WITH SWAP MARK $8X WHERE X=VCB ENTRY)
016934 *   "VCB ENTRY" DEFINED AS: HIGH ORDER NIBBLE OF LOW ORDER BYTE OF ENTRIES VCB ADDRESS
016935 *   RESTORE VCBPTR, FCBPTR
016936 * RTS
016937 *
016938 TAX                      ; SAVE DEVICE NUMBER
016939 JSR      SAVECBS
016940 STX      DEVNUM          ; PERMANENTLY
016941 SWAPOUTX JSR      DEVVCB  ; FIND MATCHING UNSWAPPED ACTIVE VCB ENTRY (BY DEVNUM)
016942 BCS      SORTS          ; NO FIND--RETURN WITHOUT ERROR
016943 LDY      #VCBSTAT
016944 LDA      (VCBPTR),Y     ; GET STATUS OF FILES ON THIS VOLUME
016945 BPL      UNLOG          ; IF NO OPEN FILES, JUST THROW VOLUME AWAY
016946 LDA      DEVNUM          ; DIRTY BM EXIST ON THIS VOLUME?
016947 LDX      #0
016948 CMP      BMADEV,X       ; IN BIT MAP "A"?
016949 BEQ      FDIRBM        ; BRANCH IF YES
016950 LDX      #6             ; BIT MAP HEADER TABLE SIZE
016951 CMP      BMADEV,X       ; IN BIT MAP "B"?
016952 BEQ      FDIRBM        ; BRANCH IF YES
016953 JMP      MARKSWAP      ; NO NEED TO WRITE BIT MAP
016954 FDIRBM  LDA      BMASTAT,X ; IS BIT MAP DIRTY?
016955 BPL      MARKSWAP      ; BRANCH IF NOT
016956 GETVOL  JSR      VERFVOL  ; IS THE CORRECT VOLUME ON LINE NOW?
016957 BCC      VONLINE       ; BRANCH IF YES

```

```

016958      JSR      USRREQ      ; OTHERWISE, REQUEST USER INSERTION
016959      BCC      GETVOL      ; AND VERIFY IT AGAIN
016960      JSR      CLOSEU      ; USER SAID "NO": UNCONDITIONALLY CLOSE VOLUME
016961      JSR      RESTCBS
016962      SEC
016963      RTS                      ; ERROR RETURN TO CALLER
016964  VONLINE  LDX      DEVNUM    ; UPDATE THE
016965      JSR      UPBMAP      ; DIRTY BIT MAP
016966  MARKSWAP LDA      VCBPTR      ; CALCULATE
016967      LSR      A          ; SWAP BYTE
016968      LSR      A          ; AND
016969      LSR      A          ; MARK ALL FILES
016970      LSR      A          ; BELONGING TO THIS VOLUME
016971      SEC                      ; AS SWAPPED OUT
016972      ORA      #$80
016973      PHA                      ; SAVE SWAP BYTE
016974      JSR      FCBSCAN
016975      PLA                      ; MARK VCBSWAP
016976      LDY      #VCBSWAP      ; BYTE
016977      STA      (VCBPTR),Y
016978  SORTS   JSR      RESTCBS      ; RESTORE FCBPTR, VCBPTR, DEVNUM
016979      CLC
016980      RTS                      ; SUCCESSFUL SWAP OUT
016981  UNLOG   LDA      #0
016982      STA      VCB,X        ; UNLOG VOLUME
016983      BEQ      SORTS      ; SWAP THE EASY WAY! (BRANCH ALWAYS)
016984      *
016985      *
016986      *
016987  SWAPIN  EQU      *
016988      *
016989      * UNSWAP A VOLUME AND ALL ITS FILES
016990      *
016991      * INPUT ARGUMENT: VOLUME NAME (VCBPTR)
016992      * OUTPUT ARGUMENT: NONE
016993      * CONDITION CODE: CARRY SET : USER DID NOT COMPLY WITH REQUEST
016994      *
016995      * SAVE VCBPTR, FCBPTR ON STACK
016996      * 1) FIND SWAPPED VOLUME IN VCB, IF NOT FOUND, THEN RTS.
016997      * 2) IF ANOTHER UNSWAPPED VOLUME ON DEVICE, THEN SWAP IT
016998      * 3) VERIFY UNSWAPPED VOLUME, IF NOT OK THEN REQUEST INSERTION
016999      * 4) UNMARK VCB'S AND FCB'S
017000      * RTS
017001      JSR      SAVECBS      ; SAVE FCB, VCB POINTERS, DEVNUM
017002      LDY      #VCBNML      ; MAKE SURE VOLUME
017003      LDA      (VCBPTR),Y    ; IS AT LEAST OPEN
017004      BEQ      USRTS      ; BRANCH IF NOT RIGHT BACK TO CALLER
017005      LDY      #VCBSWAP      ; SEE IF
017006      LDA      (VCBPTR),Y    ; CURRENTLY SWAPPED
017007      BEQ      USRTS      ; IF NOT, RETURN IMMEDIATELY TO CALLER

```

```

017008          LDY          #VCBDEV          ; SAVE DEVICE NUMBER
017009          LDA          (VCBPTR),Y
017010          STA          DEVNUM
017011          PHA          ; SAVE DEVNUM AGAIN (SWAPOUTX TRASHES DEVNUM ON RETURN)
017012          JSR          SWAPOUTX        ; AND MAKE SURE ANY CURRENT ACTIVE VOLUME IS SWAPPED OUT (NOTICE ENTRY POINT)
017013          PLA          ; RECALL CURRENT DEVICE NUMBER
017014          STA          DEVNUM        ; AND SAVE IT TO ITS PROPER PLACE
017015  SI1       JSR          VERIFYVOL     ; VERIFY THE CURRENT VOLUME MOUNTED
017016          BCC          UNMARK        ; IF THE RIGHT ONE, GO MARK IT AS UNSWAPPED
017017          JSR          USRREQ        ; ELSE REQUEST USER TO INSERT
017018          BCC          SI1           ; USER SAID 'OK'
017019          JSR          CLOSEU       ; OTHERWISE UNCONDITIONALLY CLOSE
017020          JSR          RESTCBS
017021          SEC
017022          RTS
017023  UNMARK    LDY          #VCBSWAP     ; FETCH
017024          LDA          (VCBPTR),Y    ; VOLUME
017025          PHA          ; SWAP BYTE
017026          LDA          #0           ; BUT CLEAR
017027          STA          (VCBPTR),Y    ; VOLUME SWAP
017028          PLA
017029          CLC          ; "UNSWAPPED"
017030          JSR          FCBSCAN
017031          LDA          DEVNUM        ; MAKE SURE BIT MAPS
017032          JSR          CLEARBMS     ; ARE MARKED AS INVALID ON THIS DEVICE
017033  USRTS    JSR          RESTCBS     ; RESTORE VCB, FCB PTRS
017034          CLC          ; NO ERRORS
017035          RTS
017036          *
017037  SAVEPTRS  DS          5           ; A RARE EMBEDDED TEMP SAVE AREA, USED ONLY BY ...
017038          *
017039          *
017040  SAVECBS   EQU          *           ; SAVE FCBPTR, VCBPTR IN A TEMP SAVE AREA
017041          LDA          VCBPTR
017042          STA          SAVEPTRS
017043          LDA          VCBPTR+1
017044          STA          SAVEPTRS+1
017045          LDA          FCBPTR
017046          STA          SAVEPTRS+2
017047          LDA          FCBPTR+1
017048          STA          SAVEPTRS+3
017049          LDA          DEVNUM
017050          STA          SAVEPTRS+4
017051          RTS
017052          *
017053  RESTCBS   EQU          *           ; RESTORE FCBPTR, VCBPTR
017054          * NOTICE THERE EXISTS A SEQUENCE OF CALLS (SWAPIN, WHICH MAY CALL SWAPOUT) THAT JSR'S TO SAVECBS ONCE BUT JSR'S RESTCBS
TWICE.
017055          LDA          SAVEPTRS
017056          STA          VCBPTR

```

```

017057          LDA      SAVEPTRS+1
017058          STA      VCBPTR+1
017059          LDA      SAVEPTRS+2
017060          STA      FCBPTR
017061          LDA      SAVEPTRS+3
017062          STA      FCBPTR+1
017063          LDA      SAVEPTRS+4
017064          STA      DEVNUM
017065          RTS
017066 *
017067 *
017068 * MARK ALL FILES BELONGING TO A VOLUME
017069 * AS SWAPPED-IN OR SWAPPED-OUT.
017070 *
017071 * INPUT ARGS: DEVNUM -- DEVICE NUMBER OF MOUNTED VOLUME
017072 *          A REGISTER - SWAP BYTE
017073 *          CARRY -- CARRY FLAG SET MEANS SWAP OUT; ELSE SWAP IN
017074 *
017075 * OUTPUT ARGS: NONE
017076 * GLOBALS AFFECTED: FCB, FCBPTR
017077 * REGISTER STATUS: SCRAMBLED
017078 *
017079 FCBSCAN      EQU      *          ; MARK FILES BELONGING TO VOLUME AS SWAPPED OR UNSWAPPED
017080 *
017081          TAX          ; SAVE SWAP BYTE
017082          LDY          FCBADDRH   ; POINT TO
017083          STY          FCBPTR+1   ; BEGINNING TO FCB
017084          LDY          #0
017085          STY          FCBPTR
017086          BCS          FCBOUT     ; SWAP OUT A VOLUMES FILES
017087 FCBIN        EQU      *          ; SWAPIN A VOLUMES FILES
017088          JSR          FCBFETCH    ; GET NEXT ACTIVE FCB CANDIDATE
017089          BCS          FCBRTS     ; NO MORE FILES TO PROCESS
017090          LDY          #FCBSWAP
017091          TXA
017092          CMP          (FCBPTR),Y  ; SWAP BYTES MATCH?
017093          BNE          FCBIN1     ; BRANCH IF NOT
017094          LDA          #0
017095          STA          (FCBPTR),Y  ; MARK FILE AS SWAPPED IN
017096 FCBIN1       JSR          NEXTFCB ; ADVANCE FCB POINTER
017097          BCS          FCBRTS     ; NO MORE TO LOOK AT
017098          JMP          FCBIN      ; AND LOOK AT NEXT FILE
017099 *
017100 FCBOUT        EQU      *          ; SWAPPED OUT A VOLUMES FILES
017101          JSR          FCBFETCH    ; GET NEXT ACTIVE FILE IN FCB
017102          BCS          FCBRTS     ; NO MORE FILES -- RETURN TO USER
017103          LDY          #FCBSWAP   ; COMPARE
017104          LDA          (FCBPTR),Y
017105          BNE          FCBOUT1     ; ALREADY SWAPPED OUT
017106          TXA

```



```

017107          STA      (FCBPTR),Y          ; MARK AS SWAPPED
017108 FCBOUT1     JSR      NEXTFCB          ; ADVANCE FCB POINTER
017109          BCS      FCBRTS
017110          JMP      FCBOUT          ; SWAP OUT NEXT FILE
017111 *
017112 FCBRTS       RTS
017113 FCBFETCH     EQU      *          ; GET NEXT ACTIVE FILE FROM FCB
017114 * X REGISTER MUST NOT BE DISTURBED
017115 * USES FCBPTR
017116          LDY      #FCBDEVN          ; MAKE
017117          LDA      (FCBPTR),Y          ; SURE DEVICE
017118          CMP      DEVNUM          ; MATCHES
017119          BNE      NEXTFCB
017120          LDY      #FCBREFN          ; MAKE SURE FILE IS ACTIVE
017121          LDA      (FCBPTR),Y
017122          BEQ      NEXTFCB          ; BRANCH IF NOT
017123          CLC
017124          RTS          ; RETURN WITH CARRY CLEAR SHOWING AN ACTIVE FILE
017125 NEXTFCB     LDA      FCBPTR
017126          CLC
017127          ADC      #$20          ; FCB ENTRY SIZE
017128          STA      FCBPTR
017129          BCC      FCBFETCH          ; BRANCH IF NO PAGE CROSS
017130          LDA      FCBPTR+1
017131          INC      FCBPTR+1          ; SECOND PAGE
017132          CMP      FCBADDRH
017133          BEQ      FCBFETCH          ; LOOK AT PAGE TWO
017134 NEXTEND     SEC
017135          RTS          ; SHOW NO MORE FILES TO LOOK AT
017136 USRREQ       EQU      *          ; OPERATOR CONSOLE MESSAGE INTERFACE
017137 * PRODUCES A MESSAGE REQUESTING
017138 * THE SYSTEM OPERATOR TO MOUNT THE VOLUME
017139 * SPECIFIED BY "VCBPTR" ON DEVICE SPECIFIED
017140 * BY DEVNUM. THIS MODULE INSISTS
017141 * UPON THE CORRECT OPERATOR ACTION
017142 * UPON THREE FAILURES TO COMPLY,
017143 * THE MODULE WILL SIGNIFY FAILURE WITH
017144 * CARRY SET. IF THE CORRECT ACTION IS TAKEN,
017145 * CARRY WILL BE RETURNED CLEAR
017146 *
017147 * INPUT ARGS: VOLUME NAME (VCBPTR)
017148 *             DEVICE NUMBER (DEVNUM)
017149 *
017150 * OUTPUT ARGS: CC = OPERATOR COMPLIED WITH REQUESTED ACTION
017151 *             CS = OPERATOR COULDN'T/DIDN'T COMPLY
017152 *
017153 * GLOBALS AFFECTED: NONE
017154 *
017155 * STATUS OF REGISTERS: UNCERTAIN
017156 *

```

```

017157 VNML      EQU      ZPGTEMP      ; VOLUME NAME LENGTH
017158          LDY      #VCBNML      ; IF ILLEGAL VCB
017159          LDA      (VCBPTR),Y    ; GET OUT QUICK
017160          BEQ      NEXTEND      ; BRANCH TO SEC RTS
017161          LDX      #$E          ; LENGTH OF NAMED AREA-1
017162          LDA      #$0          ; NULLS
017163 UR1       STA      MDEV,X      ; BOTH CLEAR
017164          STA      MVOL,X      ; IN ONE LOOP
017165          DEX
017166          BPL      UR1
017167 *
017168 * DO A D-INFO TO FETCH THE DEVICE NAME
017169 *
017170          LDA      #5          ; DO ALL
017171          STA      $C0          ; NECESSARY
017172          LDA      DEVNUM      ; HOUSKEEPING
017173          STA      $C1          ; TO SET UP
017174          LDA      #>MDEV-1      ; A DEVICE MANAGER CALL
017175          STA      $C2
017176          LDA      #<MDEV-1
017177          STA      $C3
017178          LDA      #$8F          ; EXTEND BYTE
017179          STA      $14C3
017180          LDA      #0
017181          STA      $14C2
017182          STA      $C4
017183          STA      $C5
017184          STA      $C6          ; ZERO SUPERFLUOUS PARMS
017185          STA      URDERR      ; RESET FAILURE COUNT
017186          JSR      RPEATIO0     ; GET INFO FROM BOBS CODE
017187          LDA      #$20          ; "SPACE" RESTORED
017188          STA      MDEV-1      ; RESTORED
017189          LDY      #VCBNML
017190          LDA      (VCBPTR),Y    ; LENGTH OF VOLUME NAME
017191          STA      VNML        ; SAVED FOR WORK
017192          LDA      #0
017193          TAX
017194          LDY      #VCBNAM      ; POINT TO BEGINNING OF VOLUME NAME
017195 UR2       LDA      (VCBPTR),Y
017196          STA      MVOL,X
017197          INX
017198          INY          ; VOLUME NAME MOVED
017199          DEC      VNML        ; TO MESSAGE BUFFER
017200          BNE      UR2        ; CHARACTER BY CHARACTER
017201 URDU      LDX      #>UMB      ; PASS THE AREA'S ADDR
017202          LDY      #<UMB      ; IN X AND Y REGS,LOW, HIGH)
017203          JSR      OPMSGRPLY     ; HAVE MESSAGE SYSTEM PRINT IT
017204          JSR      VERFYVOL     ; DID THE USER COMPLY?
017205          BCS      URDU1       ; BRANCH IF NOT
017206          RTS          ; EXIT--CARRY IS CLEAR

```

```

017207 URDU1      INC      URDERR      ; COLLECT USER ERRORS
017208          LDA      URDERR
017209          CMP      #3              ; ONLY THREE TRIES ALLOWED
017210          BCC      URDU          ; RETRY MESSAGE IF LESS THAN THREE TRIES
017211          RTS
017212 *
017213 *
017214 *
017215 *
017216 *
017217 * CLOSE UNCONDITIONAL
017218 *
017219 * (USER HAS REPLIED 'N' TO A VOLUME MOUNT REQUEST
017220 * CLOSE ALL FILES ON VOLUME/UNLOG VOLUME
017221 *
017222 * INPUT ARGUMENT: (VCBPTR)
017223 * OUTPUT ARGUMENT: NONE
017224 *
017225 CLOSEU      EQU      *
017226 VSWA        EQU      ZPGTEMP      ; THE 'SWAP BYTE' STORED HERE
017227          LDY      #VCBDEV        ; FETCH
017228          LDA      (VCBPTR),Y      ; THE DEVICE NUMBER
017229          STA      DEVNUM          ; OF THIS VOLUME & SAVE IT
017230          LDY      #VCBSWAP        ; FETCH THE
017231          LDA      (VCBPTR),Y      ; SWAP BYTE
017232          STA      VSWA            ; SAVE FOR REFERENCE, TOO
017233          LDA      #0
017234          LDY      #VCBNML          ; UNLOG THE VOLUME
017235          STA      (VCBPTR),Y      ; BY SETTING LEN OF VOL NAME TO ZERO
017236          LDY      #VCBSWAP
017237          STA      (VCBPTR),Y      ; TURN OFF SWAP FLAG
017238          LDY      FCBADDRH        ; SET UP FCB SCAN FROM BEGINNING OF FCB
017239          STY      FCBPTR+1
017240          LDY      #0
017241          STY      FCBPTR
017242 VFCBLOP    LDY      #FCBDEVN      ; FETCH
017243          LDA      (FCBPTR),Y      ; THE DEVICE
017244          CMP      DEVNUM          ; NUMBER AND SEE IF A MATCH
017245          BNE      VFCBNXT        ; BRANCH IF NO MATCH
017246          LDY      #FCBREFN      ; SEE EVEN IF FILE OPEN
017247          LDA      (FCBPTR),Y
017248          BEQ      VFCBNXT        ; BRANCH IF NOT
017249          LDY      #FCBSWAP        ; CHECK TO SEE IF ATTACHED
017250          LDA      (FCBPTR),Y      ; TO SAME VOLUME
017251          CMP      VSWA
017252          BNE      VFCBNXT        ; BRANCH IF NOT
017253          LDY      #FCBBUFN        ; RELEASE
017254          LDA      (FCBPTR),Y      ; ANY
017255          JSR      RELBUF          ; BUFFERS ASSOCIATED
017256          LDY      #FCBSWAP        ; AND CLEAR

```

```

017257          LDA      #0              ; THE SWAP BYTE
017258          STA      (FCBPTR),Y
017259          LDY      #FCBREFN        ; AND FINALLY
017260          STA      (FCBPTR),Y      ; SAY 'CLOSED'
017261 VFCBNXT   LDA      FCBPTR
017262          CLC
017263          ADC      #$20             ; FCB ENTRY SIZE
017264          STA      FCBPTR
017265          BCC      VFCBLOP
017266          LDA      FCBPTR+1
017267          INC      FCBPTR+1        ; LOOK AT SECOND PAGE
017268          CMP      FCBADDRH
017269          BEQ      VFCBLOP        ; CHECK PAGE TWO OF FCB
017270          RTS
017271 *
017272 FCBUSED    EQU      *            ; MARK AS FCB AS DIRTY SO
017273 *                                THE DIRECTORY WILL BE FLUSHED ON 'FLUSH'
017274          STY      ZPGTEMP
017275          PHA
017276          LDY      #FCBDIRTY
017277          LDA      (FCBPTR),Y      ; FETCH CURRENT FCBDIRTY BYTE
017278          ORA      #FCBMOD        ; MARK FCB AS DIRTY
017279          STA      (FCBPTR),Y      ; SAVE IT BACK
017280          PLA
017281          LDY      ZPGTEMP        ; AND RESTORE REGS
017282          RTS
017283 *
017284 URDERR      DS      1              ; ERROR COUNT FOR USRREQ
017285 *
017286 *
017287 UMB         EQU      *
017288          DFB      $49,$6E,$73,$65,$72,$74,$20
017289          DFB      $76,$6F,$6C,$75,$6D,$65
017290          DFB      $3A,$20          ; "INSERT VOLUME: "
017291 MVOL        DS      15
017292          DFB      $0D              ; CR LINE TERMINATOR
017293          DFB      $20,$20,$20,$20,$69,$6E,$20
017294          DFB      $64,$65,$76,$69,$63,$65
017295          DFB      $3A,$20          ; " IN DEVICE: "
017296 MDEV       DS      15
017297          DFB      $0D              ; CR LINE TERMINATOR
017298          DFB      $74,$68,$65,$6E,$20,$70,$72
017299          DFB      $65,$73,$73,$20,$74,$68,$65,$20
017300          DFB      $41,$4C,$50,$48,$41,$20,$4C
017301          DFB      $4F,$43,$4B,$20,$6B,$65,$79
017302          DFB      $20,$74,$77,$69,$63,$65
017303 * "THEN PRESS THE ALPHA LOCK KEY TWICE"
017304 * FOLLOWED WITH $FF MESSAGE TERMINATOR (HIGH BIT SIGNIFICANT)
017305          DFB      $FF              ; MESSAGE TERMINATOR (HIGH BIT)
017306 *

```

```
017307 ZZLEN      EQU      *-ZZORG
017308 ZZEND      EQU      *
017309          IFNE      ZZLEN-LENBFM
017310          FAIL      2,"SOSORG      FILE IS INCORRECT FORMBFM"
017311          FIN
017312
017313 *****
017314 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SWAPOUT.IN
017315 *****
017316
017317
```

```
017318 =====
017319 DOCUMENT :SOS1.3.5of5.FIVE:SOS.C.BI2.TEXT
017320 =====
017321
017322 *****
017323 * APPLE /// SOS 1.3 SOURCE CODE FILE: C.BI2
017324 *****
017325 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017326
017327 :TABS 17,23,40
017328 ::PR#1,L58      132N
017329 SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
017330
017331 *****
017332 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: C.BI2
017333 *****
017334
```

```
017335 =====
017336 DOCUMENT :SOS1.3.5of5.FIVE:SOS.C.S.TEXT
017337 =====
017338
017339 *****
017340 * APPLE /// SOS 1.3 SOURCE CODE FILE: C.S
017341 *****
017342 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017343
017344 :TABS 17,23,40
017345 ::PR#1,L58      132N
017346 SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
017347 SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
017348 END
017349
017350 *****
017351 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: C.S
017352 *****
017353
```

```

017354 =====
017355 DOCUMENT :SOS1.3.5of5.FIVE:SOS.C3.TEXT
017356 =====
017357
017358 *****
017359 * APPLE /// SOS 1.3 SOURCE CODE FILE: C3
017360 *****
017361 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017362
017363 :TABS 17,23,40
017364 ::PR#1,L58      132N
017365 SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
017366 SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
017367 SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
017368 END
017369
017370 *****
017371 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: C3
017372 *****
017373

```



```

017374 =====
017375 DOCUMENT :SOS1.3.5of5.FIVE:SOS.COMP.OPR.IPL.TEXT
017376 =====
017377
017378 *****
017379 * APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.OPR.IPL
017380 *****
017381 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017382
017383 :TABS 17,23,40
017384 ::PR#1,L58      132N
017385 SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
017386 SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
017387 SL4:DR1:A,6,1
017388 END
017389
017390 *****
017391 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.OPR.IPL
017392 *****
017393

```

```

017394 =====
017395 DOCUMENT :SOS1.3.5of5.FIVE:SOS.COMP.SOS.NOLIST.TEXT
017396 =====
017397
017398 *****
017399 * APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.SOS.NOLIST
017400 *****
017401 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017402
017403 :TABS 17,23,40
017404 SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
017405 SL4:DR1:ASM INIT.SRC,INIT.OBJ,6,1
017406 SL4:DR1:ASM SYSGLOB.SRC,SYSGLOB.OBJ,6,1
017407 SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
017408 SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
017409 SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
017410 SL4:DR1:ASM UMGR.SRC,UMGR.OBJ,6,1
017411 SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
017412 SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
017413 SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
017414 SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
017415 SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
017416 SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
017417 SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
017418 SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
017419 END
017420
017421 *****
017422 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMP.SOS.NOLIST
017423 *****
017424
017425

```

```

017426 =====
017427 DOCUMENT :SOS1.3.5of5.FIVE:SOS.COMPILE.BFM.TEXT
017428 =====
017429
017430 *****
017431 * APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.BFM
017432 *****
017433 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017434
017435 :T 17,23,40
017436 ::PR#1,L58          132N
017437 ::SL4:DR1:ASM PRINT,BFM.OBJ,6,1
017438 ::END
017439
017440 *****
017441 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.BFM
017442 *****
017443

```

```

017444 =====
017445 DOCUMENT :SOS1.3.5of5.FIVE:SOS.COMPILE.SOS.TEXT
017446 =====
017447
017448 *****
017449 * APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.SOS
017450 *****
017451 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017452
017453 :TABS 17,23,40
017454 ::PR#1,L58      132N
017455 SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
017456 SL4:DR1:ASM INIT.SRC,INIT.OBJ,6,1
017457 SL4:DR1:ASM SYSGLOB.SRC,SYSGLOB.OBJ,6,1
017458 SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
017459 SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
017460 SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
017461 SL4:DR2:ASM UMGR.SRC,UMGR.OBJ,6,1
017462 SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
017463 SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
017464 SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
017465 SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
017466 SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
017467 SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
017468 SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
017469 SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
017470 ::END
017471
017472 *****
017473 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: COMPILE.SOS
017474 *****
017475
017476

```

```

017477 =====
017478 DOCUMENT :SOS1.3.5of5.FIVE:SOS.FEB01.1982.TEXT
017479 =====
017480
017481 *****
017482 * APPLE /// SOS 1.3 SOURCE CODE FILE: FEB01.1982
017483 *****
017484 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017485
017486 SL4:DR1:ASM SOSLDR.SRC,SOSLDR.OBJ,6,1
017487 SL4:DR1:ASM INIT.SRC,INIT.OBJ,6,1
017488 SL4:DR1:ASM SYSGLOB.SRC,SYSGLOB.OBJ,6,1
017489 SL4:DR1:ASM OPRMSG.SRC,OPRMSG.OBJ,6,1
017490 SL4:DR1:ASM BFM.INIT2.SRC,BFM.INIT2.OBJ,6,1
017491 SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
017492 SL4:DR1:ASM UMGR.SRC,UMGR.OBJ,6,1
017493 SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
017494 SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
017495 SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
017496 SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
017497 SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
017498 SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
017499 SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
017500 SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
017501 END
017502
017503 *****
017504 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: FEB01.1982
017505 *****
017506

```

```

017507 =====
017508 DOCUMENT :SOS1.3.5of5.FIVE:SOS.LC.TEXT
017509 =====
017510
017511 *****
017512 * APPLE /// SOS 1.3 SOURCE CODE FILE: LC
017513 *****
017514 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017515
017516             IFNE          ZZLEN-LEN???
017517             FAIL          2,"SOSORG          FILE IS INCORRECT FOR ??????"
017518             FIN
017519
017520 *****
017521 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: LC
017522 *****
017523
017524

```

```

017525 =====
017526 DOCUMENT :SOS1.3.5of5.FIVE:SOS.LCHK.TEXT
017527 =====
017528
017529 *****
017530 * APPLE /// SOS 1.3 SOURCE CODE FILE: LCHK
017531 *****
017532 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017533
017534             INCLUDE    SOSORG,6,1,254
017535             ORG        ???????
017536 -----
017537             IFNE       ZZLEN-LEN????
017538             FAIL       2,"SOSORG           FILE IS INCORRECT FOR ??????"
017539             FIN
017540
017541 *****
017542 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: LCHK
017543 *****
017544
017545

```

```

017546 =====
017547 DOCUMENT :SOS1.3.5of5.FIVE:SOS.PUBLICRELEASE.TEXT
017548 =====
017549
017550 *****
017551 * APPLE /// SOS 1.3 SOURCE CODE FILE: PUBLICRELEASE
017552 *****
017553 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017554
017555 :T 15,19,32
017556 ::PR#1,L58          132N
017557 ::SL4:DR1:ASM PRINT,BFM.OBJ,S6,D1
017558 ::END
017559
017560 *****
017561 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: PUBLICRELEASE
017562 *****
017563

```



```

017564 =====
017565 DOCUMENT :SOS1.3.5of5.FIVE:SOS.SOS.BLOAD.TEXT
017566 =====
017567
017568 *****
017569 * APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.BLOAD
017570 *****
017571 * ASSEMBLER: APPLE |[ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017572
017573 MON I
017574 CALL-151
017575 1600:0
017576 1601<1600.93FEM
017577 3D0G
017578 MON I
017579 BLOAD SOSLDR.ABS,A$2000
017580 BLOAD INIT.ABS,A$2AF8
017581 BLOAD SYSGLOB.ABS,A$2CF8
017582 BLOAD BFM.INIT2.ABS,A$2E00
017583 BLOAD BFM.ABS,A$3200
017584 BLOAD OPRMSG.ABS,A$5466
017585 BLOAD IPL.ABS,A$55C0
017586 BLOAD UMGR.ABS,A$5A8B
017587 BLOAD DISK3.ABS,A$5E99
017588 BLOAD SYSERR.ABS,A$6404
017589 BLOAD DEVMGR.ABS,A$64D9
017590 BLOAD SCMGR.ABS,A$665E
017591 BLOAD FMGR.ABS,A$68F4
017592 BLOAD CFMGR.ABS,A$6955
017593 BLOAD BUFMGR.ABS,A$6B52
017594 BLOAD MEMMGR.ABS,A$6E6E
017595 NOMON I
017596
017597 *****
017598 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.BLOAD
017599 *****
017600
017601

```

```

017602 =====
017603 DOCUMENT :SOS1.3.5of5.FIVE:SOS.SOS.LINK.TEXT
017604 =====
017605
017606 *****
017607 * APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.LINK
017608 *****
017609 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017610
017611 SOSLDR.OBJ
017612 INIT.OBJ
017613 SYSGLOB.OBJ
017614 BFM.INIT2.OBJ
017615 BFM.OBJ
017616 OPRMSG.OBJ
017617 IPL.OBJ
017618 UMGR.OBJ
017619 DISK3.OBJ
017620 SYSERR.OBJ
017621 SCMGR.OBJ
017622 FMGR.OBJ
017623 CFMGR.OBJ
017624 DEVMGR.OBJ
017625 BUFMGR.OBJ
017626 MEMMGR.OBJ
017627 END
017628
017629 *****
017630 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.LINK
017631 *****
017632

```

```

017633 =====
017634 DOCUMENT :SOS1.3.5of5.FIVE:SOS.SOS.RENAME.TEXT
017635 =====
017636
017637 *****
017638 * APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.RENAME
017639 *****
017640 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017641
017642 MON I
017643 RENAME SOSLDR.OBJ.ABS,SOSLDR.ABS
017644 RENAME INIT.OBJ.ABS,INIT.ABS
017645 RENAME SYSGLOB.OBJ.ABS,SYSGLOB.ABS
017646 RENAME OPRMSG.OBJ.ABS,OPRMSG.ABS
017647 RENAME BFM.OBJ.ABS,BFM.ABS
017648 RENAME BFM.INIT2.OBJ.ABS,BFM.INIT2.ABS
017649 RENAME IPL.OBJ.ABS,IPL.ABS
017650 RENAME UMGR.OBJ.ABS,UMGR.ABS
017651 RENAME DISK3.OBJ.ABS,DISK3.ABS
017652 RENAME SYSERR.OBJ.ABS,SYSERR.ABS
017653 RENAME SCMGR.OBJ.ABS,SCMGR.ABS
017654 RENAME FMGR.OBJ.ABS,FMGR.ABS
017655 RENAME CFMGR.OBJ.ABS,CFMGR.ABS
017656 RENAME DEVMGR.OBJ.ABS,DEVMGR.ABS
017657 RENAME BUFMGR.OBJ.ABS,BUFMGR.ABS
017658 RENAME MEMMGR.OBJ.ABS,MEMMGR.ABS
017659 NOMON I
017660
017661 *****
017662 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOS.RENAME
017663 *****
017664

```

```

017665 =====
017666 DOCUMENT :SOS1.3.5of5.FIVE:SOS.SOSORG.TEXT
017667 =====
017668
017669 *****
017670 * APPLE /// SOS 1.3 SOURCE CODE FILE: SOSORG
017671 *****
017672 * ASSEMBLER: APPLE |[ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017673
017674             REP          100
017675 *   SOS KERNEL MODULE ORIGINS
017676 ORGLODR      EQU          $1E00             ; ORIGIN OF SOS LOADER
017677 ORGINIT      EQU          $28F8             ; ORIGIN OF INIT
017678 ORGLOB       EQU          $18FC             ; ORIGIN OF SYSGLOB
017679 ORGBFMI      EQU          $B800             ; ORIGIN OF BFM.INIT2 & BITMAPS
017680 ORGBFM       EQU          $BC00             ; ORIGIN OF BFM
017681 ORGPATCH    EQU          $DE66             ; ORIGIN OF PATCH AREA
017682 ORGOMSG     EQU          $DE66             ; ORIGIN OF OPRMSG
017683 ORGIPL      EQU          $DFC0             ; ORIGIN OF IPL
017684 ORGUMGR     EQU          $E48B             ; ORIGIN OF UMGR
017685 ORGDISK3    EQU          $E899             ; ORIGIN OF DISK3
017686 ORGSERR     EQU          $EE04             ; ORIGIN OF SYSERR
017687 ORGDMGR     EQU          $EED9             ; ORIGIN OF DEVMGR
017688 ORGSCMGR    EQU          $F05E             ; ORIGIN OF SCMGR
017689 ORGFMGR     EQU          $F2F4             ; ORIGIN OF FMGR
017690 ORGCFM       EQU          $F355             ; ORIGIN OF CFMGR
017691 ORGBUFMG    EQU          $F552             ; ORIGIN OF BUFMGR
017692 ORGMEMMG    EQU          $F86E             ; ORIGIN OF MEMMGR
017693 ORGEND      EQU          $FFBF             ; END MARKER
017694             REP          100
017695 *   LENGTH OF SOS MODULES -- THIS MUST AGREE WITH ZZLEN FOR EACH MODULE
017696 LENLODR      EQU          ORGINIT-ORGLODR    ; LENGTH OF SOS LOADER
017697 LENINIT      EQU          $01B2             ; LENGTH OF INIT
017698 LENBFMI      EQU          ORGBFM-ORGBFMI    ; LENGTH OF BFM.INIT2 & BITMAPS
017699 LENBFM       EQU          ORGPATCH-ORGBFM   ; LENGTH OF BFM
017700 LENPATCH    EQU          ORGOMSG-ORGPATCH  ; LENGTH OF PATCH AREA
017701 LENOMSG     EQU          ORGIPL-ORGOMSG    ; LENGTH OF OPRMSG
017702 LENIPL      EQU          ORGUMGR-ORGIPL    ; LENGTH OF IPL
017703 LENUMGR     EQU          ORGDISK3-ORGUMGR  ; LENGTH OF UMGR
017704 LENDISK3    EQU          ORGSERR-ORGDISK3   ; LENGTH OF DISK3
017705 LENSERR     EQU          ORGDMGR-ORGSERR   ; LENGTH OF SYSERR
017706 LENDMGR     EQU          ORGSCMGR-ORGDMGR  ; LENGTH OF DEVMGR
017707 LENSCLMGR   EQU          ORGFMGR-ORGSCMGR  ; LENGTH OF SCMGR
017708 LENFMGR     EQU          ORGCFM-ORGFMGR    ; LENGTH OF FMGR
017709 LENCFM       EQU          ORGBUFMG-ORGCFM   ; ORIGIN OF CFMGR
017710 LENBUFMG    EQU          ORGMEMMG-ORGBUFMG  ; LENGTH OF BUFMGR
017711 LENMEMMG    EQU          ORGEND-ORGMEMMG   ; LENGTH OF MEMMGR
017712             REP          100
017713 *           SOS BLOAD ADDRESSES

```

```

017714 BLALODR      EQU      $2000      ; BLOAD ADDRESS OF SOS LOADER
017715 BLAINIT      EQU      BLALODR+LENLODR ; BLOAD ADDRESS OF INIT
017716 BLAGLOB      EQU      $2CF8      ; BLOAD ADDRESS OF SYSGLOB
017717 BLABFMI      EQU      $2E00      ; BLOAD ADDRESS OF BFM.INIT2 & BITMAPS
017718 BLABFM       EQU      $3200      ; BLOAD ADDRESS OF BFM
017719 BLAPATCH     EQU      BLABFM+LENBFM ; BLOAD ADDRESS OF PATCH AREA
017720 BLAOMSG      EQU      BLAPATCH+LENPATCH ; BLOAD ADDRESS OF OPRMSG
017721 BLAIPL       EQU      BLAOMSG+LENOMSG ; BLOAD ADDRESS OF IPL
017722 BLAUMGR      EQU      BLAIPL+LENIPL  ; BLOAD ADDRESS OF UMGR
017723 BLADISK3     EQU      BLAUMGR+LENUMGR ; BLOAD ADDRESS OF DISK3
017724 BLASERR      EQU      BLADISK3+LENDISK3 ; BLOAD ADDRESS OF SYSERR
017725 BLADMGR      EQU      BLASERR+LENSERR ; BLOAD ADDRESS OF DEVMGR
017726 BLASCMGR     EQU      BLADMGR+LENDMGR ; BLOAD ADDRESS OF SCMGR
017727 BLAFMGR      EQU      BLASCMGR+LENSCMGR ; BLOAD ADDRESS OF FMGR
017728 BLACFM      EQU      BLAFMGR+LENFMGR ; BLOAD ADDRESS OF CFMGR
017729 BLABUFMG     EQU      BLACFM+LENCFM  ; BLOAD ADDRESS OF BUFMGR
017730 BLAMEMMG     EQU      BLABUFMG+LENBUFMG ; BLOAD ADDRESS OF MEMMGR
017731              REP      100
017732
017733 *****
017734 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: SOSORG
017735 *****
017736
017737

```

```

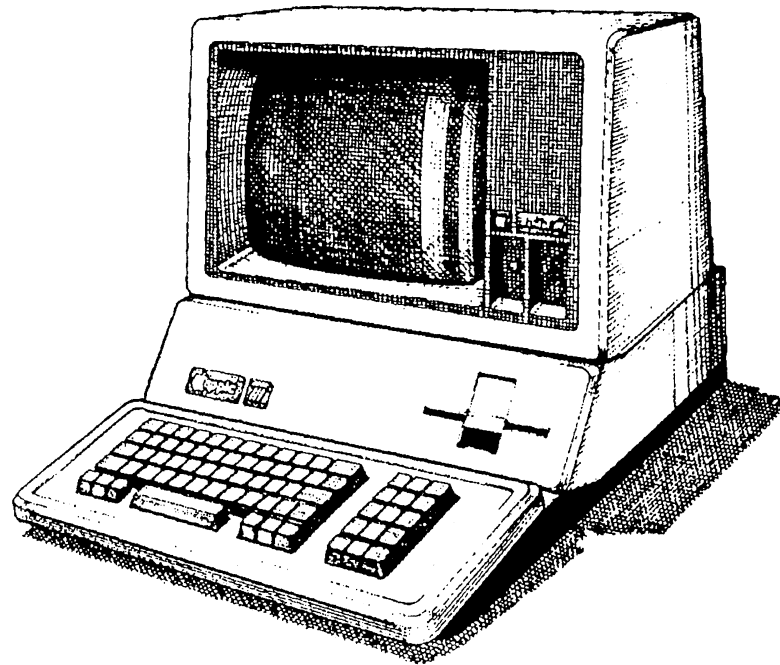
017738 =====
017739 DOCUMENT :SOS1.3.5of5.FIVE:SOS.TCOMP.SOS.TEXT
017740 =====
017741
017742 *****
017743 * APPLE /// SOS 1.3 SOURCE CODE FILE: TCOMP.SOS
017744 *****
017745 * ASSEMBLER: APPLE ][ 6502 ASSEMBLER from APPLE COMPUTER TOOLKIT
017746
017747 :TABS 17,23,40
017748 SL4:DR1:ASM IPL.SRC1,IPL.OBJ,6,1
017749 SL4:DR1:ASM UMGR.SRC,UMGR.OBJ,6,1
017750 SL4:DR2:ASM DISK3.SRC,DISK3.OBJ,6,1
017751 SL4:DR2:ASM SYSERR.SRC,SYSERR.OBJ,6,1
017752 SL4:DR2:ASM SCMGR.SRC,SCMGR.OBJ,6,1
017753 SL4:DR2:ASM FMGR.SRC,FMGR.OBJ,6,1
017754 SL4:DR2:ASM CFMGR.SRC,CFMGR.OBJ,6,1
017755 SL4:DR2:ASM DEVMGR.SRC,DEVMGR.OBJ,6,1
017756 SL4:DR2:ASM BUFMGR.SRC,BUFMGR.OBJ,6,1
017757 SL4:DR2:ASM MEMMGR.A.SRC,MEMMGR.OBJ,6,1
017758
017759 *****
017760 * END OF APPLE /// SOS 1.3 SOURCE CODE FILE: TCOMP.SOS
017761 *****

```

End of File -- Lines: 17761 Characters: 568225

SUMMARY:

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



**The End**