

!A

LLOAD INCL.L,A\$4000

LLOAD DOS.3.3.L,A\$4000

*** End of Pass 1

LLOAD INCL.L,A\$4000

LLOAD DISK.L,A\$4000

LLOAD BUFR.L,A\$4000

LLOAD CMD1.L,A\$4000

LLOAD CMD2.L,A\$4000

LLOAD CMD3.L,A\$4000

LLOAD CMD4.L,A\$4000

LLOAD MNGR1A.L,D2,A\$4000

LLOAD MNGR1B.L,A\$4000

LLOAD MNGR2A.L,A\$4000

LLOAD MNGR2B.L,A\$4000

LLOAD RWTS1.L,D2,A\$4000

LLOAD RWTS2.L,A\$4000

LLOAD DOS.3.3.L,A\$4000

*** End of Pass 2

```
0800      1          ttl "DOS 3.3 Source Code, DOS.3.3.L"
0800      2          src "DOS.3.3.L"
0800      3      ;
0800      4      ;
0800      5      ; DOS.3.3.L
0800      6      ;
0800      7      ;
0800      8      ; DOS 3.3 Source Code
0800      9      ;
0800     10      ; 2024 February 14
0800     11      ;
0800     12      ;
0800     13      ; DOS 4.5, Build 05
0800     14      ;
0800     15      ; 2024 February 14
0800     16      ;
0800     17      ;
0800     18      ; Start of Source Code: 0x4000
0800     19      ; Start of Symbol List: 0x7800
0800     20      ;
0800     21      ;
0800     22      ; Copyright (c) 2024 February 14 by
0800     23      ; Walland Philip Vrbancic Jr
0800     24      ;
0800     25      ; 6223 East Peabody Street
0800     26      ; Long Beach, California 90808
0800     27      ; Unitied States of America
0800     28      ;
0800     29      ; All Rights Reserved
0800     30      ;
0800     31      ; This software is the confidential and
0800     32      ; proprietary intellectual property of
0800     33      ; Walland Philip Vrbancic Jr
0800     34      ;
0800     35      ;
0800     36      ; The following source code is the complete Apple DOS,
0800     37      ; version 3.3 with no modifications to the code.
0800     38      ;
0800     39      ; I have documented this source code using a number of
0800     40      ; resources including APPLE ][ Reference Manual, Beneath
0800     41      ; Apple DOS by Don Worth and Pieter Lechner, and What's
0800     42      ; Where in the APPLE A Complete Guide to the Apple Computer
0800     43      ; by William F. Luebbert.
0800     44      ;
0800     45      ; The target compiler is Lisa V2.6 by Randy Hyde.
0800     46      ;
0800     47      ;
0800     48          icl "INCL.L"
```

```
LLOAD INCL.L,A$4000
```

```

0800      1          ttl "DOS 3.3 Source Code, INCL.L"
0800      2      ;
0800      3      ;
0800      4      ; INCL.L
0800      5      ;
0800      6      ;
0024      7  CH          epz $24          ; horizontal cursor position
0026      8  BUFRADRZ    epz $26          ; boot firmware data address
0026      9  TEMPZ       epz $26          ; RWTS temporary variable
0027     10  TEMP2Z      epz $27          ; RWTS temporary variable
0028     11  BASEZ       epz $28          ; current text line
002A     12  SEEKTRKZ    epz $2A          ; RWTS seek track number
002B     13  SLOT16Z     epz $2B          ; RWTS slot number
002C     14  DATAFNDZ   epz $2C          ; RWTS disk data address
002D     15  SECFNDZ     epz $2D          ; RWTS read header sector
002E     16  TRKFNDZ     epz $2E          ; RWTS read header track
002F     17  VOLFNDZ     epz $2F          ; RWTS read header volume
0800     18      ;
0033     19  PROMPT      epz $33          ; prompt character
0035     20  DRIVNO      epz $35          ; drive number
0036     21  CSWL        epz $36          ; data output address
0038     22  KSWL        epz $38          ; data input address
003C     23  ROMTEMPZ    epz $3C          ; boot firmware temp variable
003C     24  DEVCTBL     epz $3C          ; RWTS device table address
003D     25  ROMSECTR    epz $3D          ; boot sector number
003E     26  BOOTADRZ    epz $3E          ; boot firmware entry address
003E     27  BUFADR2Z    epz $3E          ; RWTS pre nibble data address
003E     28  ODDBITSZ    epz $3E          ; RWTS write header temp
003F     29  SECTORZ     epz $3F          ; RWTS write header sector
0800     30      ;
0040     31  ROMDATA     epz $40          ; boot header data (track)
0040     32  FILEBUFZ    epz $40          ; DOS file buffer address
0041     33  ROMTRACK    epz $41          ; boot track number
0041     34  VOLUMEZ     epz $41          ; RWTS write header volume
0042     35  BUFADRZ     epz $42          ; DOS data buffer address
0044     36  TRACKZ      epz $44          ; RWTS write header track
0044     37  OPRND       epz $44          ; DOS math operand
0045     38  SYNCNT      epz $45          ; RWTS format count
0046     39  MONTIME     epz $46          ; RWTS motor on time
0048     40  IOBADR      epz $48          ; File Manager IOB address
004A     41  INTLOMEM    epz $4A          ; Integer Basic program addr
004C     42  INTHIMEM    epz $4C          ; Integer Basic program addr
0800     43      ;
0067     44  ASPGMST     epz $67          ; Applesoft program address
0069     45  ASVARS      epz $69          ; Applesoft program variables
006F     46  ASSTRS      epz $6F          ; Applesoft program strings
0800     47      ;
0073     48  ASHIMEM     epz $73          ; Applesoft HIMEM location
0076     49  ASRUN       epz $76          ; Applesoft running flag
0800     50      ;
00AF     51  ASPEND      epz $AF          ; Applesoft program end
0800     52      ;
00CA     53  INTSTRT     epz $CA          ; Integer Basic program start
00CC     54  INTVEND     epz $CC          ; Integer Basic program end
0800     55      ;
00D6     56  PROTECT     epz $D6          ; Applesoft source protect
00D8     57  ASONERR     epz $D8          ; Applesoft ONERR flag
00D9     58  INTRUN      epz $D9          ; Integer Basic run flag
0800     59      ;
0800     60      enz

```

```
0800      61 ;
0000      62 INTEGER equ 0 ; compile option for BASIC
0800      63 ;
0000      64 ZERO equ 0 ; value of zero
0060      65 FLASHSPC equ $60 ; flashing space charcter
007F      66 ASCIMASK equ $7F ; mask to remove high bit
0080      67 ASCIFLAG equ $80 ; flag for normal ASCII
0800      68 ;
0084      69 CTRLD equ $84 ; ASCII value of control-D
008A      70 LINEFEED equ $8A ; ASCII value of line feed
008D      71 RETURN equ $8D ; ASCII value of return
00A0      72 SPACE equ $A0 ; ASCII value of space
00E0      73 LWRCASE equ $E0 ; start of ASCII lower case
00FE      74 DEFLTVOL equ $FE ; default volume complimented
00FF      75 NEGONE equ $FF ; value of negative one
0800      76 ;
0016      77 LNSPERPG equ 22 ; catalog lines per page
001E      78 NAMESIZE equ 30 ; length of file name
0023      79 LASTTRACK equ 35 ; last track on disk
0020      80 INTBASIC equ $20 ; Integer BASIC code
004C      81 FPBASIC equ $4C ; Applesoft BASIC code
00A5      82 PWRUPBYT equ $A5 ; power up byte
0800      83 ;
0000      84 INTACTV equ $00 ; Integer BASIC active
0040      85 FPOACTV equ $40 ; Applesoft ROM active
0080      86 FPAACTV equ $80 ; Applesoft RAM active
0800      87 ;
0000      88 STATE0 equ $00 ; start of command line
0001      89 STATE1 equ $01 ; parse for DOS command
0002      90 STATE2 equ $02 ; non-DOS command
0003      91 STATE3 equ $03 ; input data
0004      92 STATE4 equ $04 ; write data to a file
0005      93 STATE5 equ $05 ; start writing data line
0006      94 STATE6 equ $06 ; finish output
0800      95 ;
0001      96 READMODE equ $01 ; read mode for CURSTAT
0800      97 ;
0010      98 MONOMASK equ $10 ; mask to check for MON O
0020      99 MONIMASK equ $20 ; mask to check for MON I
0040      100 MONCMASK equ $40 ; mask to check for MON C
007F      101 MONMASK equ $7F ; mask for PARMBITS
0080      102 MONFLAG equ $80 ; mon flag in KYWRDFND
0800      103 ;
0000      104 TXTFLTYP equ $00 ; TEXT file type
0001      105 INTFLTYP equ $01 ; Integer BASIC file type
0002      106 ASFLTYP equ $02 ; Applesoft BASIC file type
0004      107 BINFLTYP equ $04 ; BINARY file type
0008      108 SFILETYP equ $08 ; S file type
0010      109 RECFLTYP equ $10 ; Relocatable object file type
0020      110 AFILETYP equ $20 ; A file type
0040      111 BFILETYP equ $40 ; B file type
007F      112 LOCKMASK equ $7F ; mask out lock flag
0080      113 LOCKFLAG equ $80 ; lock flag
00FF      114 DELETFLG equ $FF ; delete flag in catalog
0800      115 ;
0001      116 NWFLMSK equ $01 ; create new file mask
0002      117 PGCMDMSK equ $02 ; program command mask
0004      118 MXFLEMSK equ $04 ; MAXFILES expected mask
0008      119 SLNUMMSK equ $08 ; slot number mask
0010      120 FLNM2MSK equ $10 ; filename 2 mask
0020      121 FLNM1MSK equ $20 ; filename 1 mask
```

0040	122	CMDOPMSK equ \$40	; command operand mask
0080	123	FNOPMSK equ \$80	; filename optional mask
0800	124	;	
0001	125	AKEYMASK equ \$01	; A keyword legal
0002	126	BKEYMASK equ \$02	; B keyword legal
0004	127	RKEYMASK equ \$04	; R keyword legal
0008	128	LKEYMASK equ \$08	; L keyword legal
0010	129	SKEYMASK equ \$10	; S keyword legal
0020	130	DKEYMASK equ \$20	; D keyword legal
0040	131	VKEYMASK equ \$40	; V keyword legal
0080	132	MKEYMASK equ \$80	; C, I, O keywords legal
0800	133	;	
0002	134	VTOCMASK equ \$02	; VTOC map has changed
0040	135	DATAMASK equ \$40	; data buffer has changed
0080	136	TSBFRMSK equ \$80	; T/S buffer has changed
0800	137	;	
0001	138	FMOPENCD equ \$01	; File Manager OPEN code
0002	139	FMCLOSCD equ \$02	; File Manager CLOSE code
0003	140	FMREADCD equ \$03	; File Manager READ code
0004	141	FMWRITCD equ \$04	; File Manager WRITE code
0005	142	FMDELECD equ \$05	; File Manager DELETE code
0006	143	FMCATACD equ \$06	; File Manager CATALOG code
0007	144	FMLOCKCD equ \$07	; File Manager LOCK code
0008	145	FMUNLKCD equ \$08	; File Manager UNLOCK code
0009	146	FMRENMCD equ \$09	; File Manager RENAME code
000A	147	FMPOSICD equ \$0A	; File Manager POSITION code
000B	148	FMINITCD equ \$0B	; File Manager INIT code
000C	149	FMVERICD equ \$0C	; File Manager VERIFY code
0800	150	;	
0000	151	FMNOOPSC equ \$00	; FM No Operation subcode
0001	152	FMRW01SC equ \$01	; FM R/W one byte subcode
0002	153	FMRWNBSC equ \$02	; FM R/W range of bytes
0003	154	FMPOS1SC equ \$03	; FM Position and R/W 1 byte
0004	155	FMPOSNSC equ \$04	; FM Position and R/W range
0800	156	;	
0000	157	FMNOERR equ \$00	; FM no errors
0001	158	FMLNAERR equ \$01	; FM LANGUAGE NOT AVAILABLE
0002	159	FMOPTERR equ \$02	; FM bad call type
0003	160	FMSUBERR equ \$03	; FM bad sub-call type
0004	161	FMPROTER equ \$04	; FM WRITE PROTECTED
0005	162	FMEOFERR equ \$05	; FM END OF DATA
0006	163	FMNOFLER equ \$06	; FM FILE NOT FOUND
0007	164	FMVOLERR equ \$07	; FM VOLUME MISMATCH
0008	165	FMDIOERR equ \$08	; FM DISK I/O ERROR
0009	166	FMFULLER equ \$09	; FM DISK FULL
000A	167	FMLOCKER equ \$0A	; FM FILE LOCKED
0800	168	;	
0000	169	RWTSSEEK equ \$00	; RWTS SEEK command code
0001	170	RWTSREAD equ \$01	; RWTS READ command code
0002	171	RWTSWRIT equ \$02	; RWTS WRITE command code
0004	172	RWTSFRMT equ \$04	; RWTS FORMAT command code
0800	173	;	
0000	174	RWNOERR equ \$00	; RWTS no error
0008	175	RWINITER equ \$08	; RWTS initialization error
0010	176	RWPROTER equ \$10	; RWTS write protect error
0020	177	RWVOLERR equ \$20	; RWTS volume mismatch error
0040	178	RWDRVERR equ \$40	; RWTS drive error
0080	179	RWREADER equ \$80	; RWTS read error (obsolete)
0800	180	;	
0001	181	TSTRKOFF equ \$01	; T/S track offset
0002	182	TSSECOFF equ \$02	; T/S sector offset

0005	183	TSRECOFF equ \$05	; T/S record offset location
000C	184	TSLSTOFF equ \$0C	; T/S list begin offset
0800	185	;	
0011	186	VTOCTRK equ \$11	; VTOC track for catalog
0023	187	VTOCENSZ equ \$23	; VTOC catalog entry size
00F5	188	VTOCEND equ \$F5	; VTOC end of entries check
0800	189	;	
0001	190	ALOCFRWD equ \$01	; forward sector allocation
00FF	191	ALOCBKWD equ \$FF	; backward sector allocation
0800	192	;	
003F	193	NBUFMASK equ \$3F	; high order mask for NBUF2
0056	194	NBUF2SIZ equ \$56	; size of NBUF2
00AA	195	ODDBITS equ \$AA	; odd bits of odd/even encode
00D5	196	ADRMARK1 equ \$D5	; first address mark
00AA	197	ADRMARK2 equ \$AA	; second address mark
0096	198	ADRMARK3 equ \$96	; third address mark
00D5	199	DATMARK1 equ \$D5	; first data mark
00AA	200	DATMARK2 equ \$AA	; second data mark
00AD	201	DATMARK3 equ \$AD	; third data mark
00DE	202	SLPMARK1 equ \$DE	; first slip mark
00AA	203	SLPMARK2 equ \$AA	; second slip mark
00EB	204	SLPMARK3 equ \$EB	; third slip mark
00FF	205	SYNCMARK equ \$FF	; sync mark
0800	206	;	
0100	207	PAGESIZE equ \$100	; page size
0100	208	STACK equ \$100	; processor stack location
0800	209	;	
0200	210	INPUT equ \$200	; user input buffer
0800	211	;	
0300	212	NBUF2BT equ \$300	; boot NBUF2
036C	213	RDNIBLBT equ \$36C	; boot RDNIBL
03D0	214	DOSRST equ \$3D0	; DOS warm start
03F2	215	SOFTEV equ \$3F2	; autostart ROM reset vector
03F4	216	PWREDUP equ \$3F4	; autostart ROM power up mask
0800	217	;	
0478	218	CURTRK equ \$478	; RWTS seek current track
04F8	219	SEEKCNT equ \$4F8	; RWTS seek count
04FB	220	XMODE equ \$4FB	; video firmware mode
0800	221	;	
0478	222	DRV0TRK equ \$478	; requires slot index
04F8	223	DRV1TRK equ \$4F8	; requires slot index
0800	224	;	
0578	225	RETRYCNT equ \$578	; RWTS format retry count
05F8	226	SLOTSAV equ \$5F8	; RWTS format slot save
0800	227	;	
0678	228	SLOTABS equ \$678	; RWTS write data slot save
06F8	229	RECALCNT equ \$6F8	; RWTS seek recalibrate count
0800	230	;	
0800	231	PAGE08 equ \$800	; address of \$800
08FD	232	BOOTVALS equ \$8FD	; address of boot parameters
0800	233	;	
0C3C	234	ASRAMWRM equ \$C3C	; address of RAM warmstart
0CF2	235	ASRAMRST equ \$CF2	; address of RAM restart
0E65	236	ASRAMCLR equ \$E65	; address of RAM clear
0FD4	237	ASRAMNEW equ \$FD4	; address of RAM RUN entry
0800	238	;	
1000	239	PAGE10 equ \$1000	; generated object code
1067	240	ASRAMERR equ \$1067	; RAM A/S error processing?
0800	241	;	
9AA6	242	DOSFBADR equ \$9AA6	; DOS first file buffer addr
0800	243	;	

```
C000      244  MEMTOP      equ  $C000      ; top of physical memory
0800      245  ;
C00C      246  VID80OFF equ  $C00C      ; disable 80 column video
C00E      247  ALTCHOFF equ  $C00E      ; enable normal character set
0800      248  ;
C080      249  RAM2WP      equ  $C080      ; bank 2 on, W/P RAM
C081      250  ROM2WE      equ  $C081      ; bank 2 off, ROM on, W/E RAM
0800      251  ;
C080      252  PHASEOFF equ  $C080      ; stepper motor phase 0 off
C081      253  PHASEON  equ  $C081      ; stepper motor phase 0 on
C088      254  MOTOROFF equ  $C088      ; turn motor off
C089      255  MOTORON  equ  $C089      ; turn motor on
C08A      256  DRV0EN      equ  $C08A      ; engage drive 1
C08B      257  DRV1EN      equ  $C08B      ; engage drive 2
C08C      258  STROBE      equ  $C08C      ; strobe data latch for I/O
C08D      259  LATCH       equ  $C08D      ; load data latch
C08E      260  DATAIN     equ  $C08E      ; prepare latch for input
C08F      261  DATAOUT    equ  $C08F      ; prepare latch for output
0800      262  ;
0800      263  ;
0800      264  ; Applesoft Equates
0800      265  ;
D43C      266  ASROMWRM equ  $D43C      ; ROM Applesoft soft entry
D4F2      267  ASROMRST equ  $D4F2      ; ROM A/S set/reset links
D665      268  ASROMCLR equ  $D665      ; ROM A/S clear command
D7D2      269  ASROMNEW equ  $D7D2      ; ROM A/S NEW statement
D865      270  ASROMERR equ  $D865      ; ROM A/S error processing
0800      271  ;
0800      272  ;
0800      273  ; Integer Equates
0800      274  ;
E000      275  BASCLD      equ  $E000      ; Integer hard entry
E003      276  BASWRM      equ  $E003      ; Integer soft entry
E3E3      277  INTERR      equ  $E3E3      ; Integer error processing
E836      278  IRUN        equ  $E836      ; Integer RUN command
0800      279  ;
0800      280  ;
0800      281  ; Monitor Equates
0800      282  ;
FA59      283  OLDBRK      equ  $FA59      ; autostart monitor OLDBRK
FB2F      284  INIT        equ  $FB2F      ; monitor reset text mode
FC58      285  HOME        equ  $FC58      ; monitor HOME subroutine
FCA8      286  WAIT        equ  $FCA8      ; monitor WAIT subroutine
FD0C      287  RDKEY       equ  $FD0C      ; monitor RDKEY subroutine
FD0C      288  PRBYTE      equ  $FD0C      ; monitor PRBYTE subroutine
FD0C      289  COUT        equ  $FD0C      ; monitor COUT subroutine
FE89      290  SETKBD      equ  $FE89      ; monitor SETKBD subroutine
FE8B      291  INPORT      equ  $FE8B      ; monitor INPORT subroutine
FE93      292  SETVID      equ  $FE93      ; monitor SETVID subroutine
FE95      293  OUTPORT     equ  $FE95      ; monitor OUTPORT subroutine
FF58      294  IORTS       equ  $FF58      ; monitor IORTS subroutine
FF65      295  MON         equ  $FF65      ; monitor MON subroutine
0800      296  ;
0800      297  ;
0800      298  icl "DISK.L"
```

LLOAD DISK.L,A\$4000

```

0800          1          ttl "DOS 3.3 Source Code, DISK.L"
0800          2          ;
0800          3          ;
0800          4          ; DISK.L
0800          5          ;
0800          6          ;
C000          7          org MEMTOP
C000          8          obj PAGE10
C000          9          ;
C000         10          ;
C000         11          ; Signature bytes for DISK ][.
C000         12          ;
C000 A2 20      13          ldx #$20
C002 A0 00      14          ldy #$00
C004 A2 03      15          ldx #$03
C006          16          ;
C006          17          ;
C006          18          ; Build RDNIBLBT table from $359 to $3D5. The usefull
C006          19          ; portion of the table is $36C to $3D5.
C006          20          ;
C006          21          ; For a given X-reg, if A-reg becomes zero, save Y-reg
C006          22          ; indexed by X-reg. The first useful X-reg value is $16.
C006          23          ;
C006 86 3C      24          ^1      stx ROMTEMPZ
C008          25          ;
C008 8A         26          txa
C009 0A         27          asl
C00A          28          ;
C00A          29          ; Values that fail here:
C00A          30          ; 04 05 08 09 0A 10 11 12 14 15 20 21 22 24 25
C00A          31          ; 28 29 2A 40 41 42 44 45 48 49 4A 50 51 52 54 55
C00A          32          ;
C00A 24 3C      33          bit ROMTEMPZ
C00C F0 10      34          beq >3
C00E          35          ;
C00E 05 3C      36          ora ROMTEMPZ
C010 49 FF      37          eor #$FF
C012 29 7E      38          and #$7E
C014          39          ;
C014          40          ; Values that fail here:
C014          41          ; 03 06 07 0B 0C 0D 0E 0F 13 18 19 1C 23 30 31
C014          42          ; 38 43 46 47 4C 58 60 61 62 63 64 68 70 71 78
C014          43          ;
C014 B0 08      44          ^2      bcs >3
C016          45          ;
C016 4A         46          lsr
C017 D0 FB      47          bne <2
C019          48          ;
C019 98         49          tya
C01A 9D 56 03   50          sta RDNIBLBT-$96-$80,X
C01D          51          ;
C01D C8         52          iny
C01E          53          ;
C01E E8         54          ^3      inx
C01F 10 E5      55          bpl <1
C021          56          ;
C021          57          ;
C021          58          ; Determine slot number and multiply by 16.
C021          59          ;
C021 20 58 FF   60          jsr IORTS

```



```

C024      61 ;
C024 BA    62      tsx
C025      63 ;
C025 BD 00 01 64      lda STACK,X
C028      65 ;
C028 0A     66      asl
C029 0A     67      asl
C02A 0A     68      asl
C02B 0A     69      asl
C02C      70 ;
C02C 85 2B   71      sta SLOT16Z
C02E AA     72      tax
C02F      73 ;
C02F      74 ;
C02F      75 ; Establish read mode.
C02F      76 ;
C02F BD 8E C0 77      lda DATAIN,X
C032 BD 8C C0 78      lda STROBE,X
C035      79 ;
C035      80 ;
C035      81 ; Select drive 1 and enable motor.
C035      82 ;
C035 BD 8A C0 83      lda DRV0EN,X
C038 BD 89 C0 84      lda MOTORON,X
C03B      85 ;
C03B      86 ;
C03B      87 ; Recalibrate the disk head. Assume the disk head is at
C03B      88 ; track 40 (A-reg = 2 * 40).
C03B      89 ;
C03B      90 ; This is a four phase motor. Each phase must be
C03B      91 ; energized for a period of time, then de-energized. To
C03B      92 ; cause the motor to rotate the next phase windings are
C03B      93 ; selected.
C03B      94 ;
C03B A0 50   95      ldy #80
C03D      96 ;
C03D BD 80 C0 97 ^1    lda PHASEOFF,X
C040      98 ;
C040 98     99      tya
C041     100 ;
C041 29 03   101      and #$03
C043 0A     102      asl
C044 05 2B   103      ora SLOT16Z
C046      104 ;
C046 AA     105      tax
C047      106 ;
C047 BD 81 C0 107      lda PHASEON,X
C04A      108 ;
C04A A9 56   109      lda #$56
C04C 20 A8 FC 110      jsr WAIT
C04F      111 ;
C04F 88     112      dey
C050 10 EB   113      bpl <1
C052      114 ;
C052      115 ;
C052      116 ; Initialize variables with A-reg = 0 (return from WAIT).
C052      117 ;
C052 85 26   118      sta BUFRADRZ
C054 85 3D   119      sta ROMSECTR
C056 85 41   120      sta ROMTRACK
C058      121 ;

```

```

C058 A9 08      122      lda /PAGE08
C05A 85 27      123      sta BUFRADRZ+1
C05C           124      ;
C05C           125      ;
C05C           126      ; Boot firmware entry.  If C-flag = 0, read address header.
C05C           127      ; If C-flag = 1, read data header.
C05C           128      ;
C05C           129      BOOTFW:
C05C 18         130      clc
C05D           131      ;
C05D           132      BOOTFW2:
C05D 08         133      php
C05E           134      ;
C05E           135      ;
C05E           136      ; Look for first address/data mark ($D5).
C05E           137      ;
C05E BD 8C C0   138      ^1      lda STROBE,X
C061 10 FB      139      bpl <1
C063           140      ;
C063 49 D5      141      ^2      eor #ADRMARK1
C065 D0 F7      142      bne <1
C067           143      ;
C067           144      ;
C067           145      ; Look for second address/data mark ($AA).
C067           146      ;
C067 BD 8C C0   147      ^3      lda STROBE,X
C06A 10 FB      148      bpl <3
C06C           149      ;
C06C C9 AA      150      cmp #ADRMARK2
C06E D0 F3      151      bne <2
C070           152      ;
C070 EA         153      nop
C071           154      ;
C071           155      ;
C071           156      ; Look for third address or data mark ($96 or $AD).
C071           157      ;
C071 BD 8C C0   158      ^4      lda STROBE,X
C074 10 FB      159      bpl <4
C076           160      ;
C076 C9 96      161      cmp #ADRMARK3
C078 F0 09      162      beq FNDADDR
C07A           163      ;
C07A 28         164      plp
C07B 90 DF      165      bcc BOOTFW
C07D           166      ;
C07D 49 AD      167      eor #DATMARK3
C07F F0 25      168      beq FNDDATA
C081           169      ;
C081 D0 D9      170      bne BOOTFW
C083           171      ;
C083           172      ;
C083           173      ; An Address header has been found.  Read in volume, track,
C083           174      ; sector.  Checksum is ignored.  Only looking for sector in
C083           175      ; A-reg.  ROMDATA will contain the track.
C083           176      ;
C083           177      FNDADDR:
C083 A0 03      178      ldy #3
C085           179      ;
C085 85 40      180      ^1      sta ROMDATA
C087           181      ;
C087 BD 8C C0   182      ^2      lda STROBE,X

```

```

C08A 10 FB      183      bpl <2
C08C           184      ;
C08C 2A         185      rol
C08D 85 3C      186      sta ROMTEMPZ
C08F           187      ;
C08F BD 8C C0   188      ^3      lda STROBE,X
C092 10 FB      189      bpl <3
C094           190      ;
C094 25 3C      191      and ROMTEMPZ
C096           192      ;
C096 88         193      dey
C097 D0 EC      194      bne <1
C099           195      ;
C099           196      ;
C099           197      ; Fix stack pointer from BOOTFW.
C099           198      ;
C099 28         199      plp
C09A           200      ;
C09A           201      ;
C09A           202      ; Is this the correct sector?
C09A           203      ;
C09A C5 3D      204      cmp ROMSECTR
C09C D0 BE      205      bne BOOTFW
C09E           206      ;
C09E           207      ;
C09E           208      ; At correct sector. Is this the correct track?
C09E           209      ;
C09E A5 40      210      lda ROMDATA
C0A0 C5 41      211      cmp ROMTRACK
C0A2 D0 B8      212      bne BOOTFW
C0A4           213      ;
C0A4           214      ;
C0A4           215      ; At correct track and sector. Read the following data
C0A4           216      ; header and the data.
C0A4           217      ;
C0A4 B0 B7      218      bcs BOOTFW2
C0A6           219      ;
C0A6           220      ;
C0A6           221      ; A data header has been found. Read in the first $56
C0A6           222      ; nibbles as the index into the RDNIBLBT table, and save
C0A6           223      ; data in NBUF2BT.
C0A6           224      ;
C0A6           225      FNDDATA:
C0A6 A0 56      226      ldy #NBUF2SIZ
C0A8           227      ;
C0A8 84 3C      228      ^1      sty ROMTEMPZ
C0AA           229      ;
C0AA BC 8C C0   230      ^2      ldy STROBE,X
C0AD 10 FB      231      bpl <2
C0AF           232      ;
C0AF 59 D6 02   233      eor RDNIBLBT-$96,Y
C0B2           234      ;
C0B2 A4 3C      235      ldy ROMTEMPZ
C0B4           236      ;
C0B4 88         237      dey
C0B5           238      ;
C0B5 99 00 03   239      sta NBUF2BT,Y
C0B8           240      ;
C0B8 D0 EE      241      bne <1
C0BA           242      ;
C0BA           243      ;

```

```

C0BA      244 ; Read in next $100 nibbles as the index into the RDNIBLBT
C0BA      245 ; table, and save data at requested location, thus NBUF1 is
C0BA      246 ; not necessary.
C0BA      247 ;
C0BA 84 3C  248 ^3      sty ROMTEMPZ
C0BC      249 ;
C0BC BC 8C C0 250 ^4      ldy STROBE,X
C0BF 10 FB  251      bpl <4
C0C1      252 ;
C0C1 59 D6 02 253      eor RDNIBLBT-$96,Y
C0C4      254 ;
C0C4 A4 3C  255      ldy ROMTEMPZ
C0C6 91 26  256      sta (BUFRADRZ),Y
C0C8      257 ;
C0C8 C8     258      iny
C0C9 D0 EF  259      bne <3
C0CB      260 ;
C0CB      261 ;
C0CB      262 ; Finally read in checksum and test for zero.
C0CB      263 ;
C0CB BC 8C C0 264 ^5      ldy STROBE,X
C0CE 10 FB  265      bpl <5
C0D0      266 ;
C0D0 59 D6 02 267      eor RDNIBLBT-$96,Y
C0D3      268 ;
C0D3 D0 87  269 ^6      bne BOOTFW
C0D5      270 ;
C0D5      271 ;
C0D5      272 ; Post nibblize the data buffer with NBUF2BT.
C0D5      273 ;
C0D5 A0 00  274      ldy #ZERO
C0D7      275 ;
C0D7 A2 56  276 ^7      ldx #NBUF2SIZ
C0D9      277 ;
C0D9 CA     278 ^8      dex
C0DA 30 FB  279      bmi <7
C0DC      280 ;
C0DC B1 26  281      lda (BUFRADRZ),Y
C0DE      282 ;
C0DE 5E 00 03 283      lsr NBUF2BT,X
C0E1 2A     284      rol
C0E2      285 ;
C0E2 5E 00 03 286      lsr NBUF2BT,X
C0E5 2A     287      rol
C0E6      288 ;
C0E6 91 26  289      sta (BUFRADRZ),Y
C0E8      290 ;
C0E8 C8     291      iny
C0E9 D0 EE  292      bne <8
C0EB      293 ;
C0EB      294 ;
C0EB      295 ; Increment data buffer address and sector number.
C0EB      296 ;
C0EB E6 27  297      inc BUFRADRZ+1
C0ED E6 3D  298      inc ROMSECTR
C0EF      299 ;
C0EF      300 ;
C0EF      301 ; If sector number is less than 1 found at location $800
C0EF      302 ; restart looking for requested track and sector. For
C0EF      303 ; sector numbers greater than zero enter boot code at $801.
C0EF      304 ;

```

```
C0EF A5 3D      305      lda ROMSECTR
C0F1 CD 00 08    306      cmp PAGE08
C0F4             307      ;
C0F4 A6 2B      308      ldx SLOT16Z
C0F6             309      ;
C0F6 90 DB      310      bcc <6
C0F8             311      ;
C0F8 4C 01 08    312      jmp PAGE08+1
C0FB             313      ;
C0FB             314      dfs PAGESIZE--PAGESIZE**/PAGESIZE,ZERO
C100             315      ;
C100             316      ;
C100             317      icl "BUFR.L"

LLOAD BUFR.L,A$4000
```

```

C100      1          ttl "DOS 3.3 Source Code, BUFR.L"
C100      2      ;
C100      3      ;
C100      4      ; BUFR.L
C100      5      ;
C100      6      ;
9AA6      7          org DOSFBADR
9AA6      8          obj PAGE10
9AA6      9      ;
9AA6     10      ;
9AA6     11      DATABUFR dfs PAGESIZE,ZERO
9BA6     12      ;
9BA6     13      TSBUFFER dfs PAGESIZE,ZERO
9CA6     14      ;
9CA6     15      WORKAREA:
9CA6     16      TSFRSTTS dfs 2,ZERO          ; T/S of first T/S list
9CA8     17      TSCURRTS dfs 2,ZERO          ; T/S of current T/S list
9CAA     18      WAFLAGS  dfs 1,ZERO          ; 02 = VTOC has changed
9CAB     19      ;                          40 = data buffer has changed
9CAB     20      ;                          80 = T/S buffer has changed
9CAB     21      TSCURDAT dfs 2,ZERO          ; T/S of current data sector
9CAD     22      SECATOFF dfs 1,ZERO          ; sector offset into catalog
9CAE     23      BYCATOFF dfs 1,ZERO          ; byte offset into catalog
9CAF     24      MAXTSECR dfs 2,ZERO          ; maximum entries in T/S list
9CB1     25      SECFRSTS dfs 2,ZERO          ; offset of first T/S entry
9CB3     26      SECLASTS dfs 2,ZERO          ; offset of last T/S entry
9CB5     27      SECLSTRD dfs 2,ZERO          ; relative sector last read
9CB7     28      SECRSIZE dfs 2,ZERO          ; sector size in bytes
9CB9     29      SECRPOST dfs 2,ZERO          ; current position in sectors
9CBB     30      BYSECOFF dfs 1,ZERO          ; current sector byte offset
9CBC     31      dfs 1,ZERO                  ; not used
9CBD     32      RECDLN GH dfs 2,ZERO          ; fixed record length
9CBF     33      RECURNUM dfs 2,ZERO          ; current record number
9CC1     34      BYRECOFF dfs 2,ZERO          ; byte offset into record
9CC3     35      SECFILEN dfs 2,ZERO          ; length of file in sectors
9CC5     36      SECALOTR dfs 1,ZERO          ; next sector to get on track
9CC6     37      CURALOTR dfs 1,ZERO          ; current track to allocate
9CC7     38      SECFRETR dfs 4,ZERO          ; bit map free sectors on trk
9CCB     39      WAFILTYP dfs 1,ZERO          ; file type (^80=locked)
9CCC     40      WASLTNUM dfs 1,ZERO          ; slot number times 16
9CCD     41      WADRVNUM dfs 1,ZERO          ; drive number
9CCE     42      WAVOLNUM dfs 1,ZERO          ; volume number complemented
9CCF     43      WATRKNUM dfs 1,ZERO          ; track number
9CD0     44      dfs 3,ZERO                  ; not used
9CD3     45      ;
9CD3     46      FILNAMBF dfs NAMESIZE,ZERO    ; file name buffer
9CF1     47      ;
9CF1 A6 9C     48      WABUFADR adr WORKAREA    ; address of workarea buffer
9CF3     49      ;
9CF3 A6 9B     50      TSBUFADR adr TSBUFFER    ; address of T/S list buffer
9CF5     51      ;
9CF5 A6 9A     52      DABUFADR adr DATABUFR    ; address of data buffer
9CF7     53      ;
9CF7     54      NXTFNADR dfs 2,ZERO          ; address of next file name
9CF9     55      ;
9CF9     56      BUFREND:
9CF9     57      ;
9CF9     58      dfs 7,ZERO
9D00     59      ;
9D00     60      ;

```

9D00 61 icl "CMD1.L"

LLOAD CMD1.L,A\$4000

```
9D00      1          ttl "DOS 3.3 Source Code, CMD1.L"
9D00      2      ;
9D00      3      ;
9D00      4      ; CMD1.L
9D00      5      ;
9D00      6      ;
9D00      7          obj PAGE10
9D00      8          usr
9D00      9      ;
9D00     10      ;
9D00     11  DOSTART:
9D00 D3 9C     12  DBUFP      adr FILNAMBF      ; filename of first DOS buffer
9D02 81 9E     13  DOSKBD     adr KBDINTRC      ; keyboard intercept routine
9D04 BD 9E     14  DOSVID     adr VIDINTRC      ; video intercept routine
9D06 75 AA     15  PFNADR     adr FNAME        ; primary filename buffer
9D08 93 AA     16  SFNADR     adr SFNAME        ; secondary filename buffer
9D0A 60 AA     17  LDRNGLEN   adr LOADLEN       ; range length of LOAD command
9D0C 00 9D     18  LOADADR     adr DOSTART      ; start address of DOS
9D0E BB B5     19  FMPARMS    adr FMOPCOD       ; file Manager parameter list
9D10     20      ;
9D10     21      ;
9D10     22      ; CSWL state handler address table. States are used to
9D10     23      ; drive the handling of DOS commands as they appear as
9D10     24      ; output of PRINT statements, and this table contains the
9D10     25      ; address of the routine which handles each state from
9D10     26      ; state 0 to state 6.
9D10     27      ;
9D10     28  CSWSTADR:
9D10 EA 9E     29          adr CSWST0-1      ; start of line
9D12 11 9F     30          adr CSWST1-1      ; collect DOS command
9D14 22 9F     31          adr CSWST2-1      ; ignore non-DOS command
9D16 2E 9F     32          adr CSWST3-1      ; INPUT statement handler
9D18 51 9F     33          adr CSWST4-1      ; WRITE data to a file
9D1A 60 9F     34          adr CSWST5-1      ; start of WRITE data line
9D1C 70 9F     35          adr CSWST6-1      ; skip prompt character
9D1E     36      ;
9D1E     37      ;
9D1E     38      ; Command handler entry point table. This table contains
9D1E     39      ; the address of a command handler subroutine for each DOS
9D1E     40      ; command in the following standard order.
9D1E     41      ;
9D1E     42  CMDTBL:
9D1E 4E A5     43  CMDINIT     adr DOINIT-1
9D20 12 A4     44  CMDLOAD     adr DOLOAD-1
9D22 96 A3     45  CMDSAVE     adr DOSAVE-1
9D24 D0 A4     46  CMDRUN      adr DORUN-1
9D26 EF A4     47  CMDCHAIN    adr DOCHAIN-1
9D28 62 A2     48  CMDDELET     adr DODELETE-1
9D2A 70 A2     49  CMDLOCK     adr DOLOCK-1
9D2C 74 A2     50  CMDUNLCK    adr DOUNLOCK-1
9D2E E9 A2     51  CMDCLOSE    adr DOCLOSE-1
9D30 1A A5     52  CMDREAD     adr DOREAD-1
9D32 C5 A5     53  CMDEXEC     adr DOEXEC-1
9D34 0F A5     54  CMDWRITE    adr DOWRITE-1
9D36 DC A5     55  CMDPOSTN    adr DOPSTION-1
9D38 A2 A2     56  CMDOPEN     adr DOOPEN-1
9D3A 97 A2     57  CMDAPND     adr DOAPND-1
9D3C 80 A2     58  CMDRENAM    adr DORENAME-1
9D3E 6D A5     59  CMDCAT      adr DOCAT-1
9D40 32 A2     60  CMDMON      adr DOMON-1
```



```

9D42 3C A2      61  CMDNOMON adr DONOMON-1
9D44 28 A2      62  CMDPRNUM adr DOPRNUM-1
9D46 2D A2      63  CMDINNUM adr DOINNUM-1
9D48 50 A2      64  CMDMXFLS adr DOMXFLS-1
9D4A 79 A5      65  CMDFP      adr DOFP-1
9D4C 9D A5      66  CMDINT      adr DOINT-1
9D4E 30 A3      67  CMDBSAVE adr DOBSAVE-1
9D50 5C A3      68  CMDBLOAD adr DOBLOAD-1
9D52 8D A3      69  CMDBRUN  adr DOBRUN-1
9D54 7C A2      70  CMDVERFY adr DOVERIFY-1
9D56           71  ;
9D56           72  ;
9D56           73  ; Active BASIC entry point vector table. The addresses
9D56           74  ; stored here are maintained by DOS such that they apply
9D56           75  ; to the current version of BASIC running.
9D56           76  ;
9D56           77  .if INTEGER
9D56           78  ;
9D56           79  CHNADR      adr IRUN
9D56           80  RUNADR      adr DOSIRUN
9D56           81  BASERR      adr INTERR
9D56           82  BASCOLD     adr BASCLD
9D56           83  BASWARM     adr BASWRM
9D56           84  ASFTREL     adr ASROMRST
9D56           85  ;
9D56           86  .el
9D56           87  ;
9D56 FC A4      88  CHNADR      adr DOSARUN
9D58 FC A4      89  RUNADR      adr DOSARUN
9D5A 65 D8      90  BASERR      adr ASROMERR
9D5C 00 E0      91  BASCOLD     adr BASCLD
9D5E 3C D4      92  BASWARM     adr ASROMWRM
9D60 F2 D4      93  ASFTREL     adr ASROMRST
9D62           94  ;
9D62           95  .fi
9D62           96  ;
9D62           97  ;
9D62           98  ; Integer entry point vector table. This image is copied
9D62           99  ; to CHNADR if Integer is made active.
9D62          100  ;
9D62          101  INTTBL:
9D62 36 E8      102          adr IRUN
9D64 E5 A4      103          adr DOSIRUN
9D66 E3 E3      104          adr INTERR
9D68 00 E0      105          adr BASCLD
9D6A 03 E0      106          adr BASWRM
9D6C           107  ;
9D6C           108  ;
9D6C           109  ; ROM Applesoft entry point vector table.
9D6C           110  ;
9D6C           111  FPTBL:
9D6C FC A4      112          adr DOSARUN
9D6E FC A4      113          adr DOSARUN
9D70 65 D8      114          adr ASROMERR
9D72 00 E0      115          adr BASCLD
9D74 3C D4      116          adr ASROMWRM
9D76 F2 D4      117          adr ASROMRST
9D78           118  ;
9D78           119  ;
9D78           120  ; RAM Applesoft entry point vector table.
9D78           121  ;

```

```

9D78          122  RAMASTBL:
9D78 06 A5      123          adr DOSARUN2
9D7A 06 A5      124          adr DOSARUN2
9D7C 67 10      125          adr ASRAMERR
9D7E 84 9D      126          adr DOSSTRT
9D80 3C 0C      127          adr ASRAMWRM
9D82 F2 0C      128          adr ASRAMRST
9D84          129  ;
9D84          130  ;
9D84          131  ; DOS coldstart entry routine.  Get the default slot and
9D84          132  ; drive values and store as default values for command
9D84          133  ; keywords.
9D84          134  ;
9D84          135  DOSSTRT:
9D84 AD E9 B7    136          lda SNUM16
9D87          137  ;
9D87 4A          138          lsr
9D88 4A          139          lsr
9D89 4A          140          lsr
9D8A 4A          141          lsr
9D8B          142  ;
9D8B 8D 6A AA    143          sta SLOTVAL
9D8E          144  ;
9D8E AD EA B7    145          lda DNUM
9D91 8D 68 AA    146          sta DRVAL
9D94          147  ;
9D94          148  ;
9D94          149  ; See which BASIC is active.
9D94          150  ;
9D94 AD 00 E0    151          lda BASCLD
9D97 49 20      152          eor #INTBASIC
9D99 D0 11      153          bne >2
9D9B          154  ;
9D9B          155  ;
9D9B          156  ; Integer BASIC is active so move Integer addresses to the
9D9B          157  ; active BASIC table.
9D9B          158  ;
9D9B 8D B6 AA    159          sta WHCBASIC
9D9E          160  ;
9D9E A2 0A      161          ldx #FPTBL-INTTBL
9DA0          162  ;
9DA0 BD 61 9D    163  ^1      lda INTTBL-1,X
9DA3 9D 55 9D    164          sta CHNADR-1,X
9DA6          165  ;
9DA6 CA          166          dex
9DA7 D0 F7      167          bne <1
9DA9          168  ;
9DA9 4C BC 9D    169          jmp >4
9DAC          170  ;
9DAC          171  ;
9DAC          172  ; Applesoft BASIC is active so move Applesoft addresses to
9DAC          173  ; the active BASIC table.  On exit set C-flag for
9DAC          174  ;coldstart.
9DAC          175  ;
9DAC A9 40      176  ^2      lda #FPOACTV
9DAE 8D B6 AA    177          sta WHCBASIC
9DB1          178  ;
9DB1 A2 0C      179          ldx #RAMASTBL-FPTBL
9DB3          180  ;
9DB3 BD 6B 9D    181  ^3      lda FPTBL-1,X
9DB6 9D 55 9D    182          sta CHNADR-1,X

```

```

9DB9          183 ;
9DB9 CA       184 dex
9DBA D0 F7    185 bne <3
9DBC          186 ;
9DBC 38       187 ^4 sec
9DBD B0 12    188 bcs INITDOS
9DBF          189 ;
9DBF          190 ;
9DBF          191 ; DOS warmstart entry routine.
9DBF          192 ;
9DBF          193 DOSWARM:
9DBF AD B6 AA 194 lda WHCBASIC
9DC2 D0 04    195 bne >1
9DC4          196 ;
9DC4          197 ;
9DC4          198 ; Select Integer BASIC.
9DC4          199 ;
9DC4 A9 20    200 lda #INTBASIC
9DC6 D0 05    201 bne >2
9DC8          202 ;
9DC8 0A       203 ^1 asl
9DC9 10 05    204 bpl >3
9DCB          205 ;
9DCB          206 ;
9DCB          207 ; Select Applesoft BASIC.
9DCB          208 ;
9DCB A9 4C    209 lda #FPBASIC
9DCD          210 ;
9DCD 20 B2 A5 211 ^2 jsr SETBASIC
9DD0          212 ;
9DD0 18       213 ^3 clc
9DD1          214 ;
9DD1          215 ;
9DD1          216 ; Initialize DOS to replace the DOS keyboard and video
9DD1          217 ; intercepts. On entry C-flag = 0 for warmstart, C-flag =
9DD1          218 ; 1 for coldstart.
9DD1          219 ;
9DD1          220 INITDOS:
9DD1 08       221 php
9DD2          222 ;
9DD2 20 51 A8 223 jsr INITPTRS
9DD5          224 ;
9DD5          225 ;
9DD5          226 ; Set NOMON O,I,C and initial state to 0.
9DD5          227 ;
9DD5 A9 00    228 lda #ZERO
9DD7 8D 5E AA 229 sta MONFLGS
9DDA 8D 52 AA 230 sta CSWSTATE
9DDD          231 ;
9DDD          232 ;
9DDD          233 ; Recall whether this is a warmstart or coldstart.
9DDD          234 ;
9DDD 28       235 plp
9DDE 6A       236 ror
9DDF 8D 51 AA 237 sta CURSTAT
9DE2          238 ;
9DE2 30 03    239 bmi >1
9DE4          240 ;
9DE4 6C 5E 9D 241 jmp (BASWARM)
9DE7          242 ;
9DE7 6C 5C 9D 243 ^1 jmp (BASCOLD)

```

```

9DEA      244 ;
9DEA      245 ;
9DEA      246 ; FRSTIME is called after a DOS coldstart. It checks for
9DEA      247 ; RAM Applesoft and takes care of other coldstart tasks
9DEA      248 ; that must be done after BASIC is initialized.
9DEA      249 ;
9DEA      250 FRSTIME:
9DEA 0A      251      asl
9DEB 10 19    252      bpl >2
9DED      253 ;
9DED      254 ;
9DED      255 ; If RAM Applesoft, move required pointers into the active
9DED      256 ; BASIC area and blank out primary filename buffer.
9DED      257 ;
9DED 8D B6 AA 258      sta WHCBASIC
9DF0      259 ;
9DF0 A2 0C    260      ldx #DOSSTRT-RAMASTBL
9DF2      261 ;
9DF2 BD 77 9D 262 ^0      lda RAMASTBL-1,X
9DF5 9D 55 9D 263      sta CHNADR-1,X
9DF8      264 ;
9DF8 CA      265      dex
9DF9 D0 F7    266      bne <0
9DFB      267 ;
9DFB A2 1D    268      ldx #NAME SIZE-1
9DFD      269 ;
9DFD BD 93 AA 270 ^1      lda SFNAME,X
9E00 9D 75 AA 271      sta FNAME,X
9E03      272 ;
9E03 CA      273      dex
9E04 10 F7    274      bpl <1
9E06      275 ;
9E06      276 ;
9E06      277 ; Initialize MAXFILES to the default value of 3.
9E06      278 ;
9E06 AD B1 AA 279 ^2      lda MXFLS
9E09 8D 57 AA 280      sta MAXFILES
9E0C      281 ;
9E0C      282 ;
9E0C      283 ; Set up the DOS file buffers chain.
9E0C      284 ;
9E0C 20 D4 A7 285      jsr INITBUFS
9E0F      286 ;
9E0F      287 ;
9E0F      288 ; If EXEC is active make it inactive on the next character
9E0F      289 ; input.
9E0F      290 ;
9E0F AD B3 AA 291      lda EXFLG
9E12 F0 09    292      beq >3
9E14      293 ;
9E14 48      294      pha
9E15      295 ;
9E15 20 9D A6 296      jsr PNTEXEC
9E18      297 ;
9E18 68      298      pla
9E19      299 ;
9E19 A0 00    300      ldy #ZERO
9E1B 91 40    301      sta (FILEBUFZ),Y
9E1D      302 ;
9E1D      303 ;
9E1D      304 ; Reset the CSWL state and warmstart status to zero.

```

```

9E1D      305 ;
9E1D 20 5B A7 306 ^3      jsr RSET0
9E20      307 ;
9E20      308 ;
9E20      309 ; Check to see if we are in the middle of a command.
9E20      310 ;
9E20 AD 5F AA 311      lda CMDINDX
9E23 D0 20   312      bne TSTPEND
9E25      313 ;
9E25      314 ;
9E25      315 ; If not, copy the page 3 code to $3D0-$3FF.
9E25      316 ;
9E25 A2 2F   317      ldx #PAG3END-PAG3BGN+1
9E27      318 ;
9E27 BD 51 9E 319 ^1      lda PAG3CODE,X
9E2A 9D D0 03 320      sta DOSRST,X
9E2D      321 ;
9E2D CA     322      dex
9E2E 10 F7   323      bpl <1
9E30      324 ;
9E30 AD 53 9E 325      lda PAG3CODE+2
9E33 8D F3 03 326      sta SOFTEV+1
9E36      327 ;
9E36 49 A5   328      eor #PWRUPBYT
9E38 8D F4 03 329      sta PWREDUP
9E3B      330 ;
9E3B AD 52 9E 331      lda PAG3CODE+1
9E3E 8D F2 03 332      sta SOFTEV
9E41      333 ;
9E41      334 ;
9E41      335 ; To BRUN a file at boot use #CMDBRUN-CMDTBL ($34).
9E41      336 ; To EXEC a file at boot use #CMDEXEC-CMDTBL ($14).
9E41      337 ;
9E41 A9 06   338      lda #CMDBRUN-CMDTBL
9E43 D0 05   339      bne >1
9E45      340 ;
9E45      341 ;
9E45      342 ; If there is a command pending because RAM Applesoft may
9E45      343 ; be loaded, then execute it, otherwise return to BASIC.
9E45      344 ;
9E45      345 TSTPEND:
9E45 AD 62 AA 346      lda PENDCMD
9E48 F0 06   347      beq >2
9E4A      348 ;
9E4A 8D 5F AA 349 ^1      sta CMDINDX
9E4D      350 ;
9E4D 4C 80 A1 351      jmp DOCMD
9E50      352 ;
9E50 60     353 ^2      rts
9E51      354 ;
9E51      355 ;
9E51      356 PAG3CODE:
9E51      357 ;
9E51      358 ; Code that gets moved down to $3D0.
9E51      359 ;
9E51      360      phs $3D0
03D0      361 ;
03D0      362 PAG3BGN:
03D0      363 ;
03D0      364      .if INTEGER
03D0      365 ;

```

```

03D0          366          jmp DOSWARM          ; DOS warmstart
03D0          367          jmp DOSSTRT          ; DOS coldstart
03D0          368          jmp FMEXT            ; File Manager entry
03D0          369          jmp CALLRWTS         ; RWTS entry
03D0          370          ;
03D0          371          lda FMPARMS+1        ; FM parameter buffer address
03D0          372          ldy FMPARMS
03D0          373          rts
03D0          374          ;
03D0          375          lda RWTSPADR+1        ; RWTS parameter buffer addr.
03D0          376          ldy RWTSPADR
03D0          377          rts
03D0          378          ;
03D0          379          jmp INITPTRS         ; DOS reconnect entry
03D0          380          ;
03D0          381          nop
03D0          382          nop
03D0          383          ;
03D0          384          jmp OLDBRK           ; Autostart ROM BRK handler
03D0          385          jmp MON              ;
03D0          386          jmp IORTS           ; & handler
03D0          387          jmp MON             ; ctrl-Y handler
03D0          388          jmp MON             ; non-maskable IRQ handler
03D0          389          ;
03D0          390          adr MON              ; maskable IRQ handler address
03D0          391          ;
03D0          392          .el
03D0          393          ;
03D0 4C BF 9D 394          jmp DOSWARM          ; DOS warmstart
03D3 4C 84 9D 395          jmp DOSSTRT          ; DOS coldstart
03D6 4C FD AA 396          jmp FMEXT            ; File Manager entry
03D9 4C B5 B7 397          jmp CALLRWTS         ; RWTS entry
03DC          398          ;
03DC AD 0F 9D 399          lda FMPARMS+1        ; FM parameter buffer address
03DF AC 0E 9D 400          ldy FMPARMS
03E2 60 401          rts
03E3          402          ;
03E3 AD C2 AA 403          lda RWTSPADR+1        ; RWTS parameter buffer addr.
03E6 AC C1 AA 404          ldy RWTSPADR
03E9 60 405          rts
03EA          406          ;
03EA 4C 51 A8 407          jmp INITPTRS         ; DOS reconnect entry
03ED          408          ;
03ED EA 409          nop
03EE EA 410          nop
03EF          411          ;
03EF 4C 59 FA 412          jmp OLDBRK           ; Autostart ROM BRK handler
03F2          413          ;
03F2 BF 9D 414          adr DOSWARM             ; reset address for auto
03F4          415          ;
03F4 38 416          byt PWRUPBYT^DOSTART/PAGESIZE ; power up byte
03F5          417          ;
03F5 4C 58 FF 418          jmp IORTS           ; & handler
03F8 4C 65 FF 419          jmp MON             ; ctrl-Y handler
03FB 4C 65 FF 420          jmp MON             ; non-maskable IRQ handler
03FE          421          ;
03FE 65 FF 422          adr MON              ; maskable IRQ handler address
0400          423          ;
0400          424          .fi
0400          425          ;
0400          426          PAG3END:

```

```
0400          427 ;
0400          428 ;
0400          429 ; Return to DOS image.
0400          430 ;
0400          431           phs PAG3CODE+PAG3END-PAG3BGN
9E81          432 ;
9E81          433 ;
9E81          434 ; Keyboard intercept routine is called each time a
9E81          435 ; character is requested. Locations $38 and $39 point to
9E81          436 ; this routine. If CURSTAT = 0 then warmstart.
9E81          437 ;
9E81          438 KBDINTRC:
9E81 20 D1 9E 439           jsr SAVREG
9E84          440 ;
9E84 AD 51 AA 441           lda CURSTAT
9E87 F0 15    442           beq >2
9E89          443 ;
9E89          444 ;
9E89          445 ; Reading from a file, so remove cursor from screen. Check
9E89          446 ; for coldstart.
9E89          447 ;
9E89 48       448           pha
9E8A          449 ;
9E8A AD 5C AA 450           lda ASAVE
9E8D 91 28    451           sta (BASEZ),Y
9E8F          452 ;
9E8F 68       453           pla
9E90 30 03    454           bmi >1
9E92          455 ;
9E92 4C 26 A6 456           jmp READBYTE
9E95          457 ;
9E95          458 ;
9E95          459 ; Special case for coldstart if EXECing a file.
9E95          460 ;
9E95 20 EA 9D 461 ^1       jsr FRSTIME
9E98          462 ;
9E98 A4 24    463           ldy CH
9E9A          464 ;
9E9A A9 60    465           lda #FLASHSPC
9E9C 91 28    466           sta (BASEZ),Y
9E9E          467 ;
9E9E AD B3 AA 468 ^2       lda EXFLG
9EA1 F0 03    469           beq >3
9EA3          470 ;
9EA3 20 82 A6 471           jsr EXECD
9EA6          472 ;
9EA6 A9 03    473 ^3       lda #STATE3
9EA8 8D 52 AA 474           sta CSWSTATE
9EAB          475 ;
9EAB 20 BA 9F 476           jsr REGRST
9EAE 20 BA 9E 477           jsr RDCHAR
9EB1          478 ;
9EB1 8D 5C AA 479           sta ASAVE
9EB4 8E 5A AA 480           stx XSAVE
9EB7          481 ;
9EB7 4C B3 9F 482           jmp DOSXIT
9EBA          483 ;
9EBA 6C 38 00 484 RDCHAR   jmp (KSWL)
9EBD          485 ;
9EBD          486 ;
9EBD          487 ; Video intercept routine. Video input state handler
```

```

9EBD          488 ; transfers control to the proper output handler.
9EBD          489 ;
9EBD          490 VIDINTRC:
9EBD 20 D1 9E 491      jsr SAVREG
9EC0          492 ;
9EC0 AD 52 AA 493      lda CSWSTATE
9EC3 0A       494      asl
9EC4 AA       495      tax
9EC5          496 ;
9EC5 BD 11 9D 497      lda CSWSTADR+1,X
9EC8 48       498      pha
9EC9          499 ;
9EC9 BD 10 9D 500      lda CSWSTADR,X
9ECC 48       501      pha
9ECD          502 ;
9ECD AD 5C AA 503      lda ASAVE
9ED0          504 ;
9ED0 60       505      rts
9ED1          506 ;
9ED1          507 ;
9ED1          508 ; Common register, CSWL, and KSWL save routine.
9ED1          509 ;
9ED1          510 SAVREG:
9ED1 8D 5C AA 511      sta ASAVE
9ED4 8E 5A AA 512      stx XSAVE
9ED7 8C 5B AA 513      sty YSAVE
9EDA          514 ;
9EDA BA       515      tsx
9EDB E8       516      inx
9EDC E8       517      inx
9EDD 8E 59 AA 518      stx SSAVE
9EE0          519 ;
9EE0 A2 03     520      ldx #3
9EE2          521 ;
9EE2 BD 53 AA 522 ^1    lda CSWLSAV,X
9EE5 95 36     523      sta CSWL,X
9EE7          524 ;
9EE7 CA       525      dex
9EE8 10 F8     526      bpl <1
9EEA          527 ;
9EEA 60       528      rts
9EEB          529 ;
9EEB          530 ;
9EEB          531 ; CSWL state 0 handler: start of line. Check for
9EEB          532 ; control-D at beginning of line.
9EEB          533 ;
9EEB          534 ; If RUN command was delayed while loading Applesoft,
9EEB          535 ; continue RUN command.
9EEB          536 ;
9EEB          537 CSWST0:
9EEB AE B7 AA 538      ldx RUNINTRC
9EEE F0 03     539      beq >1
9EF0          540 ;
9EF0 4C 78 9F 541      jmp RUNDONE
9EF3          542 ;
9EF3          543 ;
9EF3          544 ; Check to see if a file is being read.
9EF3          545 ;
9EF3 AE 51 AA 546 ^1    ldx CURSTAT
9EF6 F0 08     547      beq >2
9EF8          548 ;

```



```

9EF8      549      ;
9EF8      550      ; If a file is being read, check to see if the Applesoft
9EF8      551      ; "?" prompt was output.
9EF8      552      ;
9EF8 C9 BF      553      cmp #"?"
9EFA F0 75      554      beq CSWST6
9EFC      555      ;
9EFC      556      ;
9EFC      557      ; See if at the beginning of line.
9EFC      558      ;
9EFC C5 33      559      cmp PROMPT
9EFE F0 27      560      beq CSWST2.1
9F00      561      ;
9F00      562      ;
9F00      563      ; If no control-D at beginning of line, go into state 2.
9F00      564      ; Otherwise go to state 1 and collect possible DOS command.
9F00      565      ;
9F00 A2 02      566      ^2      ldx #STATE2
9F02 8E 52 AA    567      stx CSWSTATE
9F05      568      ;
9F05 CD B2 AA    569      cmp CTLD
9F08 D0 19      570      bne CSWST2
9F0A      571      ;
9F0A CA          572      dex
9F0B 8E 52 AA    573      stx CSWSTATE
9F0E      574      ;
9F0E CA          575      dex
9F0F 8E 5D AA    576      stx CMDLNIDX
9F12      577      ;
9F12      578      ;
9F12      579      ; CSWL state 1 handler: collect DOS command from input
9F12      580      ; line and parse.
9F12      581      ;
9F12 AE 5D AA    582      CSWST1      ldx CMDLNIDX
9F15      583      ;
9F15 9D 00 02    584      CSWST1.1 sta INPUT,X
9F18      585      ;
9F18 E8          586      inx
9F19 8E 5D AA    587      stx CMDLNIDX
9F1C      588      ;
9F1C C9 8D      589      cmp #RETURN
9F1E D0 75      590      bne ECHOC
9F20      591      ;
9F20 4C CD 9F    592      jmp PARSE
9F23      593      ;
9F23      594      ;
9F23      595      ; CSWL state 2 handler: ignore non-DOS command. If
9F23      596      ; current input character isn't a return, exit DOS through
9F23      597      ; ECHO, otherwise reset command state to 0.
9F23      598      ;
9F23 C9 8D      599      CSWST2      cmp #RETURN
9F25 D0 7D      600      bne ECHO
9F27      601      ;
9F27 A2 00      602      CSWST2.1 ldx #STATE0
9F29 8E 52 AA    603      stx CSWSTATE
9F2C      604      ;
9F2C 4C A4 9F    605      jmp ECHO
9F2F      606      ;
9F2F      607      ;
9F2F      608      ; CSWL state 3 handler: INPUT statement handler. Set the
9F2F      609      ; command state to zero as a default in case input ends.

```

```

9F2F          610 ;
9F2F          611 CSWST3:
9F2F A2 00     612         ldx #STATE0
9F31 8E 52 AA  613         stx CSWSTATE
9F34          614 ;
9F34 C9 8D     615         cmp #RETURN
9F36 F0 07     616         beq >2
9F38          617 ;
9F38          618 ;
9F38          619 ; If not at end of input, check the exec flag. If set,
9F38          620 ; echo character only if MON I is set or exec is active.
9F38          621 ; In either case the state will be reset to 3.
9F38          622 ;
9F38 AD B3 AA  623 ^1         lda EXFLG
9F3B F0 67     624         beq ECHO
9F3D          625 ;
9F3D D0 5E     626         bne ECHOI
9F3F          627 ;
9F3F          628 ;
9F3F          629 ; Come here if EOLN encountered. If BASIC is not running,
9F3F          630 ; or EXECing a file go to state 1 and check for a possible
9F3F          631 ; DOS command. Otherwise exit DOS and echo character as
9F3F          632 ; appropriate after switching to state 1.
9F3F          633 ;
9F3F 48        634 ^2         pha
9F40          635 ;
9F40 38        636         sec
9F41          637 ;
9F41 AD B3 AA  638         lda EXFLG
9F44 D0 03     639         bne >3
9F46          640 ;
9F46          641 ;
9F46          642 ; On exit if C-flag = 0 then running.
9F46          643 ;
9F46 20 5E A6  644         jsr ISBASRUN
9F49          645 ;
9F49 68        646 ^3         pla
9F4A 90 EC     647         bcc <1
9F4C          648 ;
9F4C AE 5A AA  649         ldx XSAVE
9F4F 4C 15 9F  650         jmp CSWST1.1
9F52          651 ;
9F52          652 ;
9F52          653 ; CSWL state 4 handler: WRITE data to a file. Check for
9F52          654 ; EOLN and switch to state 5 if <cr> encountered.
9F52          655 ;
9F52          656 CSWST4:
9F52 C9 8D     657         cmp #RETURN
9F54 D0 05     658         bne CSWST4.1
9F56          659 ;
9F56 A9 05     660         lda #STATE5
9F58 8D 52 AA  661         sta CSWSTATE
9F5B          662 ;
9F5B 20 0E A6  663 CSWST4.1 jsr WRITBYTE
9F5E          664 ;
9F5E 4C 99 9F  665         jmp ECHOO
9F61          666 ;
9F61          667 ;
9F61          668 ; CSWL state 5 handler: start of WRITE data line. Check
9F61          669 ; for CTL-D.
9F61          670 ;

```

```

9F61      671  CSWST5:
9F61 CD B2 AA 672      cmp CTLD
9F64 F0 85    673      beq CSWST0
9F66      674      ;
9F66      675      ;
9F66      676      ; Treat any linefeeds as a continuation of the previous
9F66      677      ; line.
9F66      678      ;
9F66 C9 8A    679      cmp #LINEFEED
9F68 F0 F1    680      beq CSWST4.1
9F6A      681      ;
9F6A      682      ;
9F6A      683      ; Otherwise revert to state 4.
9F6A      684      ;
9F6A A2 04    685      ldx #STATE4
9F6C 8E 52 AA 686      stx CSWSTATE
9F6F D0 E1    687      bne CSWST4
9F71      688      ;
9F71      689      ;
9F71      690      ; CSWL state 6 handler: skip output of prompt character.
9F71      691      ;
9F71      692  CSWST6:
9F71 A9 00    693      lda #STATE0
9F73 8D 52 AA 694      sta CSWSTATE
9F76 F0 25    695      beq ECHOI
9F78      696      ;
9F78      697      ;
9F78      698      ; Finish RUN command interrupted by RAM Applesoft load.
9F78      699      ;
9F78      700  RUNDONE:
9F78 A9 00    701      lda #ZERO
9F7A 8D B7 AA 702      sta RUNINTRC
9F7D      703      ;
9F7D 20 51 A8 704      jsr INITPTRS
9F80      705      ;
9F80 4C DC A4 706      jmp DORUN2
9F83      707      ;
9F83      708      ;
9F83      709      ; End of DOS command scan. Remove DOS command and pretend
9F83      710      ; the user only pressed return.
9F83      711      ;
9F83      712  SCNXIT:
9F83 AD 00 02 713      lda INPUT
9F86 CD B2 AA 714      cmp CTLD
9F89 F0 0A    715      beq ECHOC
9F8B      716      ;
9F8B A9 8D    717      lda #RETURN
9F8D 8D 00 02 718      sta INPUT
9F90      719      ;
9F90 A2 00    720      ldx #ZERO
9F92 8E 5A AA 721      stx XSAVE
9F95      722      ;
9F95      723      ;
9F95      724      ; Echo character conditionally to the screen:
9F95      725      ;
9F95      726      ; ECHOC - IF MON C
9F95      727      ; ECHOO - IF MON O
9F95      728      ; ECHOI - IF MON I
9F95      729      ;
9F95      730      ; ECHO - UNCONDITIONALLY
9F95      731      ;

```

```

9F95          732  ECHOC:
9F95 A9 40      733          lda #MONCMASK
9F97 D0 06      734          bne >1
9F99          735  ;
9F99          736  ECHOO:
9F99 A9 10      737          lda #MONOMASK
9F9B D0 02      738          bne >1
9F9D          739  ;
9F9D          740  ECHOI:
9F9D A9 20      741          lda #MONIMASK
9F9F          742  ;
9F9F 2D 5E AA   743  ^1      and MONFLGS
9FA2 F0 0F      744          beq DOSXIT
9FA4          745  ;
9FA4          746  ECHO:
9FA4 20 BA 9F   747          jsr REGRST
9FA7 20 C5 9F   748          jsr CMDCOUT
9FAA          749  ;
9FAA 8D 5C AA   750          sta ASAVE
9FAD 8C 5B AA   751          sty YSAVE
9FB0 8E 5A AA   752          stx XSAVE
9FB3          753  ;
9FB3          754  ;
9FB3          755  ; Restore registers and exit DOS.
9FB3          756  ;
9FB3          757  DOSXIT:
9FB3 20 51 A8   758          jsr INITPTRS
9FB6          759  ;
9FB6 AE 59 AA   760          ldx SSAVE
9FB9 9A         761          txs
9FBA          762  ;
9FBA          763  REGRST:
9FBA AD 5C AA   764          lda ASAVE
9FBD AC 5B AA   765          ldy YSAVE
9FC0 AE 5A AA   766          ldx XSAVE
9FC3          767  ;
9FC3 38         768          sec
9FC4 60         769          rts
9FC5          770  ;
9FC5          771  ;
9FC5          772  ; COUT and CRLF routines.
9FC5          773  ;
9FC5          774  CMDCOUT:
9FC5 6C 36 00   775          jmp (CSWL)
9FC8          776  ;
9FC8          777  CRLF:
9FC8 A9 8D      778          lda #RETURN
9FCA 4C C5 9F   779          jmp CMDCOUT
9FCD          780  ;
9FCD          781  ;
9FCD          782  ; Parse DOS command and check syntax. Set the command
9FCD          783  ; index to -1 (none).
9FCD          784  ;
9FCD          785  PARSE:
9FCD A0 FF      786          ldy #NEGONE
9FCF 8C 5F AA   787          sty CMDINDX
9FD2          788  ;
9FD2 C8         789          iny
9FD3 8C 62 AA   790          sty PENDCMD
9FD6          791  ;
9FD6          792  ;

```

```

9FD6      793      ; Get successive characters from input buffer and compare
9FD6      794      ; against current command in table. Skip ctrl-D's and
9FD6      795      ; blanks. Compare character in table using EOR.
9FD6      796      ;
9FD6 EE 5F AA    797      ^1      inc CMDINDX
9FD9      798      ;
9FD9 A2 00      799      ldx #ZERO
9FDB 08      800      php                      ; assume equal
9FDC      801      ;
9FDC BD 00 02    802      lda INPUT,X
9FDF CD B2 AA    803      cmp CTLD
9FE2 D0 01      804      bne >2
9FE4      805      ;
9FE4 E8      806      inx
9FE5      807      ;
9FE5 8E 5D AA    808      ^2      stx CMDLNIDX
9FE8      809      ;
9FE8 20 A4 A1    810      ^3      jsr FLSHCMDL
9FEB      811      ;
9FEB 29 7F      812      and #ASCIMASK
9FED 59 84 A8    813      eor DOSCMDS,Y
9FF0      814      ;
9FF0 C8      815      iny
9FF1      816      ;
9FF1      817      ;
9FF1      818      ; The following code is a clever way of determining if the
9FF1      819      ; entire string is equal to one of the DOS commands.
9FF1      820      ;
9FF1 0A      821      asl
9FF2 F0 02      822      beq >4
9FF4      823      ;
9FF4 68      824      pla
9FF5 08      825      php
9FF6      826      ;
9FF6 90 F0      827      ^4      bcc <3
9FF8      828      ;
9FF8 28      829      plp
9FF9 F0 20      830      beq CMPUTIDX
9FFB      831      ;
9FFB B9 84 A8    832      lda DOSCMDS,Y
9FFE D0 D6      833      bne <1
A000      834      ;
A000      835      ;
A000      836      ; Command was not found, so see if first character on the
A000      837      ; line was control-D; if so SYNTAX ERROR!!!
A000      838      ;
A000      839      GETFRST:
A000 AD 00 02    840      lda INPUT
A003 CD B2 AA    841      cmp CTLD
A006 F0 03      842      beq >1
A008      843      ;
A008 4C A4 9F    844      jmp ECHO
A00B      845      ;
A00B AD 01 02    846      ^1      lda INPUT+1
A00E C9 8D      847      cmp #RETURN
A010 D0 06      848      bne >2
A012      849      ;
A012 20 5B A7    850      jsr RSET0
A015      851      ;
A015 4C 95 9F    852      jmp ECHOC
A018      853      ;

```

```
A018 4C C4 A6      854 ^2      jmp CSYNERR
A01B              855 ;
A01B              856 ;
A01B              857 ; Valid DOS command, get its address and execute it.
A01B              858 ;
A01B              859 CMPUTIDX:
A01B 0E 5F AA      860      asl CMDINDX
A01E              861 ;
A01E AC 5F AA      862      ldY CMDINDX
A021              863 ;
A021              864 ;
A021              865 ; On exit if C-flag = 0 then running.
A021              866 ;
A021 20 5E A6      867      jsr ISBASRUN
A024 90 0C          868      bcc >1
A026              869 ;
A026              870 ;
A026              871 ; Certain commands such as OPEN can only be executed if
A026              872 ; BASIC is running. If such a command is executed and
A026              873 ; BASIC is not running, fall through to here.
A026              874 ;
A026 A9 02          875      lda #PGCMDMSK
A028 39 09 A9      876      and KWRDPRMS,Y
A02B              877 ;
A02B F0 05          878      beq >1
A02D              879 ;
A02D              880 ; Print NOT A DIRECT COMMAND error message.
A02D              881 ;
A02D A9 0F          882      lda #EMSOFF15-EMSGOFFS
A02F 4C D2 A6      883      jmp DOERROR
A032              884 ;
A032              885 ;
A032              886 ; Is this the RUN command?
A032              887 ;
A032 C0 06          888 ^1      cpy #6
A034 D0 02          889      bne >1
A036              890 ;
A036 84 33          891      sty PROMPT
A038              892 ;
A038              893 ;
A038              894 ; See if a filename is required.
A038              895 ;
A038 A9 20          896 ^1      lda #FLNM1MSK
A03A 39 09 A9      897      and KWRDPRMS,Y
A03D F0 61          898      beq FNDNONAM
A03F              899 ;
A03F              900 ;
A03F              901 ; If a filename is needed, blank out the filename arrays
A03F              902 ; and collect a filename if found.
A03F              903 ;
A03F 20 95 A0      904      jsr BLKNAMES
A042              905 ;
A042 08            906      php
A043              907 ;
A043              908 ;
A043              909 ; Skip any leading blanks.
A043              910 ;
A043 20 A4 A1      911 ^1      jsr FLSHCMDL
A046 F0 1E          912      beq >6
A048              913 ;
A048 0A            914      asl
```

```

A049 90 05      915      bcc >2
A04B           916      ;
A04B 30 03      917      bmi >2
A04D           918      ;
A04D 4C 00 A0   919      jmp GETFRST
A050           920      ;
A050 6A         921      ^2      ror
A051 4C 59 A0   922      jmp >4
A054           923      ;
A054 20 93 A1   924      ^3      jsr GNXTCHR
A057 F0 0D      925      beq >6
A059           926      ;
A059 99 75 AA   927      ^4      sta FNAME,Y
A05C           928      ;
A05C C8         929      iny
A05D C0 3C      930      cpy #NAME SIZE*2
A05F 90 F3      931      bcc <3
A061           932      ;
A061 20 93 A1   933      ^5      jsr GNXTCHR
A064 D0 FB      934      bne <5
A066           935      ;
A066 28         936      ^6      plp
A067 D0 0F      937      bne >7
A069           938      ;
A069           939      ;
A069           940      ; Check if a second file name is required.
A069           941      ;
A069 AC 5F AA   942      ldy CMDINDX
A06C           943      ;
A06C A9 10      944      lda #FLNM2MSK
A06E 39 09 A9   945      and KWRDPRMS,Y
A071 F0 0C      946      beq >8
A073           947      ;
A073 A0 1E      948      ldy #NAME SIZE
A075 08         949      php
A076 D0 CB      950      bne <1
A078           951      ;
A078 AD 93 AA   952      ^7      lda SFNAME
A07B C9 A0      953      cmp #SPACE
A07D F0 13      954      beq CHKFRST
A07F           955      ;
A07F AD 75 AA   956      ^8      lda FNAME
A082 C9 A0      957      cmp #SPACE
A084 D0 4B      958      bne SETDFLTS
A086           959      ;
A086           960      ;
A086           961      ; Check if a filename is optional for a command.
A086           962      ;
A086 AC 5F AA   963      ldy CMDINDX
A089           964      ;
A089 A9 C0      965      lda #FNOPMSK|CMDOPMSK
A08B 39 09 A9   966      and KWRDPRMS,Y
A08E F0 02      967      beq CHKFRST
A090           968      ;
A090 10 3F      969      bpl SETDFLTS
A092           970      ;
A092 4C 00 A0   971      CHKFRST jmp GETFRST
A095           972      ;
A095           973      ;
A095           974      ; Subroutine to blank out both filename buffers.
A095           975      ;

```

```
A095          976  BLKNAMES:
A095 A0 3C      977          ldy #NAME SIZE*2
A097          978  ;
A097 A9 A0     979  BLKNAME2 lda #SPACE
A099          980  ;
A099 99 74 AA  981  ^1      sta FNAME-1,Y
A09C          982  ;
A09C 88        983          dey
A09D D0 FA     984          bne <1
A09F          985  ;
A09F 60        986          rts
A0A0          987  ;
A0A0          988  ;
A0A0          989          icl "CMD2.L"
```

```
LLOAD CMD2.L,A$4000
```



```

A0A0          1          ttl "DOS 3.3 Source Code, CMD2.L"
A0A0          2          ;
A0A0          3          ;
A0A0          4          ; CMD2.L
A0A0          5          ;
A0A0          6          ;
A0A0          7          ; Branch here if no filename is required.
A0A0          8          ;
A0A0 8D 75 AA    9  FNDNONAM sta FNAME
A0A3          10         ;
A0A3          11         ; Check parameter table to see if a positional parameter
A0A3          12         ; is required.
A0A3          13         ;
A0A3 A9 0C      14         lda #MXFLEMSK|SLNUMMSK
A0A5 39 09 A9    15         and KWRDPRMS,Y
A0A8 F0 27      16         beq SETDFLTS
A0AA          17         ;
A0AA          18         ;
A0AA          19         ; Yes, so read in the number.  X-reg holds OPRND and A-reg
A0AA          20         ; holds OPRND+1.
A0AA          21         ;
A0AA 20 B9 A1    22         jsr GETNUM
A0AD          23         ;
A0AD          24         ;
A0AD          25         ; If the number was omitted branch on carry set.
A0AD          26         ;
A0AD B0 1F      27         bcs >9
A0AF          28         ;
A0AF          29         ;
A0AF          30         ; Branch if anything in A-reg.
A0AF          31         ;
A0AF A8          32         tay
A0B0 D0 17      33         bne >8
A0B2          34         ;
A0B2          35         ;
A0B2          36         ; Check the range of the number.  Branch on anything
A0B2          37         ; greater than 16.
A0B2          38         ;
A0B2 E0 11      39         cpx #17
A0B4 B0 13      40         bcs >8
A0B6          41         ;
A0B6 AC 5F AA    42         ldy CMDINDX
A0B9          43         ;
A0B9 A9 08      44         lda #SLNUMMSK
A0BB 39 09 A9    45         and KWRDPRMS,Y
A0BE F0 06      46         beq >7
A0C0          47         ;
A0C0          48         ;
A0C0          49         ; Check if slot number within range.  Branch if greater
A0C0          50         ; than 7.
A0C0          51         ;
A0C0 E0 08      52         cpx #8
A0C2 B0 CE      53         bcs CHKFRST
A0C4          54         ;
A0C4          55         ;
A0C4          56         ; Slot number is within range, set defaults.
A0C4          57         ;
A0C4 90 0B      58         bcc SETDFLTS
A0C6          59         ;
A0C6          60         ;

```

```
A0C6      61 ; MAXFILES number is within range, so set defaults if not
A0C6      62 ; zero.
A0C6      63 ;
A0C6 8A    64 ^7      txa
A0C7 D0 08 65      bne SETDFLTS
A0C9      66 ;
A0C9      67 ;
A0C9      68 ; Print RANGE ERROR error message.
A0C9      69 ;
A0C9 A9 02 70 ^8      lda #EMSOFF02-EMSGOFFS
A0CB 4C D2 A6 71      jmp DOERROR
A0CE      72 ;
A0CE      73 ;
A0CE      74 ; Print SYNTAX ERROR error message.
A0CE      75 ;
A0CE 4C C4 A6 76 ^9      jmp CSYNERR
A0D1      77 ;
A0D1      78 ;
A0D1      79 ; Set up defaults for the positional values.
A0D1      80 ;
A0D1      81 SETDFLTS:
A0D1 A9 00 82      lda #ZERO
A0D3 8D 65 AA 83      sta KYWRDFND
A0D6 8D 74 AA 84      sta MONVAL
A0D9 8D 66 AA 85      sta VOLVAL
A0DC 8D 6C AA 86      sta LENVAL
A0DF 8D 6D AA 87      sta LENVAL+1
A0E2      88 ;
A0E2 20 DC BF 89      jsr AJUSTBYT
A0E5      90 ;
A0E5      91 ;
A0E5      92 ; Come here if positionals are required. First, skip
A0E5      93 ; any leading blanks and commas.
A0E5      94 ;
A0E5 AD 5D AA 95      lda CMDLNIDX
A0E8      96 ;
A0E8      97 GETNXT:
A0E8 20 A4 A1 98      jsr FLSHCMDL
A0EB D0 1F 99      bne >1
A0ED     100 ;
A0ED C9 8D 101      cmp #RETURN
A0EF D0 F7 102      bne GETNXT
A0F1     103 ;
A0F1     104 ;
A0F1     105 ; At EOLN, make sure we have all the necessary
A0F1     106 ; arguments.
A0F1     107 ;
A0F1 AE 5F AA 108      ldx CMDINDX
A0F4     109 ;
A0F4 AD 65 AA 110      lda KYWRDFND
A0F7 1D 0A A9 111      ora KWRDPRMS+1,X
A0FA 5D 0A A9 112      eor KWRDPRMS+1,X
A0FD D0 93 113      bne CHKFRST
A0FF     114 ;
A0FF AE 63 AA 115      ldx TEMP1
A102 F0 76 116      beq PROCMD
A104     117 ;
A104 8D 63 AA 118      sta TEMP1
A107     119 ;
A107 8E 5D AA 120      stx CMDLNIDX
A10A D0 DC 121      bne GETNXT
```

```

A10C          122 ;
A10C          123 ;
A10C          124 ; Locate the keyword in the Keyword name table.
A10C          125 ;
A10C A2 0A     126 ^1      ldx #PARMBITS-PPARMS
A10E          127 ;
A10E DD 40 A9  128 ^2      cmp PPARMS-1,X
A111 F0 05     129          beq >4
A113          130 ;
A113 CA        131          dex
A114 D0 F8     132          bne <2
A116          133 ;
A116 F0 B6     134 ^3      beq <9
A118          135 ;
A118          136 ;
A118          137 ; As each positional argument is encountered, make it as
A118          138 ; present in KYWRDFND. Branch if C, I, or O found.
A118          139 ;
A118 BD 4A A9  140 ^4      lda PARMBITS-1,X
A11B 30 47     141          bmi >9
A11D          142 ;
A11D 0D 65 AA  143          ora KYWRDFND
A120 8D 65 AA  144          sta KYWRDFND
A123          145 ;
A123 CA        146          dex
A124 8E 64 AA  147          stx KYWRDIDX
A127          148 ;
A127          149 ;
A127          150 ; Get the numeric value associated with the keyword. If
A127          151 ; invalid then give "SYNTAX ERROR" message.
A127          152 ;
A127 20 B9 A1  153          jsr GETNUM
A12A B0 A2     154          bcs <9
A12C          155 ;
A12C          156 ;
A12C          157 ; Check the range of the value following the keyword. The
A12C          158 ; Keyword value valid range table contains four byte
A12C          159 ; entries for each keyword. Value in OPRND/OPRND+1.
A12C          160 ;
A12C AD 64 AA  161          lda KYWRDIDX
A12F 0A        162          asl
A130 0A        163          asl
A131 A8        164          tay
A132          165 ;
A132 A5 45     166          lda OPRND+1
A134 D0 09     167          bne >5
A136          168 ;
A136 A5 44     169          lda OPRND
A138 D9 55 A9  170          cmp KWRANGE,Y
A13B 90 8C     171          bcc <8
A13D          172 ;
A13D A5 45     173          lda OPRND+1
A13F          174 ;
A13F D9 58 A9  175 ^5      cmp KWRANGE+3,Y
A142 90 0B     176          bcc >7
A144          177 ;
A144 D0 83     178 ^6      bne <8
A146          179 ;
A146 A5 44     180          lda OPRND
A148 D9 57 A9  181          cmp KWRANGE+2,Y
A14B          182 ;

```

```

A14B 90 02      183      bcc >7
A14D D0 F5      184      bne <6
A14F            185      ;
A14F AD 63 AA    186      ^7      lda TEMP1
A152 D0 94      187      bne GETNXT
A154            188      ;
A154            189      ;
A154            190      ; Save value parsed in the keyword parameters table.
A154            191      ; These are two byte entries in the order of Volume, Drive,
A154            192      ; Slot, Length, Record, Byte, and Address.
A154            193      ;
A154 98          194      tya
A155 4A          195      lsr
A156 A8          196      tay
A157            197      ;
A157 A5 45       198      lda OPRND+1
A159 99 67 AA    199      sta VOLVAL+1,Y
A15C            200      ;
A15C A5 44       201      lda OPRND
A15E 99 66 AA    202      sta VOLVAL,Y
A161            203      ;
A161 4C E8 A0    204      ^8      jmp GETNXT
A164            205      ;
A164            206      ;
A164            207      ; If O, I, or C was encountered in MON or NOMON, clear the
A164            208      ; high order bit and set the appropriate MON MASK bits.
A164            209      ;
A164 48          210      ^9      pha
A165            211      ;
A165 A9 80       212      lda #MONFLAG
A167 0D 65 AA    213      ora KYWRDFND
A16A 8D 65 AA    214      sta KYWRDFND
A16D            215      ;
A16D 68          216      pla
A16E 29 7F       217      and #MONMASK
A170 0D 74 AA    218      ora MONVAL
A173 8D 74 AA    219      sta MONVAL
A176            220      ;
A176 D0 E9       221      bne <8
A178 F0 9C       222      beq <3
A17A            223      ;
A17A            224      ;
A17A            225      ; Process valid DOS command.
A17A            226      ;
A17A            227      PROCMD:
A17A 20 80 A1    228      jsr DOCMD
A17D            229      ;
A17D 4C 83 9F    230      jmp SCNXIT
A180            231      ;
A180            232      ;
A180            233      ; Reset to CSWL state 0, clear File Manager parameter list,
A180            234      ; then jump to command.
A180            235      ;
A180            236      DOCMD:
A180 20 5B A7     237      jsr RSET0
A183 20 AE A1    238      jsr CLRFBP
A186            239      ;
A186 AD 5F AA    240      lda CMDINDX
A189 AA          241      tax
A18A            242      ;
A18A BD 1F 9D    243      lda CMDTBL+1,X

```

```

A18D 48          244          pha
A18E          245          ;
A18E BD 1E 9D    246          lda CMDTBL,X
A191 48          247          pha
A192          248          ;
A192 60          249          rts
A193          250          ;
A193          251          ;
A193          252          ; Get next character on command line and check for a
A193          253          ; <cr> or a comma. Z-flag is returned set if a comma or
A193          254          ; <cr> is found.
A193          255          ;
A193          256          GNXTCHR:
A193 AE 5D AA    257          ldx CMDLNIDX
A196          258          ;
A196 BD 00 02    259          lda INPUT,X
A199 C9 8D        260          cmp #RETURN
A19B F0 06        261          beq >1
A19D          262          ;
A19D E8          263          inx
A19E 8E 5D AA    264          stx CMDLNIDX
A1A1          265          ;
A1A1 C9 AC        266          cmp #", "
A1A3          267          ;
A1A3 60          268          ^1 rts
A1A4          269          ;
A1A4          270          ;
A1A4          271          ; Deletes leading blanks from the command line.
A1A4          272          ;
A1A4          273          FLSHCMDL:
A1A4 20 93 A1    274          jsr GNXTCHR
A1A7 F0 FA        275          beq <1
A1A9          276          ;
A1A9 C9 A0        277          cmp #SPACE
A1AB F0 F7        278          beq FLSHCMDL
A1AD          279          ;
A1AD 60          280          rts
A1AE          281          ;
A1AE          282          ;
A1AE          283          ; Clears File Manager parameter list.
A1AE          284          ;
A1AE          285          CLRFBP:
A1AE A9 00        286          lda #ZERO
A1B0          287          ;
A1B0 A0 16        288          ldy #FTSTS-FMOPCOD
A1B2          289          ;
A1B2 99 BA B5    290          ^1 sta FMOPCOD-1,Y
A1B5          291          ;
A1B5 88          292          dey
A1B6 D0 FA        293          bne <1
A1B8          294          ;
A1B8 60          295          rts
A1B9          296          ;
A1B9          297          ;
A1B9          298          ; Converts numeric (in ASCII) argument to binary.  DECIMAL
A1B9          299          ; and HEX are supported.
A1B9          300          ;
A1B9          301          GETNUM:
A1B9 A9 00        302          lda #ZERO
A1BB 85 44        303          sta OPRND
A1BD 85 45        304          sta OPRND+1

```

```
A1BF      305 ;
A1BF 20 A4 A1 306      jsr FLSHCMDL
A1C2      307 ;
A1C2 08      308      php
A1C3      309 ;
A1C3 C9 A4   310      cmp #"$"
A1C5 F0 3C   311      beq >5
A1C7      312 ;
A1C7 28      313      plp
A1C8      314 ;
A1C8 4C CE A1 315      jmp >2
A1CB      316 ;
A1CB      317 ;
A1CB      318 ; Decimal convert routine here.
A1CB      319 ;
A1CB 20 A4 A1 320 ^1      jsr FLSHCMDL
A1CE      321 ;
A1CE D0 06   322 ^2      bne >3
A1D0      323 ;
A1D0 A6 44   324      ldx OPRND
A1D2 A5 45   325      lda OPRND+1
A1D4      326 ;
A1D4 18      327      clc
A1D5 60      328      rts
A1D6      329 ;
A1D6      330 ;
A1D6      331 ; Get current character, multiply OPRND by 10, and add in
A1D6      332 ; numeric value for current digit.
A1D6      333 ;
A1D6 38      334 ^3      sec
A1D7 E9 B0   335      sbc #"0"
A1D9 30 21   336      bmi >4
A1DB      337 ;
A1DB C9 0A   338      cmp #10
A1DD B0 1D   339      bcs >4
A1DF      340 ;
A1DF      341 ;
A1DF      342 ; Multiply OPRND by 10 (OPRND*2 + OPRND*8) and add in
A1DF      343 ; digit value.
A1DF      344 ;
A1DF 20 FE A1 345      jsr MUL2
A1E2      346 ;
A1E2 65 44   347      adc OPRND
A1E4 AA      348      tax
A1E5      349 ;
A1E5 A9 00   350      lda #ZERO
A1E7 65 45   351      adc OPRND+1
A1E9 A8      352      tay
A1EA      353 ;
A1EA 20 FE A1 354      jsr MUL2
A1ED 20 FE A1 355      jsr MUL2
A1F0      356 ;
A1F0 8A      357      txa
A1F1 65 44   358      adc OPRND
A1F3 85 44   359      sta OPRND
A1F5      360 ;
A1F5 98      361      tya
A1F6 65 45   362      adc OPRND+1
A1F8 85 45   363      sta OPRND+1
A1FA 90 CF   364      bcc <1
A1FC      365 ;
```

```
A1FC 38          366 ^4          sec
A1FD 60          367          rts
A1FE          368          ;
A1FE          369          ;
A1FE          370          ; Multiply OPRND/OPRND+1 by 2.
A1FE          371          ;
A1FE          372          MUL2:
A1FE 06 44      373          asl OPRND
A200 26 45      374          rol OPRND+1
A202          375          ;
A202 60          376          rts
A203          377          ;
A203          378          ;
A203          379          ; Hex convert routine here.
A203          380          ;
A203 28          381 ^5          plp
A204          382          ;
A204 20 A4 A1    383 ^6          jsr FLSHCMDL
A207 F0 C5      384          beq <2
A209          385          ;
A209 38          386          sec
A20A E9 B0      387          sbc #"0"
A20C 30 EE      388          bmi <4
A20E          389          ;
A20E C9 0A      390          cmp #10
A210 90 08      391          bcc >7
A212          392          ;
A212 E9 07      393          sbc #7
A214 30 E6      394          bmi <4
A216          395          ;
A216 C9 10      396          cmp #16
A218 B0 E2      397          bcs <4
A21A          398          ;
A21A A2 04      399 ^7          ldx #4
A21C          400          ;
A21C 20 FE A1    401 ^8          jsr MUL2
A21F          402          ;
A21F CA          403          dex
A220 D0 FA      404          bne <8
A222          405          ;
A222 05 44      406          ora OPRND
A224 85 44      407          sta OPRND
A226          408          ;
A226 4C 04 A2    409          jmp <6
A229          410          ;
A229          411          ;
A229          412          ; Do PR#n command. Exit in Monitor ROM.
A229          413          ;
A229          414          DOPRNUM:
A229 A5 44      415          lda OPRND
A22B          416          ;
A22B 4C 95 FE    417          jmp OUTPORT
A22E          418          ;
A22E          419          ;
A22E          420          ; Do IN#n command. Exit in Monitor ROM.
A22E          421          ;
A22E          422          DOINNUM:
A22E A5 44      423          lda OPRND
A230          424          ;
A230 4C 8B FE    425          jmp INPORT
A233          426          ;
```

```
A233      427 ;
A233      428 ; Do MON and NOMON commands.
A233      429 ;
A233      430 DOMON:
A233 AD 5E AA 431      lda MONFLGS
A236 0D 74 AA 432      ora MONVAL
A239 8D 5E AA 433      sta MONFLGS
A23C      434 ;
A23C 60      435      rts
A23D      436 ;
A23D      437 DONOMON:
A23D 2C 74 AA 438      bit MONVAL
A240 50 03      439      bvc >1
A242      440 ;
A242 20 C8 9F 441      jsr CRLF
A245      442 ;
A245 A9 70      443 ^1      lda #MONOMASK|MONIMASK|MONCMASK
A247 4D 74 AA 444      eor MONVAL
A24A 2D 5E AA 445      and MONFLGS
A24D 8D 5E AA 446      sta MONFLGS
A250      447 ;
A250 60      448      rts
A251      449 ;
A251      450 ;
A251      451 ; Handle MAXFILES commmand. Turn of any active EXEC file.
A251      452 ; Close all open files. Set the new MAXFILES number.
A251      453 ; Rebuild the DOS file buffers.
A251      454 ;
A251      455 DOMXFLS:
A251 A9 00      456      lda #ZERO
A253 8D B3 AA 457      sta EXFLG
A256      458 ;
A256 A5 44      459      lda OPRND
A258 48      460      pha
A259      461 ;
A259 20 16 A3 462      jsr CLOSEALL
A25C      463 ;
A25C 68      464      pla
A25D 8D 57 AA 465      sta MAXFILES
A260      466 ;
A260 4C D4 A7 467      jmp INITBUFS
A263      468 ;
A263      469 ;
A263      470 ; Handle DELETE command.
A263      471 ;
A263      472 DODELETE:
A263 A9 05      473      lda #FMDELECD
A265 20 AA A2 474      jsr CMDHNDL2
A268      475 ;
A268      476 ;
A268      477 ; Free up the buffer used by the delete command.
A268      478 ;
A268 20 64 A7 479      jsr LOCBUF
A26B      480 ;
A26B A0 00      481      ldy #ZERO
A26D 98      482      tya
A26E 91 40      483      sta (FILEBUFZ),Y
A270      484 ;
A270 60      485      rts
A271      486 ;
A271      487 ;
```



```
A271      488 ; Handle LOCK and UNLOCK commands.
A271      489 ;
A271      490 DOLOCK:
A271 A9 07  491         lda #FMLOCKCD
A273 D0 02  492         bne >1
A275      493 ;
A275      494 DOUNLOCK:
A275 A9 08  495         lda #FMUNLKCD
A277      496 ;
A277 20 AA A2 497 ^1      jsr CMDHNDL2
A27A      498 ;
A27A 4C EA A2 499         jmp DOCLOSE
A27D      500 ;
A27D      501 ;
A27D      502 ; Handle DOS VERIFY command.
A27D      503 ;
A27D      504 DOVERIFY:
A27D A9 0C  505         lda #FMVERICD
A27F D0 F6  506         bne <1
A281      507 ;
A281      508 ;
A281      509 ; Handle RENAME command. Store the address of second file
A281      510 ; name in File Manager parameter list.
A281      511 ;
A281      512 DORENAME:
A281 AD 08 9D 513         lda SFNADR
A284 8D BD B5 514         sta RECNUM
A287      515 ;
A287 AD 09 9D 516         lda SFNADR+1
A28A 8D BE B5 517         sta RECNUM+1
A28D      518 ;
A28D A9 09  519         lda #FMRENMCD
A28F 8D 63 AA 520         sta TEMP1
A292      521 ;
A292 20 C8 A2 522         jsr CLOSOPN
A295      523 ;
A295 4C EA A2 524         jmp DOCLOSE
A298      525 ;
A298      526 ;
A298      527 ; Handle APPEND command. Just do an open and then read the
A298      528 ; file until a zero is encountered.
A298      529 ;
A298      530 DOAPND:
A298 20 A3 A2 531         jsr DOOPEN
A29B      532 ;
A29B 20 8C A6 533 ^1      jsr RDTEXT
A29E D0 FB  534         bne <1
A2A0      535 ;
A2A0 4C 71 B6 536         jmp APTCH2
A2A3      537 ;
A2A3      538 ;
A2A3      539 ; OPEN a text file here.
A2A3      540 ;
A2A3      541 DOOPEN:
A2A3 A9 00  542         lda #TXTFLTYP
A2A5      543 ;
A2A5 4C D5 A3 544         jmp OPENTST
A2A8      545 ;
A2A8      546 ;
A2A8      547 ; File Manager setup used by the various DOS commands.
A2A8      548 ; This will OPEN a file.
```

```

A2A8      549      ;
A2A8      550      CMDHNDLR:
A2A8 A9 01      551      lda #FMOPENCD
A2AA      552      ;
A2AA 8D 63 AA   553      CMDHNDL2 sta TEMP1
A2AD      554      ;
A2AD AD 6C AA   555      lda LENVAL
A2B0 D0 0A      556      bne >1
A2B2      557      ;
A2B2 AD 6D AA   558      lda LENVAL+1
A2B5 D0 05      559      bne >1
A2B7      560      ;
A2B7 A9 01      561      lda #1
A2B9 8D 6C AA   562      sta LENVAL
A2BC      563      ;
A2BC AD 6C AA   564      ^1      lda LENVAL
A2BF 8D BD B5   565      sta RECNUM
A2C2      566      ;
A2C2 AD 6D AA   567      lda LENVAL+1
A2C5 8D BE B5   568      sta RECNUM+1
A2C8      569      ;
A2C8      570      ;
A2C8      571      ; Close a file if already open. Copy filename to file
A2C8      572      ; buffer. Copy buffer pointers and parameters to File
A2C8      573      ; Manager parameter list.
A2C8      574      ;
A2C8      575      CLOSOPN:
A2C8 20 EA A2   576      jsr DOCLOSE
A2CB      577      ;
A2CB A5 45      578      lda OPRND+1
A2CD D0 03      579      bne >1
A2CF      580      ;
A2CF 4C C8 A6   581      jmp NOBUF
A2D2      582      ;
A2D2 85 41      583      ^1      sta FILEBUFZ+1
A2D4      584      ;
A2D4 A5 44      585      lda OPRND
A2D6 85 40      586      sta FILEBUFZ
A2D8      587      ;
A2D8 20 43 A7   588      jsr COPYNAME
A2DB 20 4E A7   589      jsr COPYPTRS
A2DE 20 1A A7   590      jsr COPYPARM
A2E1      591      ;
A2E1 AD 63 AA   592      lda TEMP1
A2E4 8D BB B5   593      sta FMOPCOD
A2E7      594      ;
A2E7 4C A8 A6   595      jmp FMDRVR
A2EA      596      ;
A2EA      597      ;
A2EA      598      ; Handle CLOSE command.
A2EA      599      ;
A2EA      600      DOCLOSE:
A2EA AD 75 AA   601      lda FNAME
A2ED C9 A0      602      cmp #SPACE
A2EF F0 25      603      beq CLOSEALL
A2F1      604      ;
A2F1 20 64 A7   605      jsr LOCBUF
A2F4 B0 3A      606      bcs >4
A2F6      607      ;
A2F6 20 FC A2   608      jsr CLOFREE
A2F9      609      ;

```

```

A2F9 4C EA A2    610          jmp DOCLOSE
A2FC            611          ;
A2FC            612          ;
A2FC            613          ; Free any buffers used by file being closed.  Make EXEC
A2FC            614          ; inactive.
A2FC            615          ;
A2FC            616      CLOSFREE:
A2FC 20 AF A7    617          jsr ISEXCBUF
A2FF D0 05       618          bne >1
A301            619          ;
A301 A9 00       620          lda #ZERO
A303 8D B3 AA    621          sta EXFLG
A306            622          ;
A306 A0 00       623      ^1    ld  #ZERO
A308 98          624          tya
A309 91 40       625          sta (FILEBUFZ),Y
A30B            626          ;
A30B 20 4E A7    627          jsr COPYPTRS
A30E            628          ;
A30E A9 02       629          lda #FMCLOSCD
A310 8D BB B5    630          sta FMOPCOD
A313            631          ;
A313 4C A8 A6    632          jmp FMDRVR
A316            633          ;
A316            634          ;
A316            635          ; Close all open files.
A316            636          ;
A316            637      CLOSEALL:
A316 20 92 A7    638          jsr FRSTBUF
A319 D0 05       639          bne >3
A31B            640          ;
A31B 20 9A A7    641      ^2    jsr PNTNXTBF
A31E F0 10       642          beq >4
A320            643          ;
A320 20 AF A7    644      ^3    jsr ISEXCBUF
A323 F0 F6       645          beq <2
A325            646          ;
A325 20 AA A7    647          jsr GFBFNAM
A328 F0 F1       648          beq <2
A32A            649          ;
A32A 20 FC A2    650          jsr CLOSFREE
A32D            651          ;
A32D 4C 16 A3    652          jmp CLOSEALL
A330            653          ;
A330 60          654      ^4    rts
A331            655          ;
A331            656          ;
A331            657          ; Handle BSAVE command.  Check for A and L keywords.
A331            658          ;
A331            659      DOBSAVE:
A331 A9 09       660          lda #AKEYMASK|LKEYMASK
A333 2D 65 AA    661          and KYWRDFND
A336 C9 09       662          cmp #AKEYMASK|LKEYMASK
A338 F0 03       663          beq >1
A33A            664          ;
A33A 4C 00 A0    665          jmp GETFRST
A33D            666          ;
A33D            667          ;
A33D            668          ; Open file and make sure it is Binary.
A33D            669          ;
A33D A9 04       670      ^1    lda #BINFLTYP

```

```

A33F 20 D5 A3 671      jsr OPENTST
A342          672      ;
A342          673      ;
A342          674      ; Init address parameters.
A342          675      ;
A342 AD 73 AA 676      lda ADRVAL+1
A345 AC 72 AA 677      ldy ADRVAL
A348          678      ;
A348 20 E0 A3 679      jsr WRT2BYT
A34B          680      ;
A34B AD 6D AA 681      lda LENVAL+1
A34E AC 6C AA 682      ldy LENVAL
A351          683      ;
A351 20 E0 A3 684      jsr WRT2BYT
A354          685      ;
A354 AD 73 AA 686      lda ADRVAL+1
A357 AC 72 AA 687      ldy ADRVAL
A35A          688      ;
A35A 4C FF A3 689      jmp RWRANGE
A35D          690      ;
A35D          691      ;
A35D          692      ; Handle BLOAD command.
A35D          693      ;
A35D          694      DOBLOAD:
A35D 20 A8 A2 695      jsr CMDHNDLR
A360          696      ;
A360          697      ;
A360          698      ; Make sure it is a Binary file.
A360          699      ;
A360 A9 7F      700      lda #LOCKMASK
A362 2D C2 B5 701      and FILETYPE
A365 C9 04      702      cmp #BINFLTYP
A367 F0 03      703      beq >1
A369          704      ;
A369 4C D0 A6 705      jmp FMISMTCH
A36C          706      ;
A36C A9 04      707      ^1 lda #BINFLTYP
A36E 20 D5 A3 708      jsr OPENTST
A371          709      ;
A371          710      ;
A371          711      ; Read in length and address values.
A371          712      ;
A371 20 7A A4 713      jsr RD2BYT
A374 AA        714      tax
A375          715      ;
A375 AD 65 AA 716      lda KYWRDFND
A378 29 01      717      and #AKEYMASK
A37A D0 06      718      bne >2
A37C          719      ;
A37C 8E 72 AA 720      stx ADRVAL
A37F 8C 73 AA 721      sty ADRVAL+1
A382          722      ;
A382 20 7A A4 723      ^2 jsr RD2BYT
A385          724      ;
A385 AE 72 AA 725      ldx ADRVAL
A388 AC 73 AA 726      ldy ADRVAL+1
A38B          727      ;
A38B 4C 71 A4 728      jmp RWSETUP
A38E          729      ;
A38E          730      ;
A38E          731      ; Handle BRUN command.

```

```

A38E          732 ;
A38E          733 DOBRUN:
A38E 20 5D A3  734      jsr DOBLOAD
A391 20 51 A8  735      jsr INITPTRS
A394          736 ;
A394 6C 72 AA  737      jmp (ADRVAL)
A397          738 ;
A397          739 ;
A397          740 ; Handle SAVE command.
A397          741 ;
A397          742 DOSAVE:
A397 AD B6 AA  743      lda WHCBASIC
A39A F0 20     744      beq >2
A39C          745 ;
A39C          746 ;
A39C          747 ; Applesoft BASIC at this point.  Is file write protected?
A39C          748 ;
A39C A5 D6     749      lda PROTECT
A39E 10 03     750      bpl >1
A3A0          751 ;
A3A0 4C CC A6  752      jmp PTOOBIG
A3A3          753 ;
A3A3          754 ;
A3A3          755 ; Make sure it is an Applesoft file.  Compute program
A3A3          756 ; length.
A3A3          757 ;
A3A3 A9 02     758 ^1      lda #ASFLTYP
A3A5 20 D5 A3  759      jsr OPENTST
A3A8          760 ;
A3A8 38        761      sec
A3A9 A5 AF     762      lda ASPEND
A3AB E5 67     763      sbc ASPGMST
A3AD A8        764      tay
A3AE          765 ;
A3AE A5 B0     766      lda ASPEND+1
A3B0 E5 68     767      sbc ASPGMST+1
A3B2          768 ;
A3B2 20 E0 A3  769      jsr WRT2BYT
A3B5          770 ;
A3B5 A5 68     771      lda ASPGMST+1
A3B7 A4 67     772      ldy ASPGMST
A3B9          773 ;
A3B9 4C FF A3  774      jmp RWRANGE
A3BC          775 ;
A3BC          776 ;
A3BC          777 ; Save Integer BASIC file.  Compute program length.
A3BC          778 ;
A3BC A9 01     779 ^2      lda #INTFLTYP
A3BE 20 D5 A3  780      jsr OPENTST
A3C1          781 ;
A3C1 38        782      sec
A3C2 A5 4C     783      lda INTHIMEM
A3C4 E5 CA     784      sbc INTSTRT
A3C6 A8        785      tay
A3C7          786 ;
A3C7 A5 4D     787      lda INTHIMEM+1
A3C9 E5 CB     788      sbc INTSTRT+1
A3CB          789 ;
A3CB 20 E0 A3  790      jsr WRT2BYT
A3CE          791 ;
A3CE A5 CB     792      lda INTSTRT+1

```

```

A3D0 A4 CA      793          ldy INTSTRT
A3D2                794      ;
A3D2 4C FF A3    795          jmp RWRANGE
A3D5                796      ;
A3D5                797      ;
A3D5                798      ; Open a file and check its type.
A3D5                799      ;
A3D5                800  OPENTST:
A3D5 8D C2 B5    801          sta FILETYPE
A3D8 48          802          pha
A3D9                803      ;
A3D9 20 A8 A2    804          jsr CMDHNDLR
A3DC                805      ;
A3DC 68          806          pla
A3DD 4C C4 A7    807          jmp CHKTYPE
A3E0                808      ;
A3E0                809      ;
A3E0                810      ; Write two bytes out to a file.
A3E0                811      ;
A3E0                812  WRT2BYT:
A3E0 8C C1 B5    813          sty BYTRANGE
A3E3 8C C3 B5    814          sty DATADR
A3E6 8D C2 B5    815          sta BYTRANGE+1
A3E9                816      ;
A3E9 A9 04       817          lda #FMWRITCD
A3EB 8D BB B5    818          sta FMOPCOD
A3EE                819      ;
A3EE A9 01       820          lda #FMRW01SC
A3F0 8D BC B5    821          sta SUBCODE
A3F3                822      ;
A3F3 20 A8 A6    823          jsr FMDRVR
A3F6                824      ;
A3F6 AD C2 B5    825          lda BYTRANGE+1
A3F9 8D C3 B5    826          sta DATADR
A3FC                827      ;
A3FC 4C A8 A6    828          jmp FMDRVR
A3FF                829      ;
A3FF                830      ;
A3FF                831      ; Read or write a range of bytes.
A3FF                832      ;
A3FF                833  RWRANGE:
A3FF 8C C3 B5    834          sty DATADR
A402 8D C4 B5    835          sta DATADR+1
A405                836      ;
A405 A9 02       837          lda #FMRWNBSC
A407 4C 86 B6    838          jmp VFYPTCH
A40A                839      ;
A40A 20 A8 A6    840  RWRANGE2 jsr FMDRVR
A40D                841      ;
A40D 4C EA A2    842          jmp DOCLOSE
A410                843      ;
A410 4C D0 A6    844      ^1    jmp FMISMTCH
A413                845      ;
A413                846      ;
A413                847      ; Handle LOAD command. Close all files. Check type for
A413                848      ; Integer, Applesoft, or A file.
A413                849      ;
A413                850  DOLOAD:
A413 20 16 A3    851          jsr CLOSEALL
A416                852      ;
A416 20 A8 A2    853  DOLOAD2 jsr CMDHNDLR

```

```
A419          854 ;
A419 A9 23     855         lda #AFILETYP|ASFLTYP|INTFLTYP
A41B 2D C2 B5  856         and FILETYPE
A41E F0 F0     857         beq <1
A420          858 ;
A420 8D C2 B5  859         sta FILETYPE
A423          860 ;
A423 AD B6 AA  861         lda WHCBASIC
A426 F0 28     862         beq >2
A428          863 ;
A428          864 ;
A428          865 ; Select Applesoft.
A428          866 ;
A428 A9 02     867         lda #ASFLTYP
A42A 20 B1 A4  868         jsr SELBASIC
A42D          869 ;
A42D          870 ;
A42D          871 ; Read in length of Applesoft program.
A42D          872 ;
A42D 20 7A A4  873         jsr RD2BYT
A430          874 ;
A430          875 ;
A430          876 ; Compute Applesoft pointer values.
A430          877 ;
A430 18        878         clc
A431 65 67     879         adc ASPGMST
A433 AA        880         tax
A434          881 ;
A434 98        882         tya
A435 65 68     883         adc ASPGMST+1
A437 C5 74     884         cmp ASHIMEM+1
A439 B0 70     885         bcs PTOOLRG
A43B          886 ;
A43B 85 B0     887         sta ASPEND+1
A43D 85 6A     888         sta ASVARS+1
A43F          889 ;
A43F 86 AF     890         stx ASPEND
A441 86 69     891         stx ASVARS
A443          892 ;
A443 A6 67     893         ldx ASPGMST
A445 A4 68     894         ldy ASPGMST+1
A447          895 ;
A447 20 71 A4  896         jsr RWSETUP
A44A          897 ;
A44A 20 51 A8  898         jsr INITPTRS
A44D          899 ;
A44D 6C 60 9D  900         jmp (ASFTREL)
A450          901 ;
A450          902 ;
A450          903 ; Select Integer BASIC and load Integer program.
A450          904 ;
A450 A9 01     905 ^2      lda #INTFLTYP
A452 20 B1 A4  906         jsr SELBASIC
A455          907 ;
A455 20 7A A4  908         jsr RD2BYT
A458          909 ;
A458 38        910         sec
A459 A5 4C     911         lda INTHIMEM
A45B ED 60 AA  912         sbc LOADLEN
A45E AA        913         tax
A45F          914 ;
```

```

A45F A5 4D      915      lda INTHIMEM+1
A461 ED 61 AA   916      sbc LOADLEN+1
A464 90 45      917      bcc PTOOLRG
A466           918      ;
A466 A8         919      tay
A467 C4 4B      920      cpy INTLOMEM+1
A469 90 40      921      bcc PTOOLRG
A46B           922      ;
A46B F0 3E      923      beq PTOOLRG
A46D           924      ;
A46D 84 CB      925      sty INTSTRT+1
A46F 86 CA      926      stx INTSTRT
A471           927      ;
A471           928      ;
A471           929      ; Read or Write the range of bytes required.
A471           930      ;
A471           931      RWSETUP:
A471 8E C3 B5   932      stx DATADR
A474 8C C4 B5   933      sty DATADR+1
A477           934      ;
A477           935      ;
A477           936      ; TurboDOS replaces RWRANGE2 with jump to code at HBEB2.
A477           937      ;
A477 4C 0A A4   938      jmp RWRANGE2
A47A           939      ;
A47A           940      ;
A47A           941      ; Read two bytes from the specified file. Store value read
A47A           942      ; as range length in File Manager parameter list just in
A47A           943      ; case it was a length.
A47A           944      ;
A47A           945      RD2BYT:
A47A AD 0A 9D   946      lda LDRNGLEN
A47D 8D C3 B5   947      sta DATADR
A480           948      ;
A480 AD 0B 9D   949      lda LDRNGLEN+1
A483 8D C4 B5   950      sta DATADR+1
A486           951      ;
A486 A9 00      952      lda #ZERO
A488 8D C2 B5   953      sta BYTRANGE+1
A48B           954      ;
A48B A9 02      955      lda #2
A48D 8D C1 B5   956      sta BYTRANGE
A490           957      ;
A490 A9 03      958      lda #FMREADCD
A492 8D BB B5   959      sta FMOPCOD
A495           960      ;
A495 A9 02      961      lda #FMRWNBSC
A497 8D BC B5   962      sta SUBCODE
A49A           963      ;
A49A 20 A8 A6   964      jsr FMDRVR
A49D           965      ;
A49D AD 61 AA   966      lda LOADLEN+1
A4A0 8D C2 B5   967      sta BYTRANGE+1
A4A3 A8         968      tay
A4A4           969      ;
A4A4 AD 60 AA   970      lda LOADLEN
A4A7 8D C1 B5   971      sta BYTRANGE
A4AA           972      ;
A4AA 60         973      rts
A4AB           974      ;
A4AB           975      ;

```



```
A4AB      976 ; Print PROGRAM TOO LARGE error message.
A4AB      977 ;
A4AB      978 PTOOLRG:
A4AB 20 EA A2 979      jsr DOCLOSE
A4AE      980 ;
A4AE 4C CC A6 981      jmp PTOOBIG
A4B1      982 ;
A4B1      983 ;
A4B1      984 ; Select a BASIC.  If BASIC matches filetype, quit.  If
A4B1      985 ; not, select alternate BASIC.  See if the proper BASIC
A4B1      986 ; is already selected.
A4B1      987 ;
A4B1      988 SELBASIC:
A4B1 CD C2 B5 989      cmp FILETYPE
A4B4 F0 1A   990      beq >3
A4B6      991 ;
A4B6      992 ;
A4B6      993 ; If not, save CMDINDX and change BASICs.
A4B6      994 ;
A4B6 AE 5F AA 995      ldx CMDINDX
A4B9 8E 62 AA 996      stx PENDCMD
A4BC      997 ;
A4BC 4A      998      lsr
A4BD F0 03   999      beq >1
A4BF      1000 ;
A4BF 4C 9E A5 1001     jmp DOINT
A4C2      1002 ;
A4C2      1003 ;
A4C2      1004 ; If changing to Applesoft, save filename in case RAM
A4C2      1005 ; Applesoft is loading.  Remember, RAM Applesoft is an
A4C2      1006 ; Integer program called "APPLESOFT".  During its load
A4C2      1007 ; it will wipe out FNAME.
A4C2      1008 ;
A4C2 A2 1D   1009 ^1      ldx #NAME SIZE-1
A4C4      1010 ;
A4C4 BD 75 AA 1011 ^2      lda FNAME,X
A4C7 9D 93 AA 1012      sta SFNAME,X
A4CA      1013 ;
A4CA CA      1014      dex
A4CB 10 F7   1015      bpl <2
A4CD      1016 ;
A4CD 4C 7A A5 1017     jmp DOFP
A4D0      1018 ;
A4D0 60      1019 ^3      rts
A4D1      1020 ;
A4D1      1021 ;
A4D1      1022      icl "CMD3.L"
```

LLOAD CMD3.L,A\$4000

```

A4D1      1          ttl "DOS 3.3 Source Code, CMD3.L"
A4D1      2      ;
A4D1      3      ;
A4D1      4      ; CMD3.L
A4D1      5      ;
A4D1      6      ;
A4D1      7      ; Handle RUN command.  If Applesoft is active set RUN
A4D1      8      ; intercepted flag so that RUN can complete after Applesoft
A4D1      9      ; is loaded.  Call LOAD command handler to load the
A4D1     10      ; program.
A4D1     11      ;
A4D1     12  DORUN:
A4D1 AD B6 AA     13          lda WHCBASIC
A4D4 F0 03       14          beq >1
A4D6       15      ;
A4D6 8D B7 AA     16          sta RUNINTRC
A4D9       17      ;
A4D9 20 13 A4     18      ^1      jsr DOLOAD
A4DC       19      ;
A4DC       20      ;
A4DC       21      ; Skip a line on the screen.  Put DOS intercepts back.  Go
A4DC       22      ; to the RUN entry point in the current BASIC.
A4DC       23      ;
A4DC 20 C8 9F     24  DORUN2   jsr CRLF
A4DF 20 51 A8     25          jsr INITPTRS
A4E2       26      ;
A4E2 6C 58 9D     27          jmp (RUNADR)
A4E5       28      ;
A4E5       29      ;
A4E5       30      ; Integer BASIC, so handle RUN command here.  Delete all
A4E5       31      ; variables.  Go to the CHAIN entry point in Integer BASIC.
A4E5       32      ;
A4E5       33  DOSIRUN:
A4E5 A5 4A       34          lda INTLOMEM
A4E7 85 CC       35          sta INTVEND
A4E9       36      ;
A4E9 A5 4B       37          lda INTLOMEM+1
A4EB 85 CD       38          sta INTVEND+1
A4ED       39      ;
A4ED 6C 56 9D     40          jmp (CHNADR)
A4F0       41      ;
A4F0       42      ;
A4F0       43      ; Handle CHAIN command.  Call the LOAD command handler to
A4F0       44      ; load the program.  Skip a line.  Replace DOS intercepts.
A4F0       45      ; Got to current BASIC's CHAIN entry point.
A4F0       46      ;
A4F0       47  DOCHAIN:
A4F0 20 16 A4     48          jsr DOLOAD2
A4F3 20 C8 9F     49          jsr CRLF
A4F6 20 51 A8     50          jsr INITPTRS
A4F9       51      ;
A4F9 6C 56 9D     52          jmp (CHNADR)
A4FC       53      ;
A4FC       54      ;
A4FC       55      ; Handle Applesoft ROM RUN.  Call Applesoft to clear
A4FC       56      ; variables.  Reset ONERR.  Go to RUN entry point.
A4FC       57      ;
A4FC       58  DOSARUN:
A4FC 20 65 D6     59          jsr ASROMCLR
A4FF       60      ;

```

```
A4FF 85 33      61          sta PROMPT
A501 85 D8      62          sta ASONERR
A503           63          ;
A503 4C D2 D7   64          jmp ASROMNEW
A506           65          ;
A506           66          ;
A506           67          ; Handle Applesoft RAM RUN.  Call Applesoft to clear
A506           68          ; variables.  Reset ONERR.  Go to RUN entry point.
A506           69          ;
A506 20 65 0E   70  DOSARUN2 jsr ASRAMCLR
A509           71          ;
A509 85 33      72          sta PROMPT
A50B 85 D8      73          sta ASONERR
A50D           74          ;
A50D 4C D4 0F   75          jmp ASRAMNEW
A510           76          ;
A510           77          ;
A510           78          ; Handle WRITE command.  Call Read/Write common code.
A510           79          ; Change to CSWL state 5.  Exit DOS.
A510           80          ;
A510           81  DOWRITE:
A510 20 26 A5   82          jsr RWCOMMON
A513           83          ;
A513 A9 05      84          lda #STATE5
A515 8D 52 AA   85          sta CSWSTATE
A518           86          ;
A518 4C 83 9F   87          jmp SCNEXIT
A51B           88          ;
A51B           89          ;
A51B           90          ; Handle READ command.  Call Read/Write common code.  Set
A51B           91          ; READ mode flag in status flags.  Exit DOS.
A51B           92          ;
A51B           93  DOREAD:
A51B 20 26 A5   94          jsr RWCOMMON
A51E           95          ;
A51E A9 01      96          lda #READMODE
A520 8D 51 AA   97          sta CURSTAT
A523           98          ;
A523 4C 83 9F   99          jmp SCNEXIT
A526          100         ;
A526          101         ;
A526          102         ; Code common to both read and write.  First, see if a
A526          103         ; buffer is already allocated to the file.  On exit C-flag
A526          104         ; = 0 if file found.
A526          105         ;
A526          106  RWCOMMON:
A526 20 64 A7   107         jsr LOCBUF
A529 90 06      108         bcc >1
A52B          109         ;
A52B          110         ;
A52B          111         ; If not, open a file.
A52B          112         ;
A52B 20 A3 A2   113         jsr DOOPEN
A52E          114         ;
A52E 4C 34 A5   115         jmp >2
A531          116         ;
A531          117         ;
A531          118         ; If so, copy file buffer pointers to the File Manager
A531          119         ; parameter list.
A531          120         ;
A531 20 4E A7   121  ^1      jsr COPYPTRS
```

```

A534      122 ;
A534      123 ;
A534      124 ; Check to see if "R" or "B" parameters were specified.
A534      125 ;
A534 AD 65 AA 126 ^2      lda KYWRDFND
A537 29 06    127      and #RKEYMASK|BKEYMASK
A539 F0 13    128      beq >4
A53B      129 ;
A53B      130 ;
A53B      131 ; If so, copy numeric operands to the parameter list.
A53B      132 ;
A53B A2 03    133      ldx #3
A53D      134 ;
A53D BD 6E AA 135 ^3      lda RECVAL,X
A540 9D BD B5 136      sta RECNUM,X
A543      137 ;
A543 CA      138      dex
A544 10 F7    139      bpl <3
A546      140 ;
A546      141 ;
A546      142 ; Because a R or B parameter was specified, a POSITION
A546      143 ; command must be issued before the read or write command.
A546      144 ;
A546 A9 0A    145 RWCOMMN2 lda #FMPOSICD
A548 8D BB B5 146      sta FMOPCOD
A54B      147 ;
A54B 20 A8 A6 148      jsr FMDVR
A54E      149 ;
A54E 60      150 ^4      rts
A54F      151 ;
A54F      152 ;
A54F      153 ; Handle INIT command. If "V" parameter was given, use it.
A54F      154 ; Otherwise use 254.
A54F      155 ;
A54F      156 DOINIT:
A54F A9 40    157      lda #VKEYMASK
A551 2D 65 AA 158      and KYWRDFND
A554 F0 05    159      beq >1
A556      160 ;
A556 AD 66 AA 161      lda VOLVAL
A559 D0 05    162      bne >2
A55B      163 ;
A55B A9 FE    164 ^1      lda #DEFLTVAL
A55D 8D 66 AA 165      sta VOLVAL
A560      166 ;
A560      167 ;
A560      168 ; Get page number of DOS ($9D for 48K) and store in
A560      169 ; subcode field.
A560      170 ;
A560 AD 0D 9D 171 ^2      lda LOADADR+1
A563 8D BC B5 172      sta SUBCODE
A566      173 ;
A566      174 ;
A566      175 ; Call the File Manager to do the INIT.
A566      176 ;
A566 A9 0B    177      lda #FMINITCD
A568 20 AA A2 178      jsr CMDHNDL2
A56B      179 ;
A56B 4C 97 A3 180      jmp DOSAVE
A56E      181 ;
A56E      182 ;

```

```

A56E      183 ; Handle CATALOG command.
A56E      184 ;
A56E      185 DOCAT:
A56E A9 06   186         lda #FMCATACD
A570 20 AA A2 187         jsr CMDHNDL2
A573      188 ;
A573      189 ;
A573      190 ; Set default volume to the volume number of the disk just
A573      191 ; cataloged.
A573      192 ;
A573 AD BF B5 193         lda VOLUME
A576 8D 66 AA 194         sta VOLVAL
A579      195 ;
A579 60      196         rts
A57A      197 ;
A57A      198 ;
A57A      199 ; Handle FP command to change to Applesoft BASIC. Try to
A57A      200 ; select ROM Applesoft. If successful coldstart DOS.
A57A      201 ;
A57A      202 DOFP:
A57A A9 4C   203         lda #FPBASIC
A57C      204 ;
A57C 20 B2 A5 205         jsr SETBASIC
A57F F0 2E   206         beq >3
A581      207 ;
A581      208 ;
A581      209 ; Set up for Integer BASIC.
A581      210 ;
A581 A9 00   211         lda #INTACTV
A583 8D B6 AA 212         sta WHCBASIC
A586      213 ;
A586      214 ;
A586      215 ; Attempt to load in RAM Applesoft. "APPLESOFT" is 9
A586      216 ; characters in length.
A586      217 ;
A586 A0 1E   218         ldy #NAME SIZE
A588 20 97 A0 219         jsr BLKNAME2
A58B      220 ;
A58B A2 09   221         ldx #9
A58D      222 ;
A58D BD B7 AA 223 ^1      lda PGMNAME-1,X
A590 9D 74 AA 224         sta FNAME-1,X
A593      225 ;
A593 CA      226         dex
A594 D0 F7   227         bne <1
A596      228 ;
A596 A9 C0   229         lda #FPAACTV|FPOACTV
A598 8D 51 AA 230         sta CURSTAT
A59B      231 ;
A59B      232 ;
A59B      233 ; RUN APPLESOFT and if not present on the disk the FILE
A59B      234 ; NOT FOUND message is displayed. When DOS restarts the
A59B      235 ; RUN PROGRAM command will be finished.
A59B      236 ;
A59B 4C D1 A4 237         jmp DORUN
A59E      238 ;
A59E      239 ;
A59E      240 ; Handle INT command to change to Integer BASIC.
A59E      241 ;
A59E      242 DOINT:
A59E A9 20   243         lda #INTBASIC

```

```

A5A0 20 B2 A5    244      jsr SETBASIC
A5A3              245      ;
A5A3 F0 05      246      beq >2
A5A5              247      ;
A5A5              248      ;
A5A5              249      ; If it isn't there, then error, and print LANGUAGE NOT
A5A5              250      ; AVAILABLE error message.
A5A5              251      ;
A5A5 A9 01      252      lda #EMSOFF01-EMSGOFFS
A5A7 4C D2 A6    253      jmp DOERROR
A5AA              254      ;
A5AA              255      ;
A5AA              256      ; If it is there, restart DOS with RUN command set.
A5AA              257      ;
A5AA A9 00      258      ^2      lda #ZERO
A5AC 8D B7 AA    259      sta RUNINTRC
A5AF              260      ;
A5AF 4C 84 9D    261      ^3      jmp DOSSTRT
A5B2              262      ;
A5B2              263      ;
A5B2              264      ; Set the BASIC according to the value passed in A-reg.
A5B2              265      ; If the desired BASIC is already available, exit.
A5B2              266      ;
A5B2              267      ; Integer: A-reg = $20
A5B2              268      ; Applesoft: A-reg = $4C
A5B2              269      ;
A5B2              270      SETBASIC:
A5B2 CD 00 E0    271      cmp BASCLD
A5B5 F0 0E      272      beq >1
A5B7              273      ;
A5B7 8D 80 C0    274      sta RAM2WP
A5BA CD 00 E0    275      cmp BASCLD
A5BD F0 06      276      beq >1
A5BF              277      ;
A5BF 8D 81 C0    278      sta ROM2WE
A5C2 CD 00 E0    279      cmp BASCLD
A5C5              280      ;
A5C5 60          281      ^1      rts
A5C6              282      ;
A5C6              283      ;
A5C6              284      ; Handle EXEC command. Open the file. Copy file buffer
A5C6              285      ; address of EXEC's buffer pointer. Set EXEC active flag.
A5C6              286      ; Jump into POSITION command handler to skip R lines.
A5C6              287      ;
A5C6              288      DOEXEC:
A5C6 20 A3 A2    289      jsr DOOPEN
A5C9              290      ;
A5C9 AD 4F AA    291      lda BUFADR
A5CC 8D B4 AA    292      sta EXCBUF
A5CF              293      ;
A5CF AD 50 AA    294      lda BUFADR+1
A5D2 8D B5 AA    295      sta EXCBUF+1
A5D5              296      ;
A5D5 AD 75 AA    297      lda FNAME
A5D8 8D B3 AA    298      sta EXFLG
A5DB D0 0E      299      bne >2
A5DD              300      ;
A5DD              301      ;
A5DD              302      ; Handle POSITION command. Locate the open file buffer.
A5DD              303      ; If not found, open one as a TEXT file. Copy buffer
A5DD              304      ; pointers to File Manager parameter list. On exit C-flag

```

```

A5DD      305      ; = 0 if file found.
A5DD      306      ;
A5DD      307      DOPSTION:
A5DD 20 64 A7 308      jsr LOCBUF
A5E0 90 06 309      bcc >1
A5E2      310      ;
A5E2 20 A3 A2 311      jsr DOOPEN
A5E5 4C EB A5 312      jmp >2
A5E8      313      ;
A5E8 20 4E A7 314      ^1      jsr COPYPTRS
A5EB      315      ;
A5EB      316      ;
A5EB      317      ; Check for "R" parameter.  If R was not given, exit.
A5EB      318      ;
A5EB AD 65 AA 319      ^2      lda KYWRDFND
A5EE 29 04 320      and #RKEYMASK
A5F0 F0 1B 321      beq >6
A5F2      322      ;
A5F2      323      ;
A5F2      324      ; Search for the record specified by the "R" parameter.
A5F2      325      ;
A5F2 AD 6E AA 326      ^3      lda RECVAL
A5F5 D0 08 327      bne >4
A5F7      328      ;
A5F7 AE 6F AA 329      ldx RECVAL+1
A5FA F0 11 330      beq >6
A5FC      331      ;
A5FC CE 6F AA 332      dec RECVAL+1
A5FF      333      ;
A5FF CE 6E AA 334      ^4      dec RECVAL
A602      335      ;
A602 20 8C A6 336      ^5      jsr RDTEXT
A605 F0 38 337      beq >2
A607      338      ;
A607 C9 8D 339      cmp #RETURN
A609      340      ;
A609 D0 F7 341      bne <5
A60B F0 E5 342      beq <3
A60D      343      ;
A60D 60 344      ^6      rts
A60E      345      ;
A60E      346      ;
A60E      347      ; Write a byte to a file.  Check to see if BASIC is still
A60E      348      ; running and close the file if the program stopped
A60E      349      ; without closing the file.  Close the file and warmstart
A60E      350      ; DOS.  On exit C-flag = 0 if program running.
A60E      351      ;
A60E      352      WRITBYTE:
A60E 20 5E A6 353      jsr ISBASRUN
A611 B0 66 354      bcs CLSWARM
A613      355      ;
A613      356      ;
A613      357      ; Get the character and output it to the specified file.
A613      358      ;
A613 AD 5C AA 359      lda ASAVE
A616 8D C3 B5 360      sta DATBYTE
A619      361      ;
A619 A9 04 362      lda #FMWRITCD
A61B 8D BB B5 363      sta FMOPCOD
A61E      364      ;
A61E A9 01 365      lda #FMRW01SC

```

```

A620 8D BC B5 366          sta SUBCODE
A623          367          ;
A623 4C A8 A6 368          jmp FMDRVR
A626          369          ;
A626          370          ;
A626          371          ; Read a byte from a file. Once again, make sure the
A626          372          ; program is still running. Close the file if it isn't and
A626          373          ; warmstart DOS. On exit C-flag = 0 if program running.
A626          374          ;
A626          375 READBYTE:
A626 20 5E A6 376          jsr ISBASRUN
A629 B0 4E 377          bcs CLSWARM
A62B          378          ;
A62B          379          ;
A62B          380          ; Set up for state 6 to skip output of prompt character.
A62B          381          ;
A62B A9 06 382          lda #STATE6
A62D 8D 52 AA 383 ^1      sta CSWSTATE
A630          384          ;
A630          385          ;
A630          386          ; Read a character from the file.
A630          387          ;
A630 20 8C A6 388          jsr RDTEXT
A633 D0 0F 389          bne >3
A635          390          ;
A635          391          ;
A635          392          ; At EOF, so close file. If not in state 3 (EXEC) issue
A635          393          ; END OF DATA message.
A635          394          ;
A635 20 FC A2 395          jsr CLOFREE
A638          396          ;
A638 A9 03 397          lda #STATE3
A63A CD 52 AA 398          cmp CSWSTATE
A63D F0 CE 399          beq <6
A63F          400          ;
A63F A9 05 401 ^2      lda #EMSOFF05-EMSGOFFS
A641 4C D2 A6 402          jmp DOERROR
A644          403          ;
A644          404          ;
A644          405          ; If character was a lowercase character, strip its high
A644          406          ; order bit so the Apple Monitor GETLN routine doesn't
A644          407          ; convert it to upper case.
A644          408          ;
A644 C9 E0 409 ^3      cmp #LWRCASE
A646 90 02 410          bcc >4
A648          411          ;
A648 29 7F 412          and #ASCIMASK
A64A          413          ;
A64A 8D 5C AA 414 ^4      sta ASAVE
A64D          415          ;
A64D AE 5A AA 416          ldx XSAVE
A650 F0 09 417          beq >5
A652          418          ;
A652          419          ;
A652          420          ; In the event the last character was a lower case
A652          421          ; character in the range $60-$7F, force the high order bit
A652          422          ; on.
A652          423          ;
A652 CA 424          dex
A653          425          ;
A653 BD 00 02 426          lda INPUT,X

```



```

A656 09 80      427      ora #ASCIFLAG
A658 9D 00 02   428      sta INPUT,X
A65B           429      ;
A65B 4C B3 9F   430      ^5      jmp DOSXIT
A65E           431      ;
A65E           432      ;
A65E           433      ; Check to see if BASIC is executing.  On exit C-flag = 0
A65E           434      ; if BASIC is running.
A65E           435      ;
A65E           436      ISBASRUN:
A65E 48         437      pha
A65F           438      ;
A65F           439      ;
A65F           440      ; See if Integer or Applesoft is running.
A65F           441      ;
A65F AD B6 AA   442      lda WHCBASIC
A662 F0 0E      443      beq >7
A664           444      ;
A664           445      ;
A664           446      ; Applesoft is running.  First see if the line number's
A664           447      ; high order byte is $FF (not running).
A664           448      ;
A664 A6 76      449      ldx ASRUN
A666 E8         450      inx
A667 F0 0D      451      beq >8
A669           452      ;
A669           453      ;
A669           454      ; Now see if the prompt is "]" (used only in immediate
A669           455      ; mode).
A669           456      ;
A669 A6 33      457      ldx PROMPT
A66B E0 DD      458      cpx #"]"
A66D F0 07      459      beq >8
A66F           460      ;
A66F 68         461      ^6      pla
A670           462      ;
A670 18         463      clc                      ; running
A671 60         464      rts
A672           465      ;
A672           466      ;
A672           467      ; If Integer BASIC is active, check the Integer BASIC
A672           468      ; runmode flag (negative if running).
A672           469      ;
A672 A5 D9      470      ^7      lda INTRUN
A674 30 F9      471      bmi <6
A676           472      ;
A676 68         473      ^8      pla
A677           474      ;
A677 38         475      sec                      ; not running
A678 60         476      rts
A679           477      ;
A679           478      ;
A679           479      ; Closes all files and warmstarts DOS.
A679           480      ;
A679           481      CLSWARM:
A679 20 FC A2   482      jsr CLOFREE
A67C 20 5B A7   483      jsr RSET0
A67F           484      ;
A67F 4C B3 9F   485      jmp DOSXIT
A682           486      ;
A682           487      ;

```

```

A682      488      ; Read a character from the EXEC file buffer.  Set state to
A682      489      ; CSWL state 3.
A682      490      ;
A682      491      EXECRD:
A682 20 9D A6      492          jsr PNTEXEC
A685 20 4E A7      493          jsr COPYPTRS
A688      494      ;
A688 A9 03          495          lda #STATE3
A68A D0 A1          496          bne <1
A68C      497      ;
A68C      498      ;
A68C      499      ; Read a character from a text file.  Set up File Manager
A68C      500      ; to read one byte.
A68C      501      ;
A68C      502      RDTEXT:
A68C A9 03          503          lda #FMREADCD
A68E 8D BB B5      504          sta FMOPCOD
A691      505      ;
A691 A9 01          506          lda #FMRW01SC
A693 8D BC B5      507          sta SUBCODE
A696      508      ;
A696 20 A8 A6      509          jsr FMDVR
A699      510      ;
A699 AD C3 B5      511          lda DATBYTE
A69C 60            512          rts
A69D      513      ;
A69D      514      ;
A69D      515      ; Set up pointers for EXEC file.
A69D      516      ;
A69D      517      PNTEXEC:
A69D AD B5 AA      518          lda EXCBUF+1
A6A0 85 41          519          sta FILEBUFZ+1
A6A2      520      ;
A6A2 AD B4 AA      521          lda EXCBUF
A6A5 85 40          522          sta FILEBUFZ
A6A7      523      ;
A6A7 60            524          rts
A6A8      525      ;
A6A8      526      ;
A6A8      527      ; File Manager driver routine.  If END OF DATA call APTCH3,
A6A8      528      ; otherwise exit to caller.
A6A8      529      ;
A6A8      530      FMDVR:
A6A8 20 06 AB      531          jsr FILEMNGR
A6AB 90 16          532          bcc >2
A6AD      533      ;
A6AD AD C5 B5      534          lda RTNCODE
A6B0 C9 05          535          cmp #FMEOFERR
A6B2 F0 03          536          beq >1
A6B4      537      ;
A6B4 4C 5E B6      538          jmp APNDPTCH
A6B7      539      ;
A6B7 4C 92 B6      540      ^1      jmp APTCH3
A6BA      541      ;
A6BA EA            542          nop
A6BB      543      ;
A6BB      544      ;
A6BB      545      ; From APTCH3, clear APPFLG if not APPEND command.
A6BB      546      ;
A6BB 20 69 BA      547      FMDVR2  jsr HBA69
A6BE      548      ;

```

```

A6BE A2 00      549      ldx #ZERO
A6C0 8E C3 B5   550      stx DATADR
A6C3           551      ;
A6C3 60         552      ^2      rts
A6C4           553      ;
A6C4           554      ;
A6C4           555      ; Miscellaneous error messages.
A6C4           556      ;
A6C4           557      ; Print SYNTAX ERROR error message.
A6C4           558      ;
A6C4           559      CSYNERR:
A6C4 A9 0B      560      lda #EMSOFF11-MSGOFFS
A6C6 D0 0A      561      bne DOERROR
A6C8           562      ;
A6C8           563      ;
A6C8           564      ; Print NO BUFFERS AVAILABLE error message.
A6C8           565      ;
A6C8           566      NOBUF:
A6C8 A9 0C      567      lda #EMSOFF12-MSGOFFS
A6CA D0 06      568      bne DOERROR
A6CC           569      ;
A6CC           570      ;
A6CC           571      ; Print PROGRAM TOO LARGE error message.
A6CC           572      ;
A6CC           573      PTOOBIG:
A6CC A9 0E      574      lda #EMSOFF14-MSGOFFS
A6CE D0 02      575      bne DOERROR
A6D0           576      ;
A6D0           577      ;
A6D0           578      ; Print FILE TYPE MISMATCH error message.
A6D0           579      ;
A6D0           580      FMISMTCH:
A6D0 A9 0D      581      lda #EMSOFF13-MSGOFFS
A6D2           582      ;
A6D2           583      ;
A6D2           584      ; General print error routine.
A6D2           585      ;
A6D2           586      DOERROR:
A6D2 8D 5C AA   587      sta ASAVE
A6D5           588      ;
A6D5 20 E6 BF   589      jsr SETWARM
A6D8           590      ;
A6D8 AD B6 AA   591      lda WHCBASIC
A6DB F0 04      592      beq >1
A6DD           593      ;
A6DD           594      ;
A6DD           595      ; If running under Applesoft, check the ONERR flag before
A6DD           596      ; printing an error message.
A6DD           597      ;
A6DD A5 D8      598      lda ASONERR
A6DF 30 0E      599      bmi >2
A6E1           600      ;
A6E1           601      ;
A6E1           602      ; If Integer BASIC or Applesoft without the ONERR
A6E1           603      ; condition, print an error message.
A6E1           604      ;
A6E1 A2 00      605      ^1      ldx #ZERO
A6E3 20 02 A7   606      jsr PRTERORR
A6E6           607      ;
A6E6 AE 5C AA   608      ldx ASAVE
A6E9           609      ;

```

```

A6E9 20 02 A7    610          jsr PRERROR
A6EC 20 C8 9F    611          jsr CRLF
A6EF            612          ;
A6EF            613          ;
A6EF            614          ; Reset BASIC pointers just in case.  If BASIC is running
A6EF            615          ; jump to the the error handler.  If BASIC has stopped
A6EF            616          ; running do a warmstart.  If C-flag = 0 then running.
A6EF            617          ;
A6EF 20 51 A8    618          ^2      jsr INITPTRS
A6F2            619          ;
A6F2 20 5E A6    620          jsr ISBASRUN
A6F5            621          ;
A6F5 AE 5C AA    622          ldx ASAVE
A6F8            623          ;
A6F8 A9 03       624          lda #3              ; ???
A6FA B0 03       625          bcs >3
A6FC            626          ;
A6FC 6C 5A 9D    627          jmp (BASERR)
A6FF            628          ;
A6FF 6C 5E 9D    629          ^3      jmp (BASWARM)
A702            630          ;
A702            631          ;
A702            632          ; Print the error message specified by the X-reg.  Last
A702            633          ; character has most significant bit on.
A702            634          ;
A702            635          PRERROR:
A702 BD 3F AA    636          lda EMSGOFFS,X
A705 AA          637          tax
A706            638          ;
A706 8E 63 AA    639          ^1      stx TEMP1
A709            640          ;
A709 BD 71 A9    641          lda ERRMSGs,X
A70C 48          642          pha
A70D            643          ;
A70D 09 80       644          ora #ASCIFLAG
A70F 20 C5 9F    645          jsr CMDCOUT
A712            646          ;
A712 AE 63 AA    647          ldx TEMP1
A715 E8          648          inx
A716            649          ;
A716 68          650          pla
A717 10 ED       651          bpl <1
A719            652          ;
A719 60          653          rts
A71A            654          ;
A71A            655          ;
A71A            656          ; Copy parameters to File Manager parameter list.
A71A            657          ;
A71A            658          COPYPARM:
A71A AD 66 AA    659          lda VOLVAL
A71D 8D BF B5    660          sta VOLUME
A720            661          ;
A720 AD 68 AA    662          lda DRVAL
A723 8D C0 B5    663          sta DRIVE
A726            664          ;
A726 AD 6A AA    665          lda SLOTVAL
A729 8D C1 B5    666          sta SLOT
A72C            667          ;
A72C AD 06 9D    668          lda PFNADR
A72F 8D C3 B5    669          sta FNADR
A732            670          ;

```

```
A732 AD 07 9D      671          lda PFNADR+1
A735 8D C4 B5      672          sta FNADR+1
A738              673      ;
A738 A5 40          674          lda FILEBUFZ
A73A 8D 4F AA      675          sta BUFADR
A73D              676      ;
A73D A5 41          677          lda FILEBUFZ+1
A73F 8D 50 AA      678          sta BUFADR+1
A742              679      ;
A742 60            680          rts
A743              681      ;
A743              682      ;
A743              683      ; Copy primary filename to file buffer filename field.
A743              684      ;
A743              685      COPYNAME:
A743 A0 1D          686          ldy #NAME SIZE-1
A745              687      ;
A745 B9 75 AA      688      ^1      lda FNAME,Y
A748 91 40          689          sta (FILEBUFZ),Y
A74A              690      ;
A74A 88            691          dey
A74B 10 F8          692          bpl <1
A74D              693      ;
A74D 60            694          rts
A74E              695      ;
A74E              696      ;
A74E              697      ; Copy File Manager parameter list to file buffer pointers.
A74E              698      ;
A74E              699      COPYPTRS:
A74E A0 1E          700          ldy #WABUFADR-FILNAMBF
A750              701      ;
A750 B1 40          702      ^1      lda (FILEBUFZ),Y
A752 99 A9 B5      703          sta WBADR-WABUFADR-FILNAMBF,Y
A755              704      ;
A755 C8            705          iny
A756              706      ;
A756 C0 26          707          cpy #BUFREND-FILNAMBF
A758 D0 F6          708          bne <1
A75A              709      ;
A75A 60            710          rts
A75B              711      ;
A75B              712      ;
A75B              713          icl "CMD4.L"
```

LLOAD CMD4.L,A\$4000

```

A75B      1          ttl "DOS 3.3 Source Code, CMD4.L"
A75B      2          ;
A75B      3          ;
A75B      4          ; CMD4.L
A75B      5          ;
A75B      6          ;
A75B      7          ; Reset to CSWL state 0.  Sets the warmstart flag.
A75B      8          ;
A75B      9  RSET0:
A75B A0 00      10          ldy #STATE0
A75D 8C 51 AA   11          sty CURSTAT
A760 8C 52 AA   12          sty CSWSTATE
A763      13          ;
A763 60        14          rts
A764      15          ;
A764      16          ;
A764      17          ; Locates an available file buffer.  Or it locates a file
A764      18          ; buffer that has the save filename as the file being
A764      19          ; opened.  On exit C-flag = 0 if buffer found.
A764      20          ;
A764      21  LOCBUF:
A764 A9 00      22          lda #ZERO
A766 85 45      23          sta OPRND+1
A768      24          ;
A768      25          ;
A768      26          ; Go to the first file buffer and see if it is available.
A768      27          ;
A768 20 92 A7   28          jsr FRSTBUF
A76B      29          ;
A76B 4C 73 A7   30          jmp >2
A76E      31          ;
A76E      32          ;
A76E      33          ; Locate each successive buffer in the chain and see if it
A76E      34          ; is available.
A76E      35          ;
A76E 20 9A A7   36          ^1      jsr PNTNXTBF
A771 F0 1D      37          beq >5
A773      38          ;
A773      39          ;
A773      40          ; Is the buffer in use?
A773      41          ;
A773 20 AA A7   42          ^2      jsr GFBFNAM
A776 D0 0A      43          bne >3
A778      44          ;
A778      45          ;
A778      46          ; The buffer is free, check the next buffer.
A778      47          ;
A778 A5 40      48          lda FILEBUFZ
A77A 85 44      49          sta OPRND
A77C      50          ;
A77C A5 41      51          lda FILEBUFZ+1
A77E 85 45      52          sta OPRND+1
A780 D0 EC      53          bne <1          ; always taken
A782      54          ;
A782      55          ;
A782      56          ; Compare file names.
A782      57          ;
A782 A0 1D      58          ^3      ldy #NAME SIZE-1
A784      59          ;
A784 B1 40      60          ^4      lda (FILEBUFZ),Y

```

```
A786 D9 75 AA      61          cmp FNAME,Y
A789 D0 E3          62          bne <1
A78B                63          ;
A78B 88            64          dey
A78C 10 F6          65          bpl <4
A78E                66          ;
A78E 18            67          clc
A78F 60            68          rts
A790                69          ;
A790 38            70          ^5      sec
A791 60            71          rts
A792                72          ;
A792                73          ;
A792                74          ; Point DBUFP at first buffer in chain.
A792                75          ;
A792                76      FRSTBUF:
A792 AD 00 9D       77          lda DBUFP
A795 AE 01 9D       78          ldx DBUFP+1
A798 D0 0A          79          bne >1
A79A                80          ;
A79A                81          ;
A79A                82          ; Point DBUFP at next buffer in the chain.
A79A                83          ;
A79A                84      PNTNXTBF:
A79A A0 25          85          ldy #NXTFNADR-FILNAMBF-1
A79C                86          ;
A79C B1 40          87          lda (FILEBUFZ),Y
A79E F0 09          88          beq >2
A7A0                89          ;
A7A0 AA            90          tax
A7A1                91          ;
A7A1 88            92          dey
A7A2                93          ;
A7A2 B1 40          94          lda (FILEBUFZ),Y
A7A4                95          ;
A7A4 86 41          96          ^1      stx FILEBUFZ+1
A7A6 85 40          97          sta FILEBUFZ
A7A8                98          ;
A7A8 8A            99          txa
A7A9                100         ;
A7A9 60            101         ^2      rts
A7AA                102         ;
A7AA                103         ;
A7AA                104         ; Gets first character of filename in file buffer.
A7AA                105         ;
A7AA                106      GFBFNAM:
A7AA A0 00          107         ldy #ZERO
A7AC                108         ;
A7AC B1 40          109         lda (FILEBUFZ),Y
A7AE                110         ;
A7AE 60            111         rts
A7AF                112         ;
A7AF                113         ;
A7AF                114         ; See if this buffer is being used by an EXEC command.  If
A7AF                115         ; not, exit.
A7AF                116         ;
A7AF                117      ISEXCBUF:
A7AF AD B3 AA       118         lda EXFLG
A7B2 F0 0E          119         beq >1
A7B4                120         ;
A7B4 AD B4 AA       121         lda EXCBUF
```

```

A7B7 C5 40      122      cmp FILEBUFZ
A7B9 D0 08      123      bne >2
A7BB           124      ;
A7BB AD B5 AA   125      lda EXCBUF+1
A7BE C5 41      126      cmp FILEBUFZ+1
A7C0 F0 01      127      beq >2
A7C2           128      ;
A7C2 CA        129      ^1      dex                ; why?
A7C3           130      ;
A7C3 60        131      ^2      rts
A7C4           132      ;
A7C4           133      ;
A7C4           134      ; Check the file type.  If they match, exit.
A7C4           135      ;
A7C4           136      CHKTYPE:
A7C4 4D C2 B5   137      eor FILETYPE
A7C7 F0 0A      138      beq >1
A7C9           139      ;
A7C9 29 7F      140      and #LOCKMASK
A7CB F0 06      141      beq >1
A7CD           142      ;
A7CD 20 EA A2   143      jsr DOCLOSE
A7D0           144      ;
A7D0 4C D0 A6   145      jmp FMISMTCH
A7D3           146      ;
A7D3 60        147      ^1      rts
A7D4           148      ;
A7D4           149      ;
A7D4           150      ; Initialize the file buffer chain.  Set up for first file
A7D4           151      ; buffer.
A7D4           152      ;
A7D4           153      INITBUFS:
A7D4 38        154      sec                ; why?  It's done later.
A7D5           155      ;
A7D5 AD 00 9D   156      lda DBUFP
A7D8 85 40      157      sta FILEBUFZ
A7DA           158      ;
A7DA AD 01 9D   159      lda DBUFP+1
A7DD 85 41      160      sta FILEBUFZ+1
A7DF           161      ;
A7DF           162      ;
A7DF           163      ; Set up counter for maximum number of buffers.
A7DF           164      ;
A7DF AD 57 AA   165      lda MAXFILES
A7E2 8D 63 AA   166      sta TEMP1
A7E5           167      ;
A7E5           168      ;
A7E5           169      ; Begin by zeroing out the file name, marking it as free.
A7E5           170      ;
A7E5           171      FREENAME:
A7E5 A0 00      172      ldy #ZERO
A7E7 98        173      tya
A7E8 91 40      174      sta (FILEBUFZ),Y
A7EA           175      ;
A7EA           176      ;
A7EA           177      ; Initialize pointer to File Manager work area.
A7EA           178      ;
A7EA A0 1E      179      ldy #WABUFADR-FILNAMBF
A7EC           180      ;
A7EC 38        181      sec
A7ED           182      ;

```



```

A7ED A5 40      183      lda FILEBUFZ
A7EF E9 2D      184      sbc #FILNAMBF-WORKAREA
A7F1 91 40      185      sta (FILEBUFZ),Y
A7F3           186      ;
A7F3 48         187      pha
A7F4           188      ;
A7F4 A5 41      189      lda FILEBUFZ+1
A7F6 E9 00      190      sbc #ZERO
A7F8           191      ;
A7F8 C8         192      iny
A7F9           193      ;
A7F9 91 40      194      sta (FILEBUFZ),Y
A7FB           195      ;
A7FB           196      ;
A7FB           197      ; Set up pointer to T/S list buffer area.
A7FB           198      ;
A7FB AA         199      tax
A7FC CA         200      dex
A7FD           201      ;
A7FD 68         202      pla
A7FE 48         203      pha
A7FF           204      ;
A7FF C8         205      iny
A800           206      ;
A800 91 40      207      sta (FILEBUFZ),Y
A802           208      ;
A802 8A         209      txa
A803           210      ;
A803 C8         211      iny
A804           212      ;
A804 91 40      213      sta (FILEBUFZ),Y
A806           214      ;
A806           215      ;
A806           216      ; Set up pointer to data buffer area.
A806           217      ;
A806 AA         218      tax
A807 CA         219      dex
A808           220      ;
A808 68         221      pla
A809 48         222      pha
A80A           223      ;
A80A C8         224      iny
A80B           225      ;
A80B 91 40      226      sta (FILEBUFZ),Y
A80D           227      ;
A80D C8         228      iny
A80E           229      ;
A80E 8A         230      txa
A80F 91 40      231      sta (FILEBUFZ),Y
A811           232      ;
A811           233      ;
A811           234      ; If this isn't the last file buffer, set up pointer to
A811           235      ; the next buffer. Recall TEMP1 holds MAXFILES count.
A811           236      ;
A811 CE 63 AA   237      dec TEMP1
A814 F0 17      238      beq >1
A816           239      ;
A816 AA         240      tax
A817           241      ;
A817 68         242      pla
A818 38         243      sec

```

```
A819 E9 26      244      sbc #BUFREND-FILNAMBF
A81B           245      ;
A81B C8         246      iny
A81C           247      ;
A81C 91 40      248      sta (FILEBUFZ),Y
A81E 48         249      pha
A81F           250      ;
A81F 8A         251      txa
A820 E9 00      252      sbc #ZERO
A822           253      ;
A822 C8         254      iny
A823           255      ;
A823 91 40      256      sta (FILEBUFZ),Y
A825 85 41      257      sta FILEBUFZ+1
A827           258      ;
A827 68         259      pla
A828 85 40      260      sta FILEBUFZ
A82A           261      ;
A82A 4C E5 A7   262      jmp FREENAME
A82D           263      ;
A82D           264      ;
A82D           265      ; If at last file buffer, zero out the pointer to the next
A82D           266      ; buffer and return.
A82D           267      ;
A82D 48         268      ^1      pha
A82E           269      ;
A82E A9 00      270      lda #ZERO
A830           271      ;
A830 C8         272      iny
A831           273      ;
A831 91 40      274      sta (FILEBUFZ),Y
A833           275      ;
A833 C8         276      iny
A834           277      ;
A834 91 40      278      sta (FILEBUFZ),Y
A836           279      ;
A836           280      ;
A836           281      ; Set up HIMEM for the current BASIC.
A836           282      ;
A836 AD B6 AA   283      lda WHCBASIC
A839 F0 0B      284      beq >2
A83B           285      ;
A83B           286      ;
A83B           287      ; Set up HIMEM for Applesoft.
A83B           288      ;
A83B 68         289      pla
A83C 85 74      290      sta ASHIMEM+1
A83E 85 70      291      sta ASSTRS+1
A840           292      ;
A840 68         293      pla
A841 85 73      294      sta ASHIMEM
A843 85 6F      295      sta ASSTRS
A845           296      ;
A845 60         297      rts
A846           298      ;
A846           299      ;
A846           300      ; Set up HIMEM for Integer.
A846           301      ;
A846 68         302      ^2      pla
A847 85 4D      303      sta INTTHIMEM+1
A849 85 CB      304      sta INTSTRT+1
```

```

A84B          305 ;
A84B 68       306      pla
A84C 85 4C    307      sta INTHIMEM
A84E 85 CA    308      sta INTSTRT
A850          309 ;
A850 60       310      rts
A851          311 ;
A851          312 ;
A851          313 ; Restore DOS control over the KSWL and CSWL hooks.
A851          314 ;
A851          315 INITPTRS:
A851 A5 39    316      lda KSWL+1
A853 CD 03 9D 317      cmp DOSKBD+1
A856 F0 12    318      beq >1
A858          319 ;
A858 8D 56 AA 320      sta KSWLSAV+1
A85B          321 ;
A85B A5 38    322      lda KSWL
A85D 8D 55 AA 323      sta KSWLSAV
A860          324 ;
A860 AD 02 9D 325      lda DOSKBD
A863 85 38    326      sta KSWL
A865          327 ;
A865 AD 03 9D 328      lda DOSKBD+1
A868 85 39    329      sta KSWL+1
A86A          330 ;
A86A A5 37    331 ^1    lda CSWL+1
A86C CD 05 9D 332      cmp DOSVID+1
A86F F0 12    333      beq >2
A871          334 ;
A871 8D 54 AA 335      sta CSWLSAV+1
A874          336 ;
A874 A5 36    337      lda CSWL
A876 8D 53 AA 338      sta CSWLSAV
A879          339 ;
A879 AD 04 9D 340      lda DOSVID
A87C 85 36    341      sta CSWL
A87E          342 ;
A87E AD 05 9D 343      lda DOSVID+1
A881 85 37    344      sta CSWL+1
A883          345 ;
A883 60       346 ^2    rts
A884          347 ;
A884          348 ;
A884          349 ; DOS command name text table. This table consists of the
A884          350 ; ASCII name for each DOS command in order of command
A884          351 ; index values, with the last character of each indicated
A884          352 ; by the MSB being on. The table must end with a zero.
A884          353 ; Commands in order are:
A884          354 ;
A884          355 DOSCMDs:
A884 49 4E 49 356      dci 'INIT'          ; 00
A887 D4       357
A888 4C 4F 41 357      dci 'LOAD'          ; 01
A88B C4       358
A88C 53 41 56 358      dci 'SAVE'          ; 02
A88F C5       359
A890 52 55 CE 359      dci 'RUN'           ; 03
A893 43 48 41 360      dci 'CHAIN'        ; 04
A896 49 CE    361
A898 44 45 4C 361      dci 'DELETE'       ; 05

```

```

A89B 45 54 C5
A89E 4C 4F 43 362      dci 'LOCK'      ; 06
A8A1 CB
A8A2 55 4E 4C 363      dci 'UNLOCK'    ; 07
A8A5 4F 43 CB
A8A8 43 4C 4F 364      dci 'CLOSE'    ; 08
A8AB 53 C5
A8AD 52 45 41 365      dci 'READ'    ; 09
A8B0 C4
A8B1 45 58 45 366      dci 'EXEC'    ; 0A
A8B4 C3
A8B5 57 52 49 367      dci 'WRITE'   ; 0B
A8B8 54 C5
A8BA 50 4F 53 368      dci 'POSITION' ; 0C
A8BD 49 54 49
A8C0 4F CE
A8C2 4F 50 45 369      dci 'OPEN'    ; 0D
A8C5 CE
A8C6 41 50 50 370      dci 'APPEND'   ; 0E
A8C9 45 4E C4
A8CC 52 45 4E 371      dci 'RENAME'   ; 0F
A8CF 41 4D C5
A8D2 43 41 54 372      dci 'CATALOG'  ; 10
A8D5 41 4C 4F
A8D8 C7
A8D9 4D 4F CE 373      dci 'MON'     ; 11
A8DC 4E 4F 4D 374      dci 'NOMON'   ; 12
A8DF 4F CE
A8E1 50 52 A3 375      dci 'PR#'     ; 13
A8E4 49 4E A3 376      dci 'IN#'     ; 14
A8E7 4D 41 58 377      dci 'MAXFILES' ; 15
A8EA 46 49 4C
A8ED 45 D3
A8EF 46 D0 378      dci 'FP'      ; 16
A8F1 49 4E D4 379      dci 'INT'     ; 17
A8F4 42 53 41 380      dci 'BSAVE'   ; 18
A8F7 56 C5
A8F9 42 4C 4F 381      dci 'BLOAD'   ; 19
A8FC 41 C4
A8FE 42 52 55 382      dci 'BRUN'    ; 1A
A901 CE
A902 56 45 52 383      dci 'VERIFY'  ; 1B
A905 49 46 D9
A908 00 384      hex 00
A909 385 ;
A909 386 ;
A909 387 ; Command valid keywords table. This table is used to
A909 388 ; determine which keywords are required or may be given
A909 389 ; for any DOS command. Each command has a two byte entry
A909 390 ; with 16 flags, indicating which keywords may be given.
A909 391 ; The flag bit settings are as follows:
A909 392 ;
A909 393 ; BIT VALUE MEANING
A909 394 ; F 8000 Filename legal but optional
A909 395 ; E 4000 Command has no positional operand
A909 396 ; D 2000 Filename #1 expected
A909 397 ; C 1000 Filename #2 expected
A909 398 ; B 0800 Slot number positional operand expected
A909 399 ; A 0400 MAXFILES value expected as positional operand
A909 400 ; 9 0200 Command only issued from within a program
A909 401 ; 8 0100 Command creates a new file if file not found

```

```

A909          402 ; 7 0080 C, I, O keywords legal
A909          403 ; 6 0040 V keyword legal
A909          404 ; 5 0020 D keyword legal
A909          405 ; 4 0010 S keyword legal
A909          406 ; 3 0008 L keyword legal
A909          407 ; 2 0004 R keyword legal
A909          408 ; 1 0002 B keyword legal
A909          409 ; 0 0001 A keyword legal
A909          410 ;
A909          411 KWRDPRMS:
A909 21 70      412          hex 2170          ; INIT
A90B A0 70      413          hex A070          ; LOAD
A90D A1 70      414          hex A170          ; SAVE
A90F A0 70      415          hex A070          ; RUN
A911 20 70      416          hex 2070          ; CHAIN
A913 20 70      417          hex 2070          ; DELETE
A915 20 70      418          hex 2070          ; LOCK
A917 20 70      419          hex 2070          ; UNLOCK
A919 60 00      420          hex 6000          ; CLOSE
A91B 22 06      421          hex 2206          ; READ
A91D 20 74      422          hex 2074          ; EXEC
A91F 22 06      423          hex 2206          ; WRITE
A921 22 04      424          hex 2204          ; POSITION
A923 23 78      425          hex 2378          ; OPEN
A925 22 70      426          hex 2270          ; APPEND
A927 30 70      427          hex 3070          ; RENAME
A929 40 70      428          hex 4070          ; CATALOG
A92B 40 80      429          hex 4080          ; MON
A92D 40 80      430          hex 4080          ; NOMON
A92F 08 00      431          hex 0800          ; PR#
A931 08 00      432          hex 0800          ; IN#
A933 04 00      433          hex 0400          ; MAXFILES
A935 40 70      434          hex 4070          ; FP
A937 40 00      435          hex 4000          ; INT
A939 21 79      436          hex 2179          ; BSAVE
A93B 20 71      437          hex 2071          ; BLOAD
A93D 20 71      438          hex 2071          ; BRUN
A93F 20 70      439          hex 2070          ; VERIFY
A941          440 ;
A941          441 ;
A941          442 ; Keyword name table. This table contains all the ASCII
A941          443 ; names of the DOS keywords in standard order. Each
A941          444 ; keyword name occupies one byte.
A941          445 ;
A941          446 PPARMS:
A941 D6 C4 D3   447          asc "VDSL RBACIO"
A944 CC D2 C2
A947 C1 C3 C9
A94A CF
A94B          448 ;
A94B          449 ;
A94B          450 ; Keyword flag bit positions table. This table gives the
A94B          451 ; bit positions for each keyword into the second byte of
A94B          452 ; the Command valid keyword table and in the flag ($AA65)
A94B          453 ; which indicates which keywords were present on the
A94B          454 ; command line. Entries C0, A0, and 90 are not used for
A94B          455 ; the valid keyword table. The bit positions are:
A94B          456 ;
A94B          457 PARMBITS:
A94B 40         458          hex 40          ; V
A94C 20         459          hex 20          ; D

```

```

A94D 10          460          hex 10          ; S
A94E 08          461          hex 08          ; L
A94F 04          462          hex 04          ; R
A950 02          463          hex 02          ; B
A951 01          464          hex 01          ; A
A952 C0          465          hex C0          ; C (not used in table)
A953 A0          466          hex A0          ; I (not used in table)
A954 90          467          hex 90          ; O (not used in table)
A955            468          ;
A955            469          ;
A955            470          ; Keyword value valid range table. This table indicates
A955            471          ; the range any keyword value may legally have. Each
A955            472          ; keyword has a four byte entry, two bytes of minimum
A955            473          ; value, and two bytes of maximum value. Values are:
A955            474          ;
A955            475          ;
A955            476          Keyword Min   Max
A955 00 00 FE    477          KWRANGE:      hex 0000FE00          ; V      0      254
A958 00
A959 01 00 02    478          hex 01000200          ; D      1        2
A95C 00
A95D 01 00 07    479          hex 01000700          ; S      1        7
A960 00
A961 01 00 FF    480          hex 0100FF7F          ; L      1    32767
A964 7F
A965 00 00 FF    481          hex 0000FF7F          ; R      0    32767
A968 7F
A969 00 00 FF    482          hex 0000FF7F          ; B      0    32767
A96C 7F
A96D 00 00 FF    483          hex 0000FFFF          ; A      0    65535
A970 FF
A971            484          ;
A971            485          ;
A971            486          ; Error message text table. This table contains the text
A971            487          ; for each error code in order of error code number:
A971            488          ;
A971            489          ERRMSGs:
A971 0D 07 8D    490          ERRMSG0      hex 0D078D          ; return, bell, return
A974 4C 41 4E    491          ERRMSG1      dci 'LANGUAGE NOT AVAILABLE'
A977 47 55 41
A97A 47 45 20
A97D 4E 4F 54
A980 20 41 56
A983 41 49 4C
A986 41 42 4C
A989 C5
A98A 52 41 4E    492          ERRMSG2      dci 'RANGE ERROR'
A98D 47 45 20
A990 45 52 52
A993 4F D2
A995 57 52 49    493          ERRMSG3      dci 'WRITE PROTECTED'
A998 54 45 20
A99B 50 52 4F
A99E 54 45 43
A9A1 54 45 C4
A9A4 45 4E 44    494          ERRMSG4      dci 'END OF DATA'
A9A7 20 4F 46
A9AA 20 44 41
A9AD 54 C1
A9AF 46 49 4C    495          ERRMSG5      dci 'FILE NOT FOUND'
A9B2 45 20 4E

```

```

A9B5 4F 54 20
A9B8 46 4F 55
A9BB 4E C4
A9BD 56 4F 4C    496  ERRMSG6  dci 'VOLUME MISMATCH'
A9C0 55 4D 45
A9C3 20 4D 49
A9C6 53 4D 41
A9C9 54 43 C8
A9CC 49 2F 4F    497  ERRMSG7  dci 'I/O ERROR'
A9CF 20 45 52
A9D2 52 4F D2
A9D5 44 49 53    498  ERRMSG8  dci 'DISK FULL'
A9D8 4B 20 46
A9DB 55 4C CC
A9DE 46 49 4C    499  ERRMSG9  dci 'FILE LOCKED'
A9E1 45 20 4C
A9E4 4F 43 4B
A9E7 45 C4
A9E9 53 59 4E    500  ERRMSGA  dci 'SYNTAX ERROR'
A9EC 54 41 58
A9EF 20 45 52
A9F2 52 4F D2
A9F5 4E 4F 20    501  ERRMSGB  dci 'NO BUFFERS AVAILABLE'
A9F8 42 55 46
A9FB 46 45 52
A9FE 53 20 41
AA01 56 41 49
AA04 4C 41 42
AA07 4C C5
AA09 46 49 4C    502  ERRMSGC  dci 'FILE TYPE MISMATCH'
AA0C 45 20 54
AA0F 59 50 45
AA12 20 4D 49
AA15 53 4D 41
AA18 54 43 C8
AA1B 50 52 4F    503  ERRMSGD  dci 'PROGRAM TOO LARGE'
AA1E 47 52 41
AA21 4D 20 54
AA24 4F 4F 20
AA27 4C 41 52
AA2A 47 C5
AA2C 4E 4F 54    504  ERRMSG E  dci 'NOT DIRECT COMMAND'
AA2F 20 44 49
AA32 52 45 43
AA35 54 20 43
AA38 4F 4D 4D
AA3B 41 4E C4
AA3E 8D          505          hex 8D
AA3F          506  ;
AA3F          507  ;
AA3F          508  ; Error message text offset index table. This table
AA3F          509  ; contains the offset in bytes to the text of any given
AA3F          510  ; error message in the Error message text table. Entries
AA3F          511  ; are one byte each for each error code number.
AA3F          512  ;
AA3F          513  EMSGOFFS:
AA3F          514  EMSOFF00 dfs 1,ERRMSG0-ERRMSGs ; 00
AA40          515  EMSOFF01 dfs 1,ERRMSG1-ERRMSGs ; 03
AA41          516  EMSOFF02 dfs 1,ERRMSG2-ERRMSGs ; 19
AA42          517  EMSOFF03 dfs 1,ERRMSG2-ERRMSGs ; 19
AA43          518  EMSOFF04 dfs 1,ERRMSG3-ERRMSGs ; 24

```

```

AA44      519  EMSOFF05  dfs  1,ERRMSG4-ERRMSGs ; 33
AA45      520  EMSOFF06  dfs  1,ERRMSG5-ERRMSGs ; 3E
AA46      521  EMSOFF07  dfs  1,ERRMSG6-ERRMSGs ; 4C
AA47      522  EMSOFF08  dfs  1,ERRMSG7-ERRMSGs ; 5B
AA48      523  EMSOFF09  dfs  1,ERRMSG8-ERRMSGs ; 64
AA49      524  EMSOFF10  dfs  1,ERRMSG9-ERRMSGs ; 6D
AA4A      525  EMSOFF11  dfs  1,ERRMSGa-ERRMSGs ; 78
AA4B      526  EMSOFF12  dfs  1,ERRMSGb-ERRMSGs ; 84
AA4C      527  EMSOFF13  dfs  1,ERRMSGc-ERRMSGs ; 98
AA4D      528  EMSOFF14  dfs  1,ERRMSGd-ERRMSGs ; AA
AA4E      529  EMSOFF15  dfs  1,ERRMSGe-ERRMSGs ; BB
AA4F      530  ;
AA4F      531  ;
AA4F      532  ; DOS main routine variables.
AA4F      533  ;
AA4F      534  BUFADR     dfs  2,ZERO           ; current file buffer address
AA51      535  CURSTAT   dfs  1,ZERO           ; 00 - warmstart status
AA52      536  ;                               01 - READ state status
AA52      537  ;                               40 - Applesoft RAM
AA52      538  ;                               80 - coldstart status
AA52      539  CSWSTATE  dfs  1,ZERO           ; CSWL intercept state number
AA53      540  CSWLSAV   dfs  2,ZERO           ; address of true CSWL handler
AA55      541  KSWLSAV   dfs  2,ZERO           ; address of true KSWL handler
AA57      542  MAXFILES  dfs  2,ZERO           ; MAXFILES value
AA59      543  SSAVE     dfs  1,ZERO           ; S-flag save
AA5A      544  XSAVE     dfs  1,ZERO           ; X-reg save
AA5B      545  YSAVE     dfs  1,ZERO           ; Y-reg save
AA5C      546  ASAVE     dfs  1,ZERO           ; A-reg save
AA5D      547  CMDLNIDX  dfs  1,ZERO           ; offset into command line
AA5E      548  MONFLGS   dfs  1,ZERO           ; MON flags: 10 - O
AA5F      549  ;                               20 - I
AA5F      550  ;                               40 - C
AA5F      551  CMDINDX   dfs  1,ZERO           ; index of last command * 2
AA60      552  LOADLEN   dfs  2,ZERO           ; length for LOAD and BLOAD
AA62      553  PENDCMD   dfs  1,ZERO           ; index of any pending command
AA63      554  TEMP1     dfs  1,ZERO           ; scratch variable
AA64      555  KYWRDIDX  dfs  1,ZERO           ; index of current keyword
AA65      556  KYWRDFND  dfs  1,ZERO           ; command line keyword present
AA66      557  VOLVAL    dfs  2,ZERO           ; volume
AA68 01 00      558  DRVAL   hex 0100           ; drive
AA6A      559  SLOTVAL   dfs  2,ZERO           ; slot
AA6C      560  LENVAL    dfs  2,ZERO           ; length
AA6E      561  RECVAL    dfs  2,ZERO           ; record
AA70      562  BYTVAL    dfs  2,ZERO           ; byte
AA72      563  ADRVAL    dfs  2,ZERO           ; address
AA74      564  MONVAL    dfs  1,ZERO           ; MON value
AA75 C8 C5 CC   565  FNAME   asc "HELLO"         ; primary file name buffer
AA78 CC CF
AA7A      566           dfs  NAMESIZE-5,SPACE
AA93      567  SFNAME    dfs  NAMESIZE,SPACE   ; RENAME file name buffer
AAB1      568  MXFLS     dfs  1,3              ; MAXFILES default
AAB2      569  CTLD      dfs  1,CTRLD          ; control-D default
AAB3      570  EXFLG     dfs  1,ZERO           ; EXEC active flag, or zero
AAB4      571  EXCBUF    dfs  2,ZERO           ; EXEC file buffer address
AAB6 40         572  WHCBASIC hex 40           ; 00 - Integer active
AAB7      573  ;                               40 - Applesoft ROM active
AAB7      574  ;                               80 - Applesoft RAM active
AAB7      575  RUNINTRC  dfs  1,0              ; RUN intercepted flag
AAB8 C1 D0 D0   576  PGMNAME asc "APPLESOFT"   ; Applesoft RAM program
AABB CC C5 D3
AABE CF C6 D4

```



```
AAC1          577  ;  
AAC1          578  ;
```

```
BSAVE SEG01,D1,A$1000,B,L$0DC1
```

```
AAC1          579          usr SEG01,D1  
AAC1          580  ;  
AAC1          581  ;  
AAC1          582          icl "MNGR1A.L,D2"
```

```
LLOAD MNGR1A.L,D2,A$4000
```

```

AAC1      1          ttl "DOS 3.3 Source Code, MNGR1A.L"
AAC1      2          ;
AAC1      3          ;
AAC1      4          ; MNGR1A.L
AAC1      5          ;
AAC1      6          ;
AAC1      7          obj PAGE10
AAC1      8          usr
AAC1      9          ;
AAC1     10          ;
AAC1     11          ; File Manager constants.
AAC1     12          ;
AAC1 E8 B7     13 RWTSPADR adr TBLTYPE
AAC3 BB B3     14 VTOCPADR adr VTOCSB
AAC5 BB B4     15 DIRPADR  adr DIRSECBF
AAC7 00 C0     16          adr MEMTOP
AAC9     17          ;
AAC9     18          ;
AAC9     19          ; File Manager subroutine table. This table contains a two
AAC9     20          ; byte function handler routine address for each of the 14
AAC9     21          ; file Manager opcodes in opcode order.
AAC9     22          ;
AAC9     23          FMFTBL:
AAC9 7E B3     24          adr NOERROR-1
AACB 21 AB     25          adr OPNHNDLR-1
AACD 05 AC     26          adr CLSHNDLR-1
AACF 57 AC     27          adr RDHNDLR-1
AAD1 6F AC     28          adr WRTHNDLR-1
AAD3 2A AD     29          adr DELHNDLR-1
AAD5 97 AD     30          adr CATHNDLR-1
AAD7 EE AC     31          adr LCKHNDLR-1
AAD9 F5 AC     32          adr UNLKHNDL-1
AADB 39 AC     33          adr RENHNDLR-1
AADD 11 AD     34          adr POSHNDLR-1
AADF 8D AE     35          adr INITHNDL-1
AAE1 17 AD     36          adr VFYHNDLR-1
AAE3 7E B3     37          adr NOERROR-1
AAE5     38          ;
AAE5     39          ;
AAE5     40          ; Read Subcode table. This table contains a two byte
AAE5     41          ; function handler routine address for each of the 6 read
AAE5     42          ; subcodes.
AAE5     43          ;
AAE5     44          FMRSUB:
AAE5 7E B3     45          adr NOERROR-1
AAE7 89 AC     46          adr RD1BYTE-1
AAE9 95 AC     47          adr RDRANGE-1
AAEB 86 AC     48          adr POSRD1B-1
AAED 92 AC     49          adr POSRDR-1
AAEF 7E B3     50          adr NOERROR-1
AAF1     51          ;
AAF1     52          ;
AAF1     53          ; Write subcode table. This table contains a two byte
AAF1     54          ; function hanbdlr routine address for each of the 6 write
AAF1     55          ; subcodes.
AAF1     56          ;
AAF1     57          FMWSUB:
AAF1 7E B3     58          adr NOERROR-1
AAF3 BD AC     59          adr WRT1BYTE-1
AAF5 C9 AC     60          adr WRTRANGE-1

```

```

AAF7 BA AC      61      adr POSWRT1B-1
AAF9 C6 AC      62      adr POSWRTR-1
AAFB 7E B3      63      adr NOERROR-1
AAFD           64      ;
AAFD           65      ;
AAFD           66      ; External entry point for File Manager.
AAFD           67      ;
AAFD           68      ; If X-reg = 0, allocate a new file if file not found in
AAFD           69      ; directory. If X-reg = 1, do not allocate a new file if
AAFD           70      ; file not found.
AAFD           71      ;
AAFD           72      FMEXT:
AAFD E0 00      73      cpx #ZERO
AAFF F0 02      74      beq >1
AB01           75      ;
AB01 A2 02      76      ldx #CMDLOAD-CMDTBL
AB03           77      ;
AB03 8E 5F AA   78      ^1      stx CMDINDX
AB06           79      ;
AB06           80      ;
AB06           81      ; Main entry for File Manager.
AB06           82      ;
AB06           83      FILEMNGR:
AB06 BA        84      tsx
AB07 8E 9B B3   85      stx STKSAVE
AB0A           86      ;
AB0A           87      ;
AB0A           88      ; Get File Manager parameters.
AB0A           89      ;
AB0A 20 6A AE   90      jsr LDFMW
AB0D           91      ;
AB0D           92      ;
AB0D           93      ; Check to see if there is a valid File Manager opcode.
AB0D           94      ;
AB0D AD BB B5   95      lda FMOPCOD
AB10 C9 0D      96      cmp #FMVERICD+1
AB12 B0 0B      97      bcs >1
AB14           98      ;
AB14           99      ;
AB14          100      ; Using the opcode as an index into FMFTBL, jump to
AB14          101      ; appropriate routine.
AB14          102      ;
AB14 0A        103      asl
AB15 AA        104      tax
AB16          105      ;
AB16 BD CA AA   106      lda FMFTBL+1,X
AB19 48        107      pha
AB1A          108      ;
AB1A BD C9 AA   109      lda FMFTBL,X
AB1D 48        110      pha
AB1E          111      ;
AB1E 60        112      rts
AB1F          113      ;
AB1F          114      ;
AB1F          115      ; Go to opcode range error handler.
AB1F          116      ;
AB1F 4C 63 B3   117      ^1      jmp RANGERR
AB22          118      ;
AB22          119      ;
AB22          120      ; Open a file.
AB22          121      ;

```

```

AB22      122  OPNHNDLR:
AB22 20 28 AB 123      jsr CMNOPN
AB25      124      ;
AB25 4C 7F B3 125      jmp NOERROR
AB28      126      ;
AB28      127      ;
AB28      128      ; Used to open a file and called by various other File
AB28      129      ; Manager routines.
AB28      130      ;
AB28      131      ; Initialize the File Manager work area.
AB28      132      ;
AB28      133  CMNOPN:
AB28 20 DC AB 134      jsr INITFMW
AB2B      135      ;
AB2B      136      ;
AB2B      137      ; Set the sector length to 256 bytes.
AB2B      138      ;
AB2B A9 01    139      lda #1
AB2D 8D E3 B5 140      sta SECTLEN+1
AB30      141      ;
AB30      142      ;
AB30      143      ; If the record length is zero, set it to 1.
AB30      144      ;
AB30 AE BE B5 145      ldx RECNUM+1
AB33 AD BD B5 146      lda RECNUM
AB36 D0 05    147      bne >1
AB38      148      ;
AB38 E0 00    149      cpx #ZERO
AB3A D0 01    150      bne >1
AB3C      151      ;
AB3C E8      152      inx
AB3D      153      ;
AB3D 8D E8 B5 154      ^1 sta OPNRCLen
AB40 8E E9 B5 155      stx OPNRCLen+1
AB43      156      ;
AB43      157      ;
AB43      158      ; Search for the file name in the directory, and allocate
AB43      159      ; if not found.
AB43      160      ;
AB43 20 C9 B1 161      jsr LCDIRENT
AB46 90 5E    162      bcc >3
AB48      163      ;
AB48      164      ;
AB48      165      ; File not found in directory. See if a file is required
AB48      166      ; by checking bit 8.
AB48      167      ;
AB48 8E 9C B3 168      stx DIRINDX
AB4B      169      ;
AB4B AE 5F AA 170      ldx CMDINDX
AB4E      171      ;
AB4E BD 09 A9 172      lda KWRDPRMS,X
AB51      173      ;
AB51 AE 9C B3 174      ldx DIRINDX
AB54      175      ;
AB54 4A      176      lsr
AB55 B0 0D    177      bcs >2
AB57      178      ;
AB57      179      ;
AB57      180      ; At this point, the file must be present. See if
AB57      181      ; Applesoft is loading from disk.
AB57      182      ;

```

```

AB57 AD 51 AA 183          lda CURSTAT
AB5A C9 C0 184          cmp #FPAACTV|FPOACTV
AB5C D0 03 185          bne >1
AB5E 186 ;
AB5E 187 ;
AB5E 188 ; Loading Applesoft, so print LANGUAGE NOT AVAILABLE.
AB5E 189 ;
AB5E 4C 5F B3 190          jmp LNOTAVL
AB61 191 ;
AB61 192 ;
AB61 193 ; Loading a normal program, so print FILE NOT FOUND.
AB61 194 ;
AB61 4C 73 B3 195 ^1      jmp FNOTFND
AB64 196 ;
AB64 197 ;
AB64 198 ; File was not required to be in the directory. Set it up.
AB64 199 ; Initialize sector count to 1 since there will be a T/S
AB64 200 ; List sector initially.
AB64 201 ;
AB64 A9 00 202 ^2      lda #ZERO
AB66 9D E8 B4 203          sta FILESIZE+1,X
AB69 204 ;
AB69 A9 01 205          lda #1
AB6B 9D E7 B4 206          sta FILESIZE,X
AB6E 207 ;
AB6E 8E 9C B3 208          stx DIRINDX
AB71 209 ;
AB71 210 ;
AB71 211 ; Allocate a sector for the T/S list.
AB71 212 ;
AB71 20 44 B2 213          jsr ALLOCSEC
AB74 214 ;
AB74 215 ;
AB74 216 ; Set up T/S list and work buffer.
AB74 217 ;
AB74 AE 9C B3 218          ldx DIRINDX
AB77 219 ;
AB77 9D C7 B4 220          sta TSSECTOR,X
AB7A 8D D2 B5 221          sta FTSS
AB7D 8D D4 B5 222          sta CURTSS
AB80 223 ;
AB80 AD F1 B5 224          lda CURTRACK
AB83 9D C6 B4 225          sta TSTRACK,X
AB86 8D D1 B5 226          sta FTSTS
AB89 8D D3 B5 227          sta CURTSTS
AB8C 228 ;
AB8C AD C2 B5 229          lda FILETYPE
AB8F 9D C8 B4 230          sta FILTYP,X
AB92 231 ;
AB92 232 ;
AB92 233 ; Write directory back to disk.
AB92 234 ;
AB92 20 37 B0 235          jsr WRTDIRSC
AB95 236 ;
AB95 237 ;
AB95 238 ; Initialize the work, data, and T/S buffers.
AB95 239 ;
AB95 20 0C AF 240          jsr SELTSBUF
AB98 20 D6 B7 241          jsr ZEROBUFR
AB9B 20 3A AF 242          jsr SETUPRW
AB9E 243 ;

```

```

AB9E      244 ;
AB9E      245 ; Set the return code to FILE NOT FOUND.
AB9E      246 ;
AB9E AE 9C B3 247         ldx DIRINDX
ABA1      248 ;
ABA1 A9 06   249         lda #EMSOFF06-EMSGOFFS
ABA3 8D C5 B5 250         sta RTNCODE
ABA6      251 ;
ABA6      252 ;
ABA6      253 ; Fall through from above (file created and entered into
ABA6      254 ; the directory) or come from directory lookup if file was
ABA6      255 ; found in the directory.
ABA6      256 ;
ABA6      257 ; Copy the T/S list info into the work buffer.
ABA6      258 ;
ABA6 BD C6 B4 259 ^3     lda TSTRACK,X
ABA9 8D D1 B5 260         sta FTSTS
ABAC      261 ;
ABAC BD C7 B4 262         lda TSSECTOR,X
ABAF 8D D2 B5 263         sta FTSS
ABB2      264 ;
ABB2 BD C8 B4 265         lda FILTYP,X
ABB5      266 ;
ABB5      267 ;
ABB5      268 ; Copy file type to work buffer.
ABB5      269 ;
ABB5 8D C2 B5 270         sta FILETYPE
ABB8 8D F6 B5 271         sta FTYPE
ABBB      272 ;
ABBB      273 ;
ABBB      274 ; Copy file size to work buffer.
ABBB      275 ;
ABBB BD E7 B4 276         lda FILESIZE,X
ABBE 8D EE B5 277         sta SECCNT
ABC1      278 ;
ABC1 BD E8 B4 279         lda FILESIZE+1,X
ABC4 8D EF B5 280         sta SECCNT+1
ABC7      281 ;
ABC7 8E D9 B5 282         stx DIRBYTIX
ABCA      283 ;
ABCA      284 ;
ABCA      285 ; Initialize EOF to NEGONE (infinity).
ABCA      286 ;
ABCA A9 FF   287         lda #NEGONE
ABCC 8D E0 B5 288         sta RELSLRD
ABCF 8D E1 B5 289         sta RELSLRD+1
ABD2      290 ;
ABD2      291 ;
ABD2      292 ; Set up number of entries in the T/S list.
ABD2      293 ;
ABD2 AD E2 B3 294         lda NUMTSENT
ABD5 8D DA B5 295         sta SECPERTS
ABD8      296 ;
ABD8      297 ;
ABD8      298 ; Read in the first T/S list if there is one.
ABD8      299 ;
ABD8 18      300         clc
ABD9      301 ;
ABD9 4C 5E AF 302         jmp RDTSLIST
ABDC      303 ;
ABDC      304 ;

```

```

ABDC          305 ; Initialize the File Manager work area. Zero out the T/S
ABDC          306 ; list.
ABDC          307 ;
ABDC          308 INITFMW:
ABDC A9 00     309         lda #ZERO
ABDE AA       310         tax
ABDF          311 ;
ABDF 9D D1 B5 312 ^1     sta FTSTS,X
ABE2          313 ;
ABE2 E8       314         inx
ABE3 E0 2D    315         cpx #FMWAEND-FTSTS
ABE5          316 ;
ABE5 D0 F8    317         bne <1
ABE7          318 ;
ABE7          319 ;
ABE7          320 ; Initialize volume number (complimented), slot, and drive
ABE7          321 ; number. Return to caller.
ABE7          322 ;
ABE7 AD BF B5 323         lda VOLUME
ABEA 49 FF    324         eor #NEGONE
ABEC 8D F9 B5 325         sta VOLNUMBR
ABEF          326 ;
ABEF AD C0 B5 327         lda DRIVE
ABF2 8D F8 B5 328         sta DRVNUMBR
ABF5          329 ;
ABF5 AD C1 B5 330         lda SLOT
ABF8          331 ;
ABF8 0A       332         asl
ABF9 0A       333         asl
ABFA 0A       334         asl
ABFB 0A       335         asl
ABFC          336 ;
ABFC AA       337         tax
ABFD 8E F7 B5 338         stx SLOT16
AC00          339 ;
AC00          340 ;
AC00          341 ; Set up for the directory track on track 17.
AC00          342 ;
AC00 A9 11    343         lda #VTOCTRK
AC02 8D FA B5 344         sta TRKNUMBR
AC05          345 ;
AC05 60       346         rts
AC06          347 ;
AC06          348 ;
AC06          349 ; Close file handler. Check to see if data buffer or T/S
AC06          350 ; buffer needs to be written to disk.
AC06          351 ;
AC06          352 CLSHNDLR:
AC06 20 1D AF 353         jsr CHKBUF
AC09 20 34 AF 354         jsr CHKTS
AC0C          355 ;
AC0C          356 ;
AC0C          357 ; Release any unallocated sectors.
AC0C          358 ;
AC0C 20 C3 B2 359         jsr RLSALLC
AC0F          360 ;
AC0F          361 ;
AC0F          362 ; Check to see if VTOC has changed.
AC0F          363 ;
AC0F A9 02    364         lda #VTOCMASK
AC11 2D D5 B5 365         and FLAGS

```

```

AC14 F0 21      366      beq >2
AC16            367      ;
AC16            368      ;
AC16            369      ; VTOC changed, so read VTOC, update it, read directory,
AC16            370      ; and update file size. Clear C-flag to read first
AC16            371      ; directory sector or set C-flag to read next directory
AC16            372      ; sector.
AC16            373      ;
AC16 20 F7 AF    374      jsr RDVTOC
AC19            375      ;
AC19 A9 00       376      lda #ZERO
AC1B            377      ;
AC1B 18          378      clc
AC1C            379      ;
AC1C 20 11 B0    380      ^1 jsr RDDIRSEC
AC1F            381      ;
AC1F 38          382      sec
AC20            383      ;
AC20 CE D8 B5    384      dec DIRSECIX
AC23 D0 F7       385      bne <1
AC25            386      ;
AC25 AE D9 B5    387      ldx DIRBYTIX
AC28            388      ;
AC28 AD EE B5    389      lda SECCNT
AC2B 9D E7 B4    390      sta FILESIZE,X
AC2E            391      ;
AC2E AD EF B5    392      lda SECCNT+1
AC31 9D E8 B4    393      sta FILESIZE+1,X
AC34            394      ;
AC34 20 37 B0    395      jsr WRTDIRSC
AC37            396      ;
AC37            397      ;
AC37            398      ; Normal close and exit point.
AC37            399      ;
AC37 4C 7F B3    400      ^2 jmp NOERROR
AC3A            401      ;
AC3A            402      ;
AC3A            403      ; Rename handler. Call common code to locate/open the
AC3A            404      ; file.
AC3A            405      ;
AC3A            406      RENHNDLR:
AC3A 20 28 AB    407      jsr CMNOPN
AC3D            408      ;
AC3D            409      ;
AC3D            410      ; See if the file is locked.
AC3D            411      ;
AC3D AD F6 B5    412      lda FTYPE
AC40 30 2B       413      bmi >2
AC42            414      ;
AC42            415      ;
AC42            416      ; Not locked, so point BUFADR at the second file name and
AC42            417      ; copy it into the directory.
AC42            418      ;
AC42 AD BD B5    419      lda RECNUM
AC45 85 42       420      sta BUFADRZ
AC47            421      ;
AC47 AD BE B5    422      lda RECNUM+1
AC4A 85 43       423      sta BUFADRZ+1
AC4C            424      ;
AC4C AE 9C B3    425      ldx DIRINDX
AC4F            426      ;

```



```

AC4F 20 1C B2 427      jsr COPYFNAM
AC52          428      ;
AC52          429      ;
AC52          430      ; Write the directory back to the disk.
AC52          431      ;
AC52 20 37 B0 432      jsr WRTDIRSC
AC55          433      ;
AC55 4C 7F B3 434      jmp NOERROR
AC58          435      ;
AC58          436      ;
AC58          437      ; Read function handler. Make sure subcode is les than 5.
AC58          438      ;
AC58          439      RDHNDLR:
AC58 AD BC B5 440      lda SUBCODE
AC5B C9 05    441      cmp #FMPOSNSC+1
AC5D B0 0B    442      bcs >1
AC5F          443      ;
AC5F          444      ;
AC5F          445      ; Use the subcode as an index into the FMRSUB table. Jump
AC5F          446      ; to that address.
AC5F          447      ;
AC5F 0A       448      asl
AC60 AA       449      tax
AC61          450      ;
AC61 BD E6 AA 451      lda FMRSUB+1,X
AC64 48       452      pha
AC65          453      ;
AC65 BD E5 AA 454      lda FMRSUB,X
AC68 48       455      pha
AC69          456      ;
AC69 60       457      rts
AC6A          458      ;
AC6A 4C 67 B3 459      ^1 jmp RANGERR2
AC6D          460      ;
AC6D          461      ;
AC6D          462      ; Error exit point if the file was locked.
AC6D          463      ;
AC6D 4C 7B B3 464      ^2 jmp FILELOCK
AC70          465      ;
AC70          466      ;
AC70          467      ; Write function handler. Make sure the file is not
AC70          468      ; locked.
AC70          469      ;
AC70          470      WRTHNDLR:
AC70 AD F6 B5 471      lda FTYPE
AC73 30 F8    472      bmi <2
AC75          473      ;
AC75          474      ;
AC75          475      ; Make sure the subcode is less than 5.
AC75          476      ;
AC75 AD BC B5 477      lda SUBCODE
AC78 C9 05    478      cmp #FMPOSNSC+1
AC7A B0 EE    479      bcs <1
AC7C          480      ;
AC7C          481      ;
AC7C          482      ; Use subcode as an index into the FMWSUB table.
AC7C          483      ;
AC7C 0A       484      asl
AC7D AA       485      tax
AC7E          486      ;
AC7E BD F2 AA 487      lda FMWSUB+1,X

```

```

AC81 48          488          pha
AC82          489          ;
AC82 BD F1 AA    490          lda FMWSUB,X
AC85 48          491          pha
AC86          492          ;
AC86 60          493          rts
AC87          494          ;
AC87          495          ;
AC87          496          ; Position and read 1 byte handler:  positions, then falls
AC87          497          ; through to read 1 byte.
AC87          498          ;
AC87          499          POSRD1B:
AC87 20 00 B3    500          jsr CALPOSN
AC8A          501          ;
AC8A          502          ;
AC8A          503          ; Read 1 byte routine.
AC8A          504          ;
AC8A          505          RD1BYTE:
AC8A 20 A8 AC    506          jsr RDABYTE
AC8D 8D C3 B5    507          sta DATBYTE
AC90          508          ;
AC90 4C 7F B3    509          jmp NOERROR
AC93          510          ;
AC93          511          ;
AC93          512          ; Position and read a range of bytes.
AC93          513          ;
AC93          514          POSRDR:
AC93 20 00 B3    515          jsr CALPOSN
AC96          516          ;
AC96          517          ;
AC96          518          ; Read a range of bytes.
AC96          519          ;
AC96          520          RDRANGE:
AC96 20 B5 B1    521          jsr DECRNG
AC99          522          ;
AC99 20 A8 AC    523          jsr RDABYTE
AC9C 48          524          pha
AC9D          525          ;
AC9D 20 A2 B1    526          jsr MOVRANG
ACA0          527          ;
ACA0 A0 00       528          ldy #ZERO
ACA2          529          ;
ACA2 68          530          pla
ACA3 91 42       531          sta (BUFADRZ),Y
ACA5          532          ;
ACA5 4C 96 AC    533          jmp RDRANGE
ACA8          534          ;
ACA8          535          ;
ACA8          536          ; This routine reads a data byte from the file, reading in
ACA8          537          ; additional sectors as necessary.  On exit C-flag = 1 if
ACA8          538          ; end of data.
ACA8          539          ;
ACA8          540          RDABYTE:
ACA8 20 B6 B0    541          jsr RDNXTDA
ACAB B0 0B       542          bcs >1
ACAD          543          ;
ACAD B1 42       544          lda (BUFADRZ),Y
ACAF 48          545          pha
ACB0          546          ;
ACB0 20 5B B1    547          jsr INCREC
ACB3 20 94 B1    548          jsr INCPOS

```

```
ACB6          549 ;
ACB6 68       550     pla
ACB7          551 ;
ACB7 60       552     rts
ACB8          553 ;
ACB8 4C 6F B3 554 ^1     jmp ENDATA
ACBB          555 ;
ACBB          556 ;
ACBB          557 ; Position and write 1 byte handler.
ACBB          558 ;
ACBB          559 POSWRT1B:
ACBB 20 00 B3 560         jsr CALPOSN
ACBE          561 ;
ACBE          562 ;
ACBE          563 ; Write 1 byte handler.
ACBE          564 ;
ACBE          565 WRT1BYTE:
ACBE AD C3 B5 566         lda DATBYTE
ACC1 20 DA AC 567         jsr WRTBYTE
ACC4          568 ;
ACC4 4C 7F B3 569         jmp NOERROR
ACC7          570 ;
ACC7          571 ;
ACC7          572 ; Position and write a range of bytes.
ACC7          573 ;
ACC7          574 POSWRTR:
ACC7 20 00 B3 575         jsr CALPOSN
ACCA          576 ;
ACCA          577 ;
ACCA          578 ; Write a range of bytes.
ACCA          579 ;
ACCA          580 WRTRANGE:
ACCA 20 A2 B1 581         jsr MOVVRANG
ACCD          582 ;
ACCD A0 00    583         ldy #ZERO
ACCF          584 ;
ACCF B1 42    585         lda (BUFADRZ),Y
ACD1 20 DA AC 586         jsr WRTBYTE
ACD4          587 ;
ACD4 20 B5 B1 588         jsr DECRNG
ACD7          589 ;
ACD7 4C CA AC 590         jmp WRTRANGE
ACDA          591 ;
ACDA          592 ;
ACDA          593 ; Write 1 byte to a file.
ACDA          594 ;
ACDA          595 WRTBYTE:
ACDA 48       596         pha
ACDB          597 ;
ACDB 20 B6 B0 598         jsr RDNXTDA
ACDE          599 ;
ACDE 68       600         pla
ACDF 91 42    601         sta (BUFADRZ),Y
ACE1          602 ;
ACE1 A9 40    603         lda #DATAMASK
ACE3 0D D5 B5 604         ora FLAGS
ACE6 8D D5 B5 605         sta FLAGS
ACE9          606 ;
ACE9 20 5B B1 607         jsr INCREC
ACEC          608 ;
ACEC 4C 94 B1 609         jmp INCPOS
```

```
ACEF          610 ;
ACEF          611 ;
ACEF          612 ; Lock function handler. Set mask byte to lock.
ACEF          613 ;
ACEF          614 LCKHNDLR:
ACEF A9 80     615         lda #LOCKFLAG
ACF1 8D 9E B3  616         sta ALLCFLG
ACF4 D0 05     617         bne >1
ACF6          618 ;
ACF6          619 ;
ACF6          620 ; Unlock function handler. Set mask byte to unlock.
ACF6          621 ;
ACF6          622 UNLKHNDL:
ACF6 A9 00     623         lda #ZERO
ACF8 8D 9E B3  624         sta ALLCFLG
ACFB          625 ;
ACFB 20 28 AB  626 ^1      jsr CMNOPN
ACFE          627 ;
ACFE AE 9C B3  628         ldx DIRINDX
AD01          629 ;
AD01 BD C8 B4  630         lda FILTYP,X
AD04 29 7F     631         and #LOCKMASK
AD06 0D 9E B3  632         ora ALLCFLG
AD09 9D C8 B4  633         sta FILTYP,X
AD0C          634 ;
AD0C 20 37 B0  635         jsr WRTDIRSC
AD0F          636 ;
AD0F 4C 7F B3  637 ^2      jmp NOERROR
AD12          638 ;
AD12          639 ;
AD12          640 ; Position function handler.
AD12          641 ;
AD12          642 POSHNDLR:
AD12 20 00 B3  643         jsr CALPOSN
AD15          644 ;
AD15 4C 7F B3  645         jmp NOERROR
AD18          646 ;
AD18          647 ;
AD18          648 ; Verify handler.
AD18          649 ;
AD18          650 VFYHNDLR:
AD18 20 28 AB  651         jsr CMNOPN
AD1B          652 ;
AD1B          653 ;
AD1B          654 ; Read sectors until end of file.
AD1B          655 ;
AD1B 20 B6 B0  656 ^3      jsr RDNXTDA
AD1E B0 EF     657         bcs <2
AD20          658 ;
AD20 EE E4 B5  659         inc FILEPOSN
AD23 D0 F6     660         bne <3
AD25          661 ;
AD25 EE E5 B5  662         inc FILEPOSN+1
AD28          663 ;
AD28 4C 1B AD  664         jmp <3
AD2B          665 ;
AD2B          666 ;
AD2B          667 ; Delete handler. Locate and open the file.
AD2B          668 ;
AD2B          669 DELHNDLR:
AD2B 20 28 AB  670         jsr CMNOPN
```

```

AD2E      671 ;
AD2E      672 ;
AD2E      673 ; Check if the file is locked.
AD2E      674 ;
AD2E AE 9C B3 675      ldx DIRINDX
AD31      676 ;
AD31 BD C8 B4 677      lda FILTYP,X
AD34 10 03    678      bpl >1
AD36      679 ;
AD36 4C 7B B3 680      jmp FILELOCK
AD39      681 ;
AD39      682 ;
AD39      683 ; Not locked, so copy the T/S list pointer and put $FF in
AD39      684 ; the old T/S list pointer location.
AD39      685 ;
AD39 AE 9C B3 686 ^1      ldx DIRINDX
AD3C      687 ;
AD3C BD C6 B4 688      lda TSTRACK,X
AD3F 8D D1 B5 689      sta FTSTS
AD42 9D E6 B4 690      sta FILESIZE-1,X
AD45      691 ;
AD45 A9 FF    692      lda #DELETF LG
AD47 9D C6 B4 693      sta TSTRACK,X
AD4A      694 ;
AD4A BC C7 B4 695      ldy TSSECTOR,X
AD4D 8C D2 B5 696      sty FTSS
AD50      697 ;
AD50      698 ;
AD50      699 ; Write the directory back to disk.
AD50      700 ;
AD50 20 37 B0 701      jsr WRTDIRSC
AD53      702 ;
AD53      703 ;
AD53      704 ; Read the T/S list sectors and update the VTOC until there
AD53      705 ; are no more T/S sectors. Clear C-flag to read first T/S
AD53      706 ; sector. On exit C-flag = 1 if no more to read.
AD53      707 ;
AD53 18      708      clc
AD54      709 ;
AD54 20 5E AF 710 ^1      jsr RDTSLIST
AD57 B0 2A    711      bcs >4
AD59      712 ;
AD59 20 0C AF 713      jsr SELTSBUF
AD5C      714 ;
AD5C A0 0C    715      ldy #TSLSTOFF
AD5E      716 ;
AD5E 8C 9C B3 717 ^2      sty DIRINDX
AD61      718 ;
AD61 B1 42    719      lda (BUFADRZ),Y
AD63      720 ;
AD63      721 ;
AD63      722 ; If track number is zero or minus, skip it.
AD63      723 ;
AD63 30 0B    724      bmi >3
AD65 F0 09    725      beq >3
AD67      726 ;
AD67      727 ;
AD67      728 ; Otherwise update the VTOC.
AD67      729 ;
AD67 48      730      pha
AD68      731 ;

```

```
AD68 C8          732          iny
AD69             733          ;
AD69 B1 42       734          lda (BUFADRZ),Y
AD6B A8          735          tay
AD6C             736          ;
AD6C 68          737          pla
AD6D 20 89 AD    738          jsr FREESECT
AD70             739          ;
AD70             740          ;
AD70             741          ; Get the next T/S pair.
AD70             742          ;
AD70 AC 9C B3    743          ^3      ldy DIRINDX
AD73             744          ;
AD73 C8          745          iny
AD74 C8          746          iny
AD75 D0 E7       747          bne <2
AD77             748          ;
AD77             749          ;
AD77             750          ; If at the end of the T/S sector, free the buffer it uses
AD77             751          ; and try to get the next sector.
AD77             752          ;
AD77 AD D3 B5    753          lda CURTSTS
AD7A AC D4 B5    754          ldy CURTSS
AD7D             755          ;
AD7D 20 89 AD    756          jsr FREESECT
AD80             757          ;
AD80 38          758          sec
AD81 B0 D1       759          bcs <1
AD83             760          ;
AD83 20 FB AF    761          ^4      jsr WRTVTOC
AD86             762          ;
AD86 4C 7F B3    763          jmp NOERROR
AD89             764          ;
AD89             765          ;
AD89             766          icl "MNGR1B.L"
```

LLOAD MNGR1B.L,A\$4000

```

AD89          1          ttl "DOS 3.3 Source Code, MNGR1B.L"
AD89          2          ;
AD89          3          ;
AD89          4          ; MNGR1B.L
AD89          5          ;
AD89          6          ;
AD89          7          ; Deallocate a sector in the track bit map.  Set C-flag to
AD89          8          ; deallocate a sector.
AD89          9          ;
AD89         10         FREESECT:
AD89 38        11         sec
AD8A          12         ;
AD8A 20 DD B2  13         jsr RORBITMP
AD8D          14         ;
AD8D A9 00     15         lda #ZERO
AD8F          16         ;
AD8F A2 05     17         ldx #FTYPE-NEXTSECR+1
AD91          18         ;
AD91 9D F0 B5  19         ^1 sta NEXTSECR,X
AD94          20         ;
AD94 CA       21         dex
AD95 10 FA     22         bpl <1
AD97          23         ;
AD97 60        24         rts
AD98          25         ;
AD98          26         ;
AD98          27         ; Catalog handler.  Initialize the File Manager work area.
AD98          28         ;
AD98          29         CATHNDLR:
AD98 20 DC AB  30         jsr INITFMW
AD9B          31         ;
AD9B          32         ;
AD9B          33         ; Allow any volume of disk ($FF complimented equals volume
AD9B          34         ; zero).
AD9B          35         ;
AD9B A9 FF     36         lda #NEGONE
AD9D 8D F9 B5  37         sta VOLNUMBR
ADA0          38         ;
ADA0          39         ;
ADA0          40         ; Read VTOC to find out where the catalog is.
ADA0          41         ;
ADA0 20 F7 AF  42         jsr RDVTOC
ADA3          43         ;
ADA3          44         ;
ADA3          45         ; Set up counter to stop every 22 lines.
ADA3          46         ;
ADA3 A9 16     47         lda #LNSPERPG
ADA5 8D 9D B3  48         sta CNTR
ADA8          49         ;
ADA8          50         ;
ADA8          51         ; Skip 2 lines and print DISK VOLUME.
ADA8          52         ;
ADA8 20 2F AE  53         jsr SKIPLN
ADAB 20 2F AE  54         jsr SKIPLN
ADAE          55         ;
ADAE A2 0B     56         ldx #DSKVOLND-DISKVOL+1
ADB0          57         ;
ADB0 BD AF B3  58         ^1 lda DISKVOL,X
ADB3 20 ED FD  59         jsr COUT
ADB6          60         ;

```

```

ADB6 CA          61          dex
ADB7 10 F7       62          bpl <1
ADB9            63          ;
ADB9            64          ;
ADB9            65          ; Print the volume number.
ADB9            66          ;
ADB9 86 45       67          stx OPRND+1
ADBB            68          ;
ADBB AD F6 B7    69          lda VOLFND
ADBE 85 44       70          sta OPRND
ADC0            71          ;
ADC0 20 42 AE    72          jsr PRTDEC
ADC3            73          ;
ADC3 20 2F AE    74          jsr SKIPLN
ADC6 20 2F AE    75          jsr SKIPLN
ADC9            76          ;
ADC9            77          ;
ADC9            78          ; Print all the directory entries.  Clear C-flag to read
ADC9            79          ; first directory sector.
ADC9            80          ;
ADC9 18          81          clc
ADCA            82          ;
ADCA 20 11 B0    83          ^1 jsr RDDIRSEC
ADCD B0 5D       84          bcs >8
ADCF            85          ;
ADCF A2 00       86          ldx #ZERO
ADD1            87          ;
ADD1 8E 9C B3    88          ^2 stx DIRINDX
ADD4            89          ;
ADD4 BD C6 B4    90          lda TSTRACK,X
ADD7            91          ;
ADD7            92          ;
ADD7            93          ; If A-reg is zero, the end of the directory has been
ADD7            94          ; reached.  If A-reg < 0, the file was deleted and should
ADD7            95          ; not be listed.
ADD7            96          ;
ADD7 F0 53       97          beq >8
ADD9 30 4A       98          bmi >7
ADDB            99          ;
ADDB           100          ;
ADDB           101          ; Check to see if the file is locked.
ADDB           102          ;
ADDB A0 A0       103          ldy #" "
ADDD           104          ;
ADDD BD C8 B4    105          lda FILTYP,X
ADE0 10 02       106          bpl >3
ADE2           107          ;
ADE2 A0 AA       108          ldy #"*"
ADE4           109          ;
ADE4 98          110          ^3 tya
ADE5 20 ED FD    111          jsr COUT
ADE8           112          ;
ADE8           113          ;
ADE8           114          ; Output the file type.
ADE8           115          ;
ADE8 BD C8 B4    116          lda FILTYP,X
ADEB 29 7F       117          and #LOCKMASK
ADED           118          ;
ADED A0 07       119          ldy #DISKVOL-FTTBL+1
ADEF           120          ;
ADEF 0A          121          asl

```



```

ADF0          122 ;
ADF0 0A       123 ^4      asl
ADF1 B0 03    124      bcs >5
ADF3          125 ;
ADF3 88       126      dey
ADF4 D0 FA    127      bne <4
ADF6          128 ;
ADF6 B9 A7 B3 129 ^5      lda FTTBL,Y
ADF9 20 ED FD 130      jsr COUT
ADFC          131 ;
ADFC A9 A0    132      lda #SPACE
ADFE 20 ED FD 133      jsr COUT
AE01          134 ;
AE01          135 ;
AE01          136 ; Output the file size.
AE01          137 ;
AE01 BD E7 B4 138      lda FILESIZE,X
AE04 85 44    139      sta OPRND
AE06          140 ;
AE06 BD E8 B4 141      lda FILESIZE+1,X
AE09 85 45    142      sta OPRND+1
AE0B          143 ;
AE0B 20 42 AE 144      jsr PRTDEC
AE0E          145 ;
AE0E A9 A0    146      lda #SPACE
AE10 20 ED FD 147      jsr COUT
AE13          148 ;
AE13 E8       149      inx
AE14 E8       150      inx
AE15 E8       151      inx
AE16          152 ;
AE16          153 ;
AE16          154 ; Output the file name.
AE16          155 ;
AE16 A0 1D    156      ldy #NAME SIZE-1
AE18          157 ;
AE18 BD C6 B4 158 ^6      lda FILNAME-3,X
AE1B 20 ED FD 159      jsr COUT
AE1E          160 ;
AE1E E8       161      inx
AE1F          162 ;
AE1F 88       163      dey
AE20 10 F6    164      bpl <6
AE22          165 ;
AE22 20 2F AE 166      jsr SKIPLN
AE25          167 ;
AE25          168 ;
AE25          169 ; Get the next directory entry reading another directory
AE25          170 ; sector if necessary.
AE25          171 ;
AE25 20 30 B2 172 ^7      jsr NXTDIREN
AE28          173 ;
AE28 90 A7    174      bcc <2
AE2A B0 9E    175      bcs <1
AE2C          176 ;
AE2C          177 ;
AE2C          178 ; No more sectors to read.
AE2C          179 ;
AE2C 4C 7F B3 180 ^8      jmp NOERROR
AE2F          181 ;
AE2F          182 ;

```

```

AE2F      183      ; Prints a <cr> and checks to see if the counter has become
AE2F      184      ; zero or not.
AE2F      185      ;
AE2F      186      SKIPLN:
AE2F A9 8D      187      lda #RETURN
AE31 20 ED FD   188      jsr COUT
AE34      189      ;
AE34 CE 9D B3   190      dec CNTR
AE37 D0 08      191      bne >1
AE39      192      ;
AE39 20 0C FD   193      jsr RDKEY
AE3C      194      ;
AE3C A9 15      195      lda #LNSPERPG-1
AE3E 8D 9D B3   196      sta CNTR
AE41      197      ;
AE41 60        198      ^1      rts
AE42      199      ;
AE42      200      ;
AE42      201      ; Prints OPRND/OPRND+1 as a decimal integer.
AE42      202      ;
AE42      203      PRTDEC:
AE42 A0 02      204      ldy #FTTBL-DECTBL+1
AE44      205      ;
AE44 A9 00      206      ^1      lda #ZERO
AE46 48        207      pha
AE47      208      ;
AE47 A5 44      209      ^2      lda OPRND
AE49 D9 A4 B3   210      cmp DECTBL,Y
AE4C 90 12      211      bcc >3
AE4E      212      ;
AE4E F9 A4 B3   213      sbc DECTBL,Y
AE51 85 44      214      sta OPRND
AE53      215      ;
AE53 A5 45      216      lda OPRND+1
AE55 E9 00      217      sbc #ZERO
AE57 85 45      218      sta OPRND+1
AE59      219      ;
AE59 68        220      pla
AE5A 69 00      221      adc #ZERO
AE5C 48        222      pha
AE5D      223      ;
AE5D 4C 47 AE   224      jmp <2
AE60      225      ;
AE60 68        226      ^3      pla
AE61 09 B0      227      ora #"0"
AE63 20 ED FD   228      jsr COUT
AE66      229      ;
AE66 88        230      dey
AE67 10 DB      231      bpl <1
AE69      232      ;
AE69 60        233      rts
AE6A      234      ;
AE6A      235      ;
AE6A      236      ; Load File Manager work area from the file buffer.
AE6A      237      ;
AE6A      238      LDFMW:
AE6A 20 08 AF   239      jsr SELBUF
AE6D      240      ;
AE6D A0 00      241      ldy #FMNOERR
AE6F 8C C5 B5   242      sty RTNCODE
AE72      243      ;

```

```

AE72 B1 42      244 ^1      lda (BUFADRZ),Y
AE74 99 D1 B5   245      sta FTSTS,Y
AE77           246      ;
AE77 C8         247      iny
AE78 C0 2D     248      cpy #FMWAEND-FTSTS
AE7A D0 F6     249      bne <1
AE7C           250      ;
AE7C 18        251      clc
AE7D 60        252      rts
AE7E           253      ;
AE7E           254      ;
AE7E           255      ; Save File Manager work area into the file buffer.
AE7E           256      ;
AE7E           257      SAVFMW:
AE7E 20 08 AF   258      jsr SELBUF
AE81           259      ;
AE81 A0 00     260      ldy #ZERO
AE83           261      ;
AE83 B9 D1 B5  262 ^9      lda FTSTS,Y
AE86 91 42     263      sta (BUFADRZ),Y
AE88           264      ;
AE88 C8        265      iny
AE89           266      ;
AE89 C0 2D     267      cpy #FMWAEND-FTSTS
AE8B D0 F6     268      bne <9
AE8D           269      ;
AE8D 60        270      rts
AE8E           271      ;
AE8E           272      ;
AE8E           273      ; Format disk function handler.
AE8E           274      ;
AE8E           275      INITHNDL:
AE8E 20 DC AB   276      jsr INITFMW
AE91           277      ;
AE91 A9 04     278      lda #RWTSFRMT
AE93           279      ;
AE93           280      ;
AE93           281      ; Initialize the VTOC information.
AE93           282      ;
AE93 20 58 B0   283      jsr SETCMDCD
AE96           284      ;
AE96           285      ;
AE96           286      ; Volume number must be complimented.
AE96           287      ;
AE96 AD F9 B5  288      lda VOLNUMBR
AE99 49 FF     289      eor #NEGONE
AE9B 8D C1 B3  290      sta DSKVOL
AE9E           291      ;
AE9E           292      ;
AE9E           293      ; Set up next track to allocate at track 17.
AE9E           294      ;
AE9E A9 11     295      lda #VTOCTRK
AEA0 8D EB B3  296      sta NXTTOALC
AEA3           297      ;
AEA3           298      ;
AEA3           299      ; Allocate sectors in positive direction (from 17 up).
AEA3           300      ;
AEA3 A9 01     301      lda #ALOCFRWD
AEA5 8D EC B3  302      sta ALLCDIR
AEA8           303      ;
AEA8           304      ;

```

```

AEA8      305 ; Mark the "possible" 50 tracks as all being used.
AEA8      306 ;
AEA8 A2 38  307         ldx #BITMAP-VTOCSB
AEAA      308 ;
AEAA A9 00  309         lda #ZERO
AEAC      310 ;
AEAC 9D BB B3 311 ^1     sta VTOCSB,X
AEAF      312 ;
AEAF E8     313         inx
AEB0 D0 FA  314         bne <1
AEB2      315 ;
AEB2      316 ;
AEB2      317 ; Mark the disk's 35 tracks as unused except for tracks 0,
AEB2      318 ; 1, 2, and 17.
AEB2      319 ;
AEB2 A2 0C  320         ldx #3*4                ; track 3
AEB4      321 ;
AEB4 E0 8C  322 ^1     cpx #35*4                ; track 35
AEB6 F0 14  323         beq >3
AEB8      324 ;
AEB8 A0 03  325         ldy #3
AEBA      326 ;
AEBA B9 A0 B3 327 ^2     lda FREEMASK,Y
AEBD 9D F3 B3 328         sta BITMAP,X
AEC0      329 ;
AEC0 E8     330         inx
AEC1      331 ;
AEC1 88     332         dey
AEC2 10 F6  333         bpl <2
AEC4      334 ;
AEC4 E0 44  335         cpx #17*4                ; track 17
AEC6 D0 EC  336         bne <1
AEC8      337 ;
AEC8 A2 48  338         ldx #18*4                ; track 18
AECA D0 E8  339         bne <1
AECC      340 ;
AECC 20 FB AF 341 ^3     jsr WRTVTOC
AECF      342 ;
AECF      343 ;
AECF      344 ; Initialize the directory information.
AECF      345 ;
AECF A2 00  346         ldx #ZERO
AED1 8A     347         txa
AED2      348 ;
AED2 9D BB B4 349 ^1     sta DIRSECBF,X
AED5      350 ;
AED5 E8     351         inx
AED6 D0 FA  352         bne <1
AED8      353 ;
AED8 20 45 B0 354         jsr PRWTS DIR
AEDB      355 ;
AEDB      356 ;
AEDB      357 ; Set up for the directory track, track 17.
AEDB      358 ;
AEDB A9 11  359         lda #VTOCTRK
AEDD      360 ;
AEDD      361 ;
AEDD      362 ; Set up for the directory sector.
AEDD      363 ;
AEDD AC F0 B3 364         ldy NUMSCTRS
AEE0      365 ;

```

```
AEE0 88          366          dey
AEE1 88          367          dey
AEE2          368          ;
AEE2 8D EC B7    369          sta TNUM
AEE5          370          ;
AEE5          371          ;
AEE5          372          ; Write directory sectors out to disk.
AEE5          373          ;
AEE5 8D BC B4    374          ^1      sta TSNXTDIR
AEE8          375          ;
AEE8 8C BD B4    376          ^2      sty TSNXTDIR+1
AEEB          377          ;
AEEB C8          378          iny
AEEC 8C ED B7    379          sty SNUM
AEEF          380          ;
AEEF A9 02       381          lda #RWTSWRIT
AEF1 20 58 B0    382          jsr SETCMDCD
AEF4          383          ;
AEF4 AC BD B4    384          ld y TSNXTDIR+1
AEF7          385          ;
AEF7 88          386          dey
AEF8          387          ;
AEF8 30 05       388          bmi >3
AEFA D0 EC       389          bne <2
AEFC          390          ;
AEFC 98          391          tya
AEFD F0 E6       392          beq <1
AEFF          393          ;
AEFF 20 C2 B7    394          ^3      jsr RWTSPRMS
AF02          395          ;
AF02          396          ;
AF02          397          ; Write DOS out to the disk.
AF02          398          ;
AF02 20 4A B7    399          jsr PUTDOS
AF05          400          ;
AF05 4C 7F B3    401          jmp NOERROR
AF08          402          ;
AF08          403          ;
AF08          404          ; Select a File Manager work buffer.
AF08          405          ;
AF08          406          SELBUF:
AF08 A2 00       407          ldx #WBADR-WBADR
AF0A F0 06       408          beq >1
AF0C          409          ;
AF0C          410          ;
AF0C          411          ; Select a T/S list buffer.
AF0C          412          ;
AF0C          413          SELTSBUF:
AF0C A2 02       414          ldx #TSLSTADR-WBADR
AF0E D0 02       415          bne >1
AF10          416          ;
AF10          417          ;
AF10          418          ; Select a data buffer.
AF10          419          ;
AF10          420          SELDABUF:
AF10 A2 04       421          ldx #DATASADR-WBADR
AF12          422          ;
AF12 BD C7 B5    423          ^1      lda WBADR,X
AF15 85 42       424          sta BUFADRZ
AF17          425          ;
AF17 BD C8 B5    426          lda WBADR+1,X
```

```

AF1A 85 43      427      sta BUFADRZ+1
AF1C            428      ;
AF1C 60         429      rts
AF1D            430      ;
AF1D            431      ;
AF1D            432      ; Check data buffer to see if it needs to be written to
AF1D            433      ; disk.  If so, clear DATAMASK from FLAGS.
AF1D            434      ;
AF1D            435      CHKBUF:
AF1D 2C D5 B5   436      bit FLAGS
AF20 70 01      437      bvs >1
AF22            438      ;
AF22 60         439      rts
AF23            440      ;
AF23 20 E4 AF   441      ^1    jsr PREPDATA
AF26            442      ;
AF26 A9 02      443      lda #RWTSWRIT
AF28 20 52 B0   444      jsr RWTSDRVR
AF2B            445      ;
AF2B A9 BF      446      lda #DATAMASK^NEGONE
AF2D 2D D5 B5   447      and FLAGS
AF30 8D D5 B5   448      sta FLAGS
AF33            449      ;
AF33 60         450      rts
AF34            451      ;
AF34            452      ;
AF34            453      ; Check T/S list buffer to see if it must be written to
AF34            454      ; disk.  If so, clear TSBFRMSK from FLAGS.
AF34            455      ;
AF34            456      CHKTS:
AF34 AD D5 B5   457      lda FLAGS
AF37 30 01      458      bmi SETUPRW
AF39            459      ;
AF39 60         460      rts
AF3A            461      ;
AF3A            462      ;
AF3A            463      SETUPRW:
AF3A 20 4B AF   464      jsr PREPRWTS
AF3D            465      ;
AF3D A9 02      466      lda #RWTSWRIT
AF3F 20 52 B0   467      jsr RWTSDRVR
AF42            468      ;
AF42 A9 7F      469      lda #TSBFRMSK^NEGONE
AF44 2D D5 B5   470      and FLAGS
AF47 8D D5 B5   471      sta FLAGS
AF4A            472      ;
AF4A 60         473      rts
AF4B            474      ;
AF4B            475      ;
AF4B            476      ; Set up IOB for RWTS call.
AF4B            477      ;
AF4B            478      PREPRWTS:
AF4B AD C9 B5   479      lda TSLSTADR
AF4E 8D F0 B7   480      sta USRBUF
AF51            481      ;
AF51 AD CA B5   482      lda TSLSTADR+1
AF54 8D F1 B7   483      sta USRBUF+1
AF57            484      ;
AF57 AE D3 B5   485      ldx CURTSTS
AF5A AC D4 B5   486      ldy CURTSS
AF5D            487      ;

```

```

AF5D 60          488          rts
AF5E            489          ;
AF5E            490          ;
AF5E            491          ; Read a T/S list into the file buffer.  On entry if C-flag
AF5E            492          ; = 0 then read first T/S sector.  If C-flag = 1 then read
AF5E            493          ; the next T/S sector.
AF5E            494          ;
AF5E            495          RDTSLIST:
AF5E 08          496          php
AF5F            497          ;
AF5F            498          ;
AF5F            499          ; First, check to see if the current T/S list must be
AF5F            500          ; written to disk.
AF5F            501          ;
AF5F 20 34 AF    502          jsr CHKTS
AF62 20 4B AF    503          jsr PREPRWTS
AF65 20 0C AF    504          jsr SELTSBUF
AF68            505          ;
AF68 28          506          plp
AF69 B0 09       507          bcs >1
AF6B            508          ;
AF6B            509          ;
AF6B            510          ; C-flag = 0, so read the first sector of the T/S list.
AF6B            511          ;
AF6B AE D1 B5    512          ldx FTSTS
AF6E AC D2 B5    513          ldy FTSS
AF71            514          ;
AF71 4C B5 AF    515          jmp OLDRWTS
AF74            516          ;
AF74            517          ;
AF74            518          ; C-flag = 1, so read in the next sector of the T/S list.
AF74            519          ;
AF74 A0 01       520          ^1 ldy #TSTRKOFF
AF76            521          ;
AF76            522          ;
AF76            523          ; If the track number of the next sector is zero, there
AF76            524          ; are no more T/S sectors available.
AF76            525          ;
AF76 B1 42       526          lda (BUFADRZ),Y
AF78 F0 08       527          beq >2
AF7A            528          ;
AF7A            529          ;
AF7A            530          ; Another T/S list is available so read it in.
AF7A            531          ;
AF7A AA          532          tax
AF7B            533          ;
AF7B C8          534          iny
AF7C            535          ;
AF7C B1 42       536          lda (BUFADRZ),Y
AF7E A8          537          tay
AF7F            538          ;
AF7F 4C B5 AF    539          jmp OLDRWTS
AF82            540          ;
AF82            541          ;
AF82            542          ; If the File Manager Opcode isn't write, exit with C-flag
AF82            543          ; = 1.
AF82            544          ;
AF82 AD BB B5    545          ^2 lda FMOPCOD
AF85 C9 04       546          cmp #FMWRITCD
AF87 F0 02       547          beq >3
AF89            548          ;

```

```

AF89 38          549          sec
AF8A 60          550          rts
AF8B          551          ;
AF8B          552          ;
AF8B          553          ; Writing to disk, simply allocate another sector for
AF8B          554          ; the T/S list and continue. The relative offsets are
AF8B          555          ; saved in words 6 and 7 of the T/S list.
AF8B          556          ;
AF8B 20 44 B2    557          ^3      jsr ALLOCSEC
AF8E          558          ;
AF8E A0 02       559          ldy #TSSECOFF
AF90          560          ;
AF90 91 42       561          sta (BUFADRZ),Y
AF92 48          562          pha
AF93          563          ;
AF93 88          564          dey
AF94          565          ;
AF94 AD F1 B5    566          lda CURTRACK
AF97 91 42       567          sta (BUFADRZ),Y
AF99 48          568          pha
AF9A          569          ;
AF9A 20 3A AF     570          jsr SETUPRW
AF9D 20 D6 B7    571          jsr ZEROBUFR
AFA0          572          ;
AFA0 A0 05       573          ldy #TSRECOFF
AFA2          574          ;
AFA2 AD DE B5    575          lda RELSLAST
AFA5 91 42       576          sta (BUFADRZ),Y
AFA7          577          ;
AFA7 C8          578          iny
AFA8          579          ;
AFA8 AD DF B5    580          lda RELSLAST+1
AFAB 91 42       581          sta (BUFADRZ),Y
AFAD          582          ;
AFAD 68          583          pla
AFAE AA          584          tax
AFAF          585          ;
AFAF 68          586          pla
AFB0 A8          587          tay
AFB1          588          ;
AFB1 A9 02       589          lda #RWTSWRIT
AFB3 D0 02       590          bne >4
AFB5          591          ;
AFB5          592          ;
AFB5          593          ; If A-reg = 1, read old T/S list. If A-reg = 2, write new
AFB5          594          ; sector for T/S list.
AFB5          595          ;
AFB5          596          OLDRWTS:
AFB5 A9 01       597          lda #RWTSREAD
AFB7          598          ;
AFB7 8E D3 B5    599          ^4      stx CURTSTS
AFBA 8C D4 B5    600          sty CURTSS
AFBD          601          ;
AFBD 20 52 B0    602          jsr RWTSDRVR
AFC0          603          ;
AFC0          604          ;
AFC0          605          ; Compute the relative sector number of the last sector
AFC0          606          ; represented in this T/S list and store in work area.
AFC0          607          ;
AFC0 A0 05       608          ldy #TSRECOFF
AFC2          609          ;

```



```

AFC2 B1 42      610      lda (BUFADRZ),Y
AFC4 8D DC B5   611      sta RELSFRST
AFC7          612      ;
AFC7 18         613      clc
AFC8          614      ;
AFC8 6D DA B5   615      adc SECPERTS
AFCB 8D DE B5   616      sta RELSLAST
AFCE          617      ;
AFCE C8         618      iny
AFCF          619      ;
AFCF B1 42      620      lda (BUFADRZ),Y
AFD1 8D DD B5   621      sta RELSFRST+1
AFD4          622      ;
AFD4 6D DB B5   623      adc SECPERTS+1
AFD7 8D DF B5   624      sta RELSLAST+1
AFDA          625      ;
AFDA 18         626      clc
AFDB 60         627      rts
AFDC          628      ;
AFDC          629      ;
AFDC          630      ; Read a data sector.
AFDC          631      ;
AFDC          632      RDDASEC:
AFDC 20 E4 AF   633      jsr PREPDATA
AFDF          634      ;
AFDF A9 01      635      lda #RWTSREAD
AFE1          636      ;
AFE1 4C 52 B0   637      jmp RWTSDRVR
AFE4          638      ;
AFE4          639      ;
AFE4          640      ; Prepare RWTS IOB for reading a data sector.
AFE4          641      ;
AFE4          642      PREPDATA:
AFE4 AC CB B5   643      ldy DATASADR
AFE7 AD CC B5   644      lda DATASADR+1
AFEA          645      ;
AFEA 8C F0 B7   646      sty USRBUF
AFED 8D F1 B7   647      sta USRBUF+1
AFF0          648      ;
AFF0 AE D6 B5   649      ldx CURDATS
AFF3 AC D7 B5   650      ldy CURDAS
AFF6          651      ;
AFF6 60         652      rts
AFF7          653      ;
AFF7          654      ;
AFF7          655      ; Read or write the VTOC buffer.
AFF7          656      ;
AFF7          657      RDVTOC:
AFF7 A9 01      658      lda #RWTSREAD
AFF9 D0 02      659      bne >1
AFFB          660      ;
AFFB          661      WRTVTOC:
AFFB A9 02      662      lda #RWTSWRIT
AFFD          663      ;
AFFD AC C3 AA   664      ^1 ldy VTOCPADR
B000 8C F0 B7   665      sty USRBUF
B003          666      ;
B003 AC C4 AA   667      ldy VTOCPADR+1
B006 8C F1 B7   668      sty USRBUF+1
B009          669      ;
B009 AE FA B5   670      ldx TRKNUMBR

```

```

B00C A0 00      671      ldy #ZERO
B00E           672      ;
B00E 4C 52 B0   673      jmp RWTSDRVR
B011           674      ;
B011           675      ;
B011           676      ; Read a directory sector.  If C-flag = 0 read first
B011           677      ; directory sector.  If C-flag = 1 read next directory
B011           678      ; sector.  On exit C-flag = 1 if no more sectors to read.
B011           679      ;
B011           680      RDDIRSEC:
B011 08         681      php
B012           682      ;
B012 20 45 B0   683      jsr PRWTSDIR
B015           684      ;
B015 28         685      plp
B016 B0 08      686      bcs >2
B018           687      ;
B018           688      ;
B018           689      ; Read first directory sector.
B018           690      ;
B018 AC BD B3   691      ldy FRSTTS+1
B01B AE BC B3   692      ldx FRSTTS
B01E D0 0A      693      bne >4
B020           694      ;
B020           695      ;
B020           696      ; Read next directory sector.
B020           697      ;
B020 AE BC B4   698      ^2      ldx TSNXTDIR
B023 D0 02      699      bne >3
B025           700      ;
B025           701      ;
B025           702      ; If there are no more directory sectors, exit with C-flag
B025           703      ; set.
B025           704      ;
B025 38         705      sec
B026 60         706      rts
B027           707      ;
B027           708      ;
B027           709      ; Read next directory sector in chain here.
B027           710      ;
B027 AC BD B4   711      ^3      ldy TSNXTDIR+1
B02A           712      ;
B02A 8E 97 B3   713      ^4      stx DIRTS
B02D 8C 98 B3   714      sty DIRTS+1
B030           715      ;
B030 A9 01      716      lda #RWTSREAD
B032           717      ;
B032 20 52 B0   718      jsr RWTSDRVR
B035           719      ;
B035 18         720      clc
B036 60         721      rts
B037           722      ;
B037           723      ;
B037           724      ; Write a directory sector.
B037           725      ;
B037           726      WRTDIRSC:
B037 20 45 B0   727      jsr PRWTSDIR
B03A           728      ;
B03A AE 97 B3   729      ldx DIRTS
B03D AC 98 B3   730      ldy DIRTS+1
B040           731      ;

```

```
B040 A9 02      732      lda #RWTSWRIT
B042           733      ;
B042 4C 52 B0   734      jmp RWTSDRVR
B045           735      ;
B045           736      ;
B045           737      ; Prepare RWTS for directory read/write.
B045           738      ;
B045           739      PRWTSDIR:
B045 AD C5 AA   740      lda DIRPADR
B048 8D F0 B7   741      sta USRBUF
B04B           742      ;
B04B AD C6 AA   743      lda DIRPADR+1
B04E 8D F1 B7   744      sta USRBUF+1
B051           745      ;
B051 60         746      rts
B052           747      ;
B052           748      ;
B052           749      icl "MNGR2A.L"
```

```
LLOAD MNGR2A.L,A$4000
```

```
B052      1          ttl "DOS 3.3 Source Code, MNGR2A.L"
B052      2      ;
B052      3      ;
B052      4      ; MNGR2A.L
B052      5      ;
B052      6      ;
B052      7      ; RWTS driver finishes up the IOB and calls the RWTS
B052      8      ; routine.
B052      9      ;
B052     10      ; A-reg = command, X-reg = track, Y-reg = sector.
B052     11      ;
B052     12      RWTSDRVR:
B052 8E EC B7     13          stx TNUM
B055 8C ED B7     14          sty SNUM
B058     15      ;
B058     16      ;
B058     17      SETCMDCD:
B058 8D F4 B7     18          sta CMDCODE
B05B     19      ;
B05B C9 02       20          cmp #RWTSWRIT
B05D D0 06       21          bne >1
B05F     22      ;
B05F     23      ;
B05F     24      ; Writing to disk, so note this in FLAGS.
B05F     25      ;
B05F 0D D5 B5    26          ora FLAGS
B062 8D D5 B5    27          sta FLAGS
B065     28      ;
B065     29      ;
B065     30      ; Compliment the volume number.
B065     31      ;
B065 AD F9 B5    32      ^1      lda VOLNUMBR
B068 49 FF       33          eor #NEGONE
B06A 8D EB B7    34          sta VOLEXPT
B06D     35      ;
B06D     36      ;
B06D     37      ; Initialize slot, drive, and sector.  TBLTYPE is a
B06D     38      ; mystery.
B06D     39      ;
B06D AD F7 B5    40          lda SLOT16
B070 8D E9 B7    41          sta SNUM16
B073     42      ;
B073 AD F8 B5    43          lda DRVNUMBR
B076 8D EA B7    44          sta DNUM
B079     45      ;
B079 AD E2 B5    46          lda SECTLEN
B07C 8D F2 B7    47          sta BYTCNT
B07F     48      ;
B07F AD E3 B5    49          lda SECTLEN+1
B082 8D F3 B7    50          sta BYTCNT+1
B085     51      ;
B085     52      ;
B085     53      ; This is already set as a define.
B085     54      ;
B085 A9 01       55          lda #1
B087 8D E8 B7    56          sta TBLTYPE
B08A     57      ;
B08A     58      ;
B08A     59      ; Call the RWTS subroutine.
B08A     60      ;
```

```

B08A AC C1 AA      61          ldy RWTSPADR
B08D AD C2 AA      62          lda RWTSPADR+1
B090                63          ;
B090 20 B5 B7      64          jsr CALLRWTS
B093                65          ;
B093                66          ;
B093                67          ; Fiddle with the volume numbers before checking for an
B093                68          ; error.
B093                69          ;
B093                70          ;
B093 AD F6 B7      71          lda VOLFND
B096 8D BF B5      72          sta VOLUME
B099                73          ;
B099 A9 FF          74          lda #NEGONE
B09B 8D EB B7      75          sta VOLEXPT
B09E                76          ;
B09E                77          ;
B09E                78          ; If no error (C-flag = 0), quit.
B09E                79          ;
B09E B0 01          80          bcs >2
B0A0                81          ;
B0A0 60            82          rts
B0A1                83          ;
B0A1                84          ;
B0A1                85          ; Convert RWTS error number to a number usable by the File
B0A1                86          ; Manager.
B0A1                87          ;
B0A1 AD F5 B7      88          ^2      lda ERRRCODE
B0A4                89          ;
B0A4 A0 07          90          ldy #FMVOLERR
B0A6                91          ;
B0A6 C9 20          92          cmp #RWVOLERR
B0A8 F0 08          93          beq >3
B0AA                94          ;
B0AA A0 04          95          ldy #FMPROTER
B0AC                96          ;
B0AC C9 10          97          cmp #RWPROTER
B0AE F0 02          98          beq >3
B0B0                99          ;
B0B0 A0 08          100         ldy #FMDIOERR
B0B2                101         ;
B0B2 98            102         ^3      tya
B0B3                103         ;
B0B3 4C 85 B3      104         jmp SETERROR
B0B6                105         ;
B0B6                106         ;
B0B6                107         ; Read the next data sector if required. Check to see if
B0B6                108         ; the current file position is in the current data sector
B0B6                109         ; buffered up in memory.
B0B6                110         ;
B0B6                111         RDNXTDA:
B0B6 AD E4 B5      112         lda FILEPOSN
B0B9 CD E0 B5      113         cmp RELSLRD
B0BC D0 08          114         bne >1
B0BE                115         ;
B0BE AD E5 B5      116         lda FILEPOSN+1
B0C1 CD E1 B5      117         cmp RELSLRD+1
B0C4 F0 66          118         beq >9
B0C6                119         ;
B0C6                120         ;
B0C6                121         ; If not, see if the current sector needs to be written to

```

```

B0C6      122      ; disk.
B0C6      123      ;
B0C6 20 1D AF 124      ^1      jsr CHKBUF
B0C9      125      ;
B0C9      126      ;
B0C9      127      ; Is the desired file position within the range of sectors
B0C9      128      ; represented by the T/S list currently buffered in memory?
B0C9      129      ;
B0C9 AD E5 B5 130      ^2      lda FILEPOSN+1
B0CC CD DD B5 131      cmp RELSFRST+1
B0CF 90 1C 132      bcc >4
B0D1      133      ;
B0D1 D0 08 134      bne >3
B0D3      135      ;
B0D3 AD E4 B5 136      lda FILEPOSN
B0D6 CD DC B5 137      cmp RELSFRST
B0D9 90 12 138      bcc >4
B0DB      139      ;
B0DB AD E5 B5 140      ^3      lda FILEPOSN+1
B0DE CD DF B5 141      cmp RELSLAST+1
B0E1 90 10 142      bcc >5
B0E3      143      ;
B0E3 D0 08 144      bne >4
B0E5      145      ;
B0E5 AD E4 B5 146      lda FILEPOSN
B0E8 CD DE B5 147      cmp RELSLAST
B0EB 90 06 148      bcc >5
B0ED      149      ;
B0ED      150      ;
B0ED      151      ; Read in new T/S list. C-flag is set or cleared from
B0ED      152      ; the compare statements above. If C-flag = 0 read first
B0ED      153      ; T/S sector. If C-flag = 1 read next T/S sector.
B0ED      154      ;
B0ED 20 5E AF 155      ^4      jsr RDTSLIST
B0F0 90 D7 156      bcc <2
B0F2      157      ;
B0F2 60 158      rts
B0F3      159      ;
B0F3      160      ;
B0F3      161      ; Data was found in current T/S list. Get T/S entry in
B0F3      162      ; this T/S list.
B0F3      163      ;
B0F3 38 164      ^5      sec
B0F4      165      ;
B0F4 AD E4 B5 166      lda FILEPOSN
B0F7 ED DC B5 167      sbc RELSFRST
B0FA      168      ;
B0FA 0A 169      asl
B0FB      170      ;
B0FB 69 0C 171      adc #TSLSTOFF
B0FD A8 172      tay
B0FE      173      ;
B0FE 20 0C AF 174      jsr SELTSBUF
B101      175      ;
B101 B1 42 176      lda (BUFADRZ),Y
B103 D0 0F 177      bne >7
B105      178      ;
B105 AD BB B5 179      lda FMOPCOD
B108 C9 04 180      cmp #FMWRITCD
B10A F0 02 181      beq >6
B10C      182      ;

```

```

B10C 38          183          sec
B10D 60          184          rts
B10E             185          ;
B10E 20 34 B1    186          ^6      jsr ADDDATA
B111             187          ;
B111 4C 20 B1    188          jmp >8
B114             189          ;
B114             190          ;
B114             191          ; Read existing data using current T/S list.
B114             192          ;
B114 8D D6 B5    193          ^7      sta CURDATS
B117             194          ;
B117 C8          195          iny
B118             196          ;
B118 B1 42       197          lda (BUFADRZ),Y
B11A 8D D7 B5    198          sta CURDAS
B11D             199          ;
B11D 20 DC AF    200          jsr RDDASEC
B120             201          ;
B120             202          ;
B120             203          ; Save sector number of sector last read in the work area.
B120             204          ;
B120 AD E4 B5    205          ^8      lda FILEPOSN
B123 8D E0 B5    206          sta RELSLRD
B126             207          ;
B126 AD E5 B5    208          lda FILEPOSN+1
B129 8D E1 B5    209          sta RELSLRD+1
B12C             210          ;
B12C             211          ;
B12C             212          ; Sselect data buffer.
B12C             213          ;
B12C 20 10 AF    214          ^9      jsr SELDABUF
B12F             215          ;
B12F AC E6 B5    216          ldY FILEBYTE
B132             217          ;
B132 18          218          clc
B133 60          219          rts
B134             220          ;
B134             221          ;
B134             222          ; Add a new data sector to the file. Save Y-reg for later.
B134             223          ;
B134             224          ADDATA:
B134 8C 9D B3    225          sty CNTR
B137             226          ;
B137             227          ;
B137             228          ; Allocate a sector for the data.
B137             229          ;
B137 20 44 B2    230          jsr ALLOCSEC
B13A             231          ;
B13A             232          ;
B13A             233          ; Put T/S numbers into T/S list.
B13A             234          ;
B13A AC 9D B3    235          ldY CNTR
B13D C8          236          iny
B13E             237          ;
B13E 91 42       238          sta (BUFADRZ),Y
B140 8D D7 B5    239          sta CURDAS
B143             240          ;
B143 88          241          dey
B144             242          ;
B144 AD F1 B5    243          lda CURTRACK

```

```

B147 91 42      244      sta (BUFADRZ),Y
B149 8D D6 B5   245      sta CURDATS
B14C           246      ;
B14C           247      ;
B14C           248      ; Select data buffer and zero it out.
B14C           249      ;
B14C 20 10 AF   250      jsr SELDABUF
B14F 20 D6 B7   251      jsr ZEROBUFR
B152           252      ;
B152           253      ;
B152           254      ; Set up flags noting that the current data and T/S
B152           255      ; sectors must be written to disk.
B152           256      ;
B152 A9 C0      257      lda #DATAMASK|TSBFRMSK
B154 0D D5 B5   258      ora FLAGS
B157 8D D5 B5   259      sta FLAGS
B15A           260      ;
B15A 60         261      rts
B15B           262      ;
B15B           263      ;
B15B           264      ; The following are various utilities used by many
B15B           265      ; routines.
B15B           266      ;
B15B           267      ; Increment record number and byte offset into the file.
B15B           268      ;
B15B           269      INCREC:
B15B AE EA B5   270      ldx RECNUMBR
B15E 8E BD B5   271      stx RECNUM
B161           272      ;
B161 AE EB B5   273      ldx RECNUMBR+1
B164 8E BE B5   274      stx RECNUM+1
B167           275      ;
B167 AE EC B5   276      ldx BYTEOFFS
B16A AC ED B5   277      ldy BYTEOFFS+1
B16D           278      ;
B16D 8E BF B5   279      stx BYTOFFST
B170 8C C0 B5   280      sty BYTOFFST+1
B173           281      ;
B173 E8         282      inx
B174 D0 01      283      bne >1
B176           284      ;
B176 C8         285      iny
B177           286      ;
B177 CC E9 B5   287      ^1 cpy OPNRCLen+1
B17A D0 11      288      bne >2
B17C           289      ;
B17C EC E8 B5   290      cpx OPNRCLen
B17F D0 0C      291      bne >2
B181           292      ;
B181 A2 00      293      ldx #ZERO
B183 A0 00      294      ldy #ZERO
B185           295      ;
B185 EE EA B5   296      inc RECNUMBR
B188 D0 03      297      bne >2
B18A           298      ;
B18A EE EB B5   299      inc RECNUMBR+1
B18D           300      ;
B18D 8E EC B5   301      ^2 stx BYTEOFFS
B190 8C ED B5   302      sty BYTEOFFS+1
B193           303      ;
B193 60         304      rts

```



```

B194          305 ;
B194          306 ;
B194          307 ; Increment file position offset.
B194          308 ;
B194          309 INCPOS:
B194 EE E6 B5 310          inc FILEBYTE
B197 D0 08    311          bne >1
B199          312 ;
B199 EE E4 B5 313          inc FILEPOSN
B19C D0 03    314          bne >1
B19E          315 ;
B19E EE E5 B5 316          inc FILEPOSN+1
B1A1          317 ;
B1A1 60       318 ^1      rts
B1A2          319 ;
B1A2          320 ;
B1A2          321 ; Copy and advance range address.
B1A2          322 ;
B1A2          323 MOVRANG:
B1A2 AC C3 B5 324          ldy DATADR
B1A5 AE C4 B5 325          ldx DATADR+1
B1A8          326 ;
B1A8 84 42    327          sty BUFADRZ
B1AA 86 43    328          stx BUFADRZ+1
B1AC          329 ;
B1AC EE C3 B5 330          inc DATADR
B1AF D0 03    331          bne >1
B1B1          332 ;
B1B1 EE C4 B5 333          inc DATADR+1
B1B4          334 ;
B1B4 60       335 ^1      rts
B1B5          336 ;
B1B5          337 ;
B1B5          338 ; Decrement range address.
B1B5          339 ;
B1B5          340 DECRNG:
B1B5 AC C1 B5 341          ldy BYTRANGE
B1B8 D0 08    342          bne >1
B1BA          343 ;
B1BA AE C2 B5 344          ldx BYTRANGE+1
B1BD F0 07    345          beq >2
B1BF          346 ;
B1BF CE C2 B5 347          dec BYTRANGE+1
B1C2          348 ;
B1C2 CE C1 B5 349 ^1      dec BYTRANGE
B1C5          350 ;
B1C5 60       351          rts
B1C6          352 ;
B1C6 4C 7F B3 353 ^2      jmp NOERROR
B1C9          354 ;
B1C9          355 ;
B1C9          356 ; Locate a directory filename. Read VTOC to get track and
B1C9          357 ; sector number of first directory sector. On exit C-flag
B1C9          358 ; = 0 if filename found.
B1C9          359 ;
B1C9          360 LCDIRENT:
B1C9 20 F7 AF 361          jsr RDVTOC
B1CC          362 ;
B1CC AD C3 B5 363          lda FNADR
B1CF 85 42    364          sta BUFADRZ
B1D1          365 ;

```

```

B1D1 AD C4 B5      366          lda FNADR+1
B1D4 85 43         367          sta BUFADRZ+1
B1D6              368          ;
B1D6              369          ;
B1D6              370          ; Set up for two passes through the directory.
B1D6              371          ;
B1D6 A9 01         372          lda #1
B1D8              373          ;
B1D8 8D 9D B3      374          ^1      sta CNTR
B1DB              375          ;
B1DB              376          ;
B1DB              377          ; Set up index into VTOC for the directory track.
B1DB              378          ;
B1DB A9 00         379          lda #ZERO
B1DD 8D D8 B5      380          sta DIRSECIX
B1E0              381          ;
B1E0 18            382          clc
B1E1              383          ;
B1E1              384          ;
B1E1              385          ; Read in the current directory sector.
B1E1              386          ;
B1E1 EE D8 B5      387          ^2      inc DIRSECIX
B1E4              388          ;
B1E4              389          ;
B1E4              390          ; If C-flag = 1 there are no more sectors to read.
B1E4              391          ;
B1E4 20 11 B0      392          jsr RDDIRSEC
B1E7 B0 51         393          bcs >9
B1E9              394          ;
B1E9              395          ;
B1E9              396          ; Check for end of directory (if first byte is zero) or a
B1E9              397          ; deleted file (if $FF).
B1E9              398          ;
B1E9 A2 00         399          ldx #ZERO
B1EB              400          ;
B1EB 8E 9C B3      401          ^3      stx DIRINDX
B1EE              402          ;
B1EE BD C6 B4      403          lda TSTRACK,X
B1F1              404          ;
B1F1 F0 1F         405          beq >6
B1F3 30 22         406          bmi >7
B1F5              407          ;
B1F5 A0 00         408          ldy #ZERO
B1F7              409          ;
B1F7 E8            410          inx
B1F8 E8            411          inx
B1F9              412          ;
B1F9              413          ;
B1F9              414          ; Compare the filename against the current directory entry.
B1F9              415          ;
B1F9 E8            416          ^4      inx
B1FA              417          ;
B1FA B1 42         418          lda (BUFADRZ),Y
B1FC DD C6 B4      419          cmp FILNAME-3,X
B1FF D0 0A         420          bne >5
B201              421          ;
B201 C8            422          iny
B202              423          ;
B202 C0 1E         424          cpy #NAME SIZE
B204 D0 F3         425          bne <4
B206              426          ;

```

```

B206      427      ;
B206      428      ; If found, return with index into sector in the X-reg and
B206      429      ; clear the C-flag.
B206      430      ;
B206 AE 9C B3      431      ldx DIRINDX
B209      432      ;
B209 18          433      clc
B20A 60          434      rts
B20B      435      ;
B20B      436      ;
B20B      437      ; If filename doesn't match current entry, on to the next
B20B      438      ; entry.
B20B      439      ;
B20B 20 30 B2     440      ^5      jsr NXTDIREN
B20E      441      ;
B20E 90 DB       442      bcc <3
B210 B0 CF       443      bcs <2
B212      444      ;
B212 AC 9D B3     445      ^6      ldy CNTR
B215 D0 C1       446      bne <1
B217      447      ;
B217 AC 9D B3     448      ^7      ldy CNTR
B21A D0 EF       449      bne <5
B21C      450      ;
B21C      451      ;
B21C      452      ; Copy filename to directory entry.
B21C      453      ;
B21C      454      COPYFNAM:
B21C A0 00       455      ldy #ZERO
B21E      456      ;
B21E E8          457      inx
B21F E8          458      inx
B220      459      ;
B220 E8          460      ^8      inx
B221      461      ;
B221 B1 42       462      lda (BUFADRZ),Y
B223 9D C6 B4     463      sta FILNAME-3,X
B226      464      ;
B226 C8          465      iny
B227      466      ;
B227 C0 1E       467      cpy #NAME SIZE
B229 D0 F5       468      bne <8
B22B      469      ;
B22B AE 9C B3     470      ldx DIRINDX
B22E      471      ;
B22E 38          472      sec
B22F 60          473      rts
B230      474      ;
B230      475      ;
B230      476      ; Move on to the next directory entry. On exit C-flag = 0
B230      477      ; entry found and C-flag = 1 if no more entries.
B230      478      ;
B230      479      NXTDIREN:
B230 18          480      clc
B231      481      ;
B231 AD 9C B3     482      lda DIRINDX
B234 69 23       483      adc #VTOCENSZ
B236      484      ;
B236 AA          485      tax
B237 E0 F5       486      cpx #VTOCEND
B239      487      ;

```

```
B239 60          488          rts
B23A          489          ;
B23A A9 00      490 ^9          lda #ZERO
B23C          491          ;
B23C AC 9D B3   492          ldy CNTR
B23F D0 97      493          bne <1
B241          494          ;
B241 4C 77 B3   495          jmp DSKFULL
B244          496          ;
B244          497          ;
B244          498          ; Allocate a disk sector.
B244          499          ;
B244          500          ; This routine finds a track in the track bit map stored
B244          501          ; in memory that has some empty sectors left.  ALLOCSEC
B244          502          ; then allocates the rest of those sectors and gives them
B244          503          ; to the requesting file.
B244          504          ;
B244          505          ; First, see if there is a track already allocated to this
B244          506          ; file.
B244          507          ;
B244          508          ALLOCSEC:
B244 AD F1 B5   509          lda CURTRACK
B247 F0 21      510          beq >4
B249          511          ;
B249          512          ;
B249          513          ; If so, decrement sector count to find the next possible
B249          514          ; sector number.
B249          515          ;
B249 CE F0 B5   516 ^0          dec NEXTSECR
B24C 30 17      517          bmi >3
B24E          518          ;
B24E          519          ;
B24E          520          ; If a valid sector, rotate the bit map to see if it is
B24E          521          ; free.  For C-flag = 0, clear sector.
B24E          522          ;
B24E 18         523          clc
B24F          524          ;
B24F A2 04      525          ldx #FTYPE-SECBTMAP
B251          526          ;
B251 3E F1 B5   527 ^1          rol SECBTMAP-1,X
B254          528          ;
B254 CA         529          dex
B255 D0 FA      530          bne <1
B257          531          ;
B257          532          ;
B257          533          ; If not free, repeat for next sector.
B257          534          ;
B257 90 F0      535          bcc <0
B259          536          ;
B259          537          ;
B259          538          ; If free, allocate and increment file size by one.
B259          539          ;
B259 EE EE B5   540          inc SECCNT
B25C D0 03      541          bne >2
B25E          542          ;
B25E EE EF B5   543          inc SECCNT+1
B261          544          ;
B261 AD F0 B5   545 ^2          lda NEXTSECR
B264          546          ;
B264 60         547          rts
B265          548          ;
```

```

B265          549 ;
B265          550 ; Indicate that a track is not currently allocated.
B265          551 ;
B265 A9 00    552 ^3      lda #ZERO
B267 8D F1 B5 553          sta CURTRACK
B26A          554 ;
B26A          555 ;
B26A          556 ; Reset allocation flag to allow a complete scan of the
B26A          557 ; bit map for available sectors.
B26A          558 ;
B26A A9 00    559 ^4      lda #ZERO
B26C 8D 9E B3 560          sta ALLCFLG
B26F          561 ;
B26F 20 F7 AF 562          jsr RDVTOC
B272          563 ;
B272          564 ;
B272          565 ; Add allocation direction to NXTTOALLC and see if we are
B272          566 ; at track zero, or past track 34.
B272          567 ;
B272 18       568 ^5      clc
B273          569 ;
B273 AD EB B3 570          lda NXTTOALC
B276 6D EC B3 571          adc ALLCDIR
B279 F0 09     572          beq >6
B27B          573 ;
B27B CD EF B3 574          cmp NUMTRKS
B27E 90 14     575          bcc >8
B280          576 ;
B280          577 ;
B280          578 ; If past track 34, set allocation direction to -1. If
B280          579 ; this is the second time reaching track 0 exit with "DISK
B280          580 ; FULL" error.
B280          581 ;
B280 A9 FF     582          lda #ALOCBKWD
B282 D0 0A     583          bne >7
B284          584 ;
B284 AD 9E B3 585 ^6      lda ALLCFLG
B287 D0 37     586          bne >2
B289          587 ;
B289 A9 01     588          lda #ALOCFRWD
B28B 8D 9E B3 589          sta ALLCFLG
B28E          590 ;
B28E          591 ;
B28E          592 ; Start at the directory track and work backwards.
B28E          593 ;
B28E 8D EC B3 594 ^7      sta ALLCDIR
B291          595 ;
B291 18       596          clc
B292 69 11     597          adc #VTOCTRK
B294          598 ;
B294 8D EB B3 599 ^8      sta NXTTOALC
B297 8D F1 B5 600          sta CURTRACK
B29A A8       601          tay ; unnecessary
B29B          602 ;
B29B 0A       603          asl
B29C 0A       604          asl
B29D A8       605          tay
B29E          606 ;
B29E A2 04    607          ldx #FTYPE-SECBTMAP
B2A0          608 ;
B2A0          609 ;

```

```

B2A0      610 ; C-flag = 0 if no sectors available and C-flag = 1 if
B2A0      611 ; sectors are available.
B2A0      612 ;
B2A0 18    613         clc
B2A1      614 ;
B2A1 B9 F6 B3 615 ^9      lda BITMAP+3,Y
B2A4 9D F1 B5 616         sta SECBTMAP-1,X
B2A7 F0 06    617         beq >1
B2A9      618 ;
B2A9 38      619         sec
B2AA      620 ;
B2AA A9 00    621         lda #ZERO
B2AC 99 F6 B3 622         sta BITMAP+3,Y
B2AF      623 ;
B2AF 88      624 ^1      dey
B2B0      625 ;
B2B0 CA      626         dex
B2B1 D0 EE    627         bne <9
B2B3      628 ;
B2B3 90 BD    629         bcc <5
B2B5      630 ;
B2B5      631 ;
B2B5      632 ; Allocated an entire track so write out the VTOC so other
B2B5      633 ; files don't use them.
B2B5      634 ;
B2B5 20 FB AF 635         jsr WRTVTOC
B2B8      636 ;
B2B8 AD F0 B3 637         lda NUMSCTRS
B2BB 8D F0 B5 638         sta NEXTSECR
B2BE D0 89    639         bne <0
B2C0      640 ;
B2C0 4C 77 B3 641 ^2      jmp DSKFULL
B2C3      642 ;
B2C3      643 ;
B2C3      644 ; Release any allocated sectors that were not used. If
B2C3      645 ; C-flag = 0 allocate the sector.
B2C3      646 ;
B2C3      647 RLSALLC:
B2C3 AD F1 B5 648         lda CURTRACK
B2C6 D0 01    649         bne >1
B2C8      650 ;
B2C8 60      651         rts
B2C9      652 ;
B2C9 48      653 ^1      pha
B2CA      654 ;
B2CA 20 F7 AF 655         jsr RDVTOC
B2CD      656 ;
B2CD AC F0 B5 657         ldy NEXTSECR
B2D0      658 ;
B2D0 68      659         pla
B2D1      660 ;
B2D1 18      661         clc
B2D2      662 ;
B2D2 20 DD B2 663         jsr RORBITMP
B2D5      664 ;
B2D5 A9 00    665         lda #ZERO
B2D7 8D F1 B5 666         sta CURTRACK
B2DA      667 ;
B2DA 4C FB AF 668         jmp WRTVTOC
B2DD      669 ;
B2DD      670 ;

```

```
B2DD      671  ; Rotate the bit map right one bit (16 - Y-reg) times with
B2DD      672  ; initial C-flag clearing or setting the desired bit.
B2DD      673  ;
B2DD      674  ; A-reg = track, Y-reg = sector, C-flag = 0 to allocate or
B2DD      675  ; C-flag = 1 to deallocate.
B2DD      676  ;
B2DD      677  RORBITMP:
B2DD A2 FC      678          ldx #$FC
B2DF      679  ;
B2DF 7E F6 B4    680  ^1      ror SECBTMAP-$FC,X
B2E2      681  ;
B2E2 E8          682          inx
B2E3 D0 FA      683          bne <1
B2E5      684  ;
B2E5 C8          685          iny
B2E6      686  ;
B2E6 CC F0 B3    687          cpy NUMSCTRS
B2E9 D0 F2      688          bne RORBITMP
B2EB      689  ;
B2EB 0A          690          asl
B2EC 0A          691          asl
B2ED      692  ;
B2ED A8          693          tay
B2EE F0 0F      694          beq >3
B2F0      695  ;
B2F0 A2 04      696          ldx #FTYPE-SECBTMAP
B2F2      697  ;
B2F2 BD F1 B5    698  ^2      lda SECBTMAP-1,X
B2F5 19 F6 B3    699          ora BITMAP+3,Y
B2F8 99 F6 B3    700          sta BITMAP+3,Y
B2FB      701  ;
B2FB 88          702          dey
B2FC      703  ;
B2FC CA          704          dex
B2FD D0 F3      705          bne <2
B2FF      706  ;
B2FF 60          707  ^3      rts
B300      708  ;
B300      709  ;
B300      710          icl "MNGR2B.L"
```

LLOAD MNGR2B.L,A\$4000

```
B300          1          ttl "DOS 3.3 Source Code, MNGR2B.L"
B300          2          ;
B300          3          ;
B300          4          ; MNGR2B.L
B300          5          ;
B300          6          ;
B300          7          ; Calculate file position.
B300          8          ;
B300          9          ; Given a record number, a file length, and a byte offset,
B300         10          ; this routine calculates the offset into the file to find
B300         11          ; the desired record. Do the loop for 16 times.
B300         12          ;
B300         13          CALPOSN:
B300 AD BD B5   14          lda RECNUM
B303 8D E6 B5   15          sta FILEBYTE
B306 8D EA B5   16          sta RECNUMBR
B309          17          ;
B309 AD BE B5   18          lda RECNUM+1
B30C 8D E4 B5   19          sta FILEPOSN
B30F 8D EB B5   20          sta RECNUMBR+1
B312          21          ;
B312 A9 00      22          lda #ZERO
B314 8D E5 B5   23          sta FILEPOSN+1
B317          24          ;
B317 A0 10      25          ldy #16
B319          26          ;
B319 AA         27          ^1 tax
B31A          28          ;
B31A AD E6 B5   29          lda FILEBYTE
B31D 4A         30          lsr
B31E B0 03      31          bcs >2
B320          32          ;
B320 8A         33          txa
B321 90 0E      34          bcc >3
B323          35          ;
B323 18         36          ^2 clc
B324          37          ;
B324 AD E5 B5   38          lda FILEPOSN+1
B327 6D E8 B5   39          adc OPNRCLN
B32A 8D E5 B5   40          sta FILEPOSN+1
B32D          41          ;
B32D 8A         42          txa
B32E 6D E9 B5   43          adc OPNRCLN+1
B331          44          ;
B331 6A         45          ^3 ror
B332          46          ;
B332 6E E5 B5   47          ror FILEPOSN+1
B335 6E E4 B5   48          ror FILEPOSN
B338 6E E6 B5   49          ror FILEBYTE
B33B          50          ;
B33B 88         51          dey
B33C D0 DB      52          bne <1
B33E          53          ;
B33E 18         54          clc
B33F          55          ;
B33F AD BF B5   56          lda BYTOFFST
B342 8D EC B5   57          sta BYTEOFFS
B345          58          ;
B345 6D E6 B5   59          adc FILEBYTE
B348 8D E6 B5   60          sta FILEBYTE
```



```
B34B          61 ;
B34B AD C0 B5 62      lda BYTOFFST+1
B34E 8D ED B5 63      sta BYTEOFFS+1
B351          64 ;
B351 6D E4 B5 65      adc FILEPOSN
B354 8D E4 B5 66      sta FILEPOSN
B357 90 03     67      bcc >4
B359          68 ;
B359 EE E5 B5 69      inc FILEPOSN+1
B35C          70 ;
B35C 60       71      ^4      rts
B35D          72 ;
B35D          73 ;
B35D          74      dfs 2,ZERO
B35F          75 ;
B35F          76 ;
B35F          77 ; Error messages.
B35F          78 ;
B35F          79 LNOTAVL:
B35F A9 01     80      lda #EMSOFF01-MSGOFFS
B361 D0 22     81      bne SETERROR
B363          82 ;
B363          83 RANGERR:
B363 A9 02     84      lda #EMSOFF02-MSGOFFS
B365 D0 1E     85      bne SETERROR
B367          86 ;
B367          87 RANGERR2:
B367 A9 03     88      lda #EMSOFF03-MSGOFFS
B369 D0 1A     89      bne SETERROR
B36B          90 ;
B36B A9 04     91      lda #EMSOFF04-MSGOFFS
B36D D0 16     92      bne SETERROR
B36F          93 ;
B36F          94 ENDATA:
B36F A9 05     95      lda #EMSOFF05-MSGOFFS
B371 D0 12     96      bne SETERROR
B373          97 ;
B373          98 FNOTFND:
B373 A9 06     99      lda #EMSOFF06-MSGOFFS
B375 D0 0E    100      bne SETERROR
B377         101 ;
B377         102 DSKFULL:
B377 4C ED BF 103      jmp DISKFULL
B37A         104 ;
B37A EA      105      nop
B37B         106 ;
B37B         107 FILELOCK:
B37B A9 0A    108      lda #EMSOFF10-MSGOFFS
B37D D0 06    109      bne SETERROR
B37F         110 ;
B37F         111 NOERROR:
B37F AD C5 B5 112      lda RTNCODE
B382         113 ;
B382 18      114      clc
B383 90 01    115      bcc >1
B385         116 ;
B385         117 ;
B385         118 ; Standard error entry point.
B385         119 ;
B385         120 SETERROR:
B385 38      121      sec
```

```

B386          122 ;
B386 08        123 ^1      php
B387          124 ;
B387 8D C5 B5  125      sta RTNCODE
B38A          126 ;
B38A A9 00     127      lda #ZERO
B38C 85 48     128      sta IOBADR
B38E          129 ;
B38E 20 7E AE  130      jsr SAVFMW
B391          131 ;
B391 28        132      plp
B392          133 ;
B392 AE 9B B3  134      ldx STKSAVE
B395 9A        135      txs
B396          136 ;
B396 60        137      rts
B397          138 ;
B397          139 ;
B397          140 ; Working variables.
B397          141 ;
B397 11 0F     142 DIRTS    hex 110F          ; directory track/sector
B399          143          dfs 2,ZERO          ; not used
B39B          144 STKSAVE   dfs 1,ZERO          ; stack save
B39C          145 DIRINDX   dfs 1,ZERO          ; directory index
B39D          146 CNTR      dfs 1,1            ; counter
B39E          147 ALLCFLG   dfs 2,ZERO          ; allocate flag
B3A0 00 00 FF  148 FREEMASK hex 0000FFFF        ; free sector mask
B3A3 FF
B3A4 01 0A 64  149 DECTBL   hex 010A64          ; decimal lookup table
B3A7 D4 C9 C1  150 FTTBL    asc "TIABSRAB"        ; file type table
B3AA C2 D3 D2
B3AD C1 C2
B3AF A0 C5 CD  151 DISKVOL  asc " EMULOV KSID"    ; DISK VOLUME
B3B2 D5 CC CF
B3B5 D6 A0 CB
B3B8 D3 C9 C4
B3BB          152 DSKVOLND:
B3BB          153 ;
B3BB          154 ;
B3BB          155 ; VTOC structure showing default values.
B3BB          156 ;
B3BB          157 VTOCSB    dfs 1,4              ; VTOC structure block
B3BC 11 0F     158 FRSTTS    hex 110F          ; catalog track and sector
B3BE          159 DOSRLS    dfs 1,4              ; release number
B3BF          160          dfs 2,ZERO
B3C1          161 DSKVOL    dfs 1,ZERO          ; volume number
B3C2          162          dfs 32,ZERO
B3E2 7A        163 NUMTSENT  hex 7A              ; T/S pairs in a sector
B3E3          164          dfs 8,ZERO
B3EB          165 NXTTOALC  dfs 1,17            ; next sector to allocate
B3EC          166 ALLCDIR   dfs 1,1            ; allocation direction
B3ED          167          dfs 2,ZERO
B3EF          168 NUMTRKS   dfs 1,35            ; tracks per disk
B3F0          169 NUMSCTRS  dfs 1,16            ; sectors per track
B3F1 00 01     170 BYTPRSEC  hex 0001          ; bytes per sector
B3F3          171 BITMAP     dfs PAGE SIZE-* -VTOCSB,ZERO ; allocation bit map
B4BB          172 ;
B4BB          173 ;
B4BB          174 ; Catalog sector showing first entry.
B4BB          175 ;
B4BB          176 DIRSECBF  dfs 1,2              ; catalog structure block

```

```

B4BC      177  TSNXTDIR  dfs 2,ZERO      ; next track and sector
B4BE      178          dfs 8,ZERO
B4C6      179  TSTRACK  dfs 1,ZERO      ; file T/S track
B4C7      180  TSSECTOR dfs 1,ZERO      ; file T/S sector
B4C8      181  FILTYP  dfs 1,ZERO      ; file type
B4C9      182  FILNAME  dfs 30,ZERO     ; file name
B4E7      183  FILESIZE dfs 2,ZERO     ; file size
B4E9      184  ;
B4E9      185          dfs PAGESIZE-*-DIRSECBF,ZERO
B5BB      186  ;
B5BB      187  ;
B5BB      188  ; File Manager parameter list.
B5BB      189  ;
B5BB      190  FMOPCOD  dfs 1,ZERO      ; FM opcode
B5BC      191  SUBCODE  dfs 1,ZERO      ; FM subcode
B5BD      192  RECNUM   dfs 2,ZERO      ; record number
B5BF      193  BYTOFFST:
B5BF      194  VOLUME   dfs 1,ZERO      ; volume
B5C0      195  DRIVE    dfs 1,ZERO      ; drive
B5C1      196  BYTRANGE:
B5C1      197  SLOT     dfs 1,ZERO      ; slot
B5C2      198  FILETYPE dfs 1,ZERO      ; file type
B5C3      199  DATADR:
B5C3      200  DATBYTE:
B5C3      201  FNADR     dfs 2,ZERO      ; file name address
B5C5      202  RTNCODE   dfs 2,ZERO      ; return code
B5C7      203  WBADR     dfs 2,ZERO      ; work buffer address
B5C9      204  TSLSTADR  dfs 2,ZERO      ; T/S last address
B5CB      205  DATASADR  dfs 2,ZERO      ; data buffer address
B5CD      206          dfs 4,ZERO      ; not used
B5D1      207  ;
B5D1      208  ;
B5D1      209  ; File Manager work area.
B5D1      210  ;
B5D1      211  FTSTS     dfs 1,ZERO      ; first T/S track
B5D2      212  FTSS      dfs 1,ZERO      ; first T/S sector
B5D3      213  CURTSTS   dfs 1,ZERO      ; current T/S track
B5D4      214  CURTSS    dfs 1,ZERO      ; current T/S sector
B5D5      215  FLAGS     dfs 1,ZERO      ; current disk activity flag
B5D6      216  CURDATS   dfs 1,ZERO      ; current data track
B5D7      217  CURDAS    dfs 1,ZERO      ; current data sector
B5D8      218  DIRSECIX  dfs 1,ZERO      ; directory sector index
B5D9      219  DIRBYTIX  dfs 1,ZERO      ; directory byte index
B5DA      220  SECPERTS   dfs 2,ZERO      ; T/S entries in a sector
B5DC      221  RELSFRST   dfs 2,ZERO      ; relative sector to first
B5DE      222  RELSLAST   dfs 2,ZERO      ; relative sector to last
B5E0      223  RELSLRD    dfs 2,ZERO      ; relative sector just read
B5E2      224  SECTLEN    dfs 2,ZERO      ; sector size in bytes
B5E4      225  FILEPOSN   dfs 2,ZERO      ; file position
B5E6      226  FILEBYTE   dfs 2,ZERO      ; file byte
B5E8      227  OPNRCLN    dfs 2,ZERO      ; file open record length
B5EA      228  RECNUMBR   dfs 2,ZERO      ; record number
B5EC      229  BYTEOFFS   dfs 2,ZERO      ; byte offset
B5EE      230  SECCNT     dfs 2,ZERO      ; sector count
B5F0      231  NEXTSECR   dfs 1,ZERO      ; next sector
B5F1      232  CURTRACK   dfs 1,ZERO      ; current track
B5F2      233  SECBTMAP   dfs 4,ZERO      ; sector bit map
B5F6      234  FTYPE      dfs 1,ZERO      ; file type
B5F7      235  SLOT16     dfs 1,ZERO      ; slot * 16
B5F8      236  DRVNUMBR   dfs 1,ZERO      ; drive number
B5F9      237  VOLNUMBR   dfs 1,ZERO      ; volume number

```

```

B5FA      238  TRKNUMBR dfs 1,ZERO          ; track number
B5FB      239          dfs 3,ZERO          ; not used
B5FE      240  FMWAEND:
B5FE      241          dfs 2,ZERO          ; not used
B600      242  ;
B600      243  ;
B600      244  BOOTCODE:
B600      245  ;
B600      246  ; Compile this code at $800.
B600      247  ;
B600      248  ; Sector to Address map:
B600      249  ;
B600      250  ; Track Sector Address      Track Sector Address
B600      251  ;   00      0      $B600      01      3      $A400
B600      252  ;   00      1      $B700      01      4      $A500
B600      253  ;   00      2      $B800      01      5      $A600
B600      254  ;   00      3      $B900      01      6      $A700
B600      255  ;   00      4      $BA00      01      7      $A800
B600      256  ;   00      5      $BB00      01      8      $A900
B600      257  ;   00      6      $BC00      01      9      $AA00
B600      258  ;   00      7      $BD00      01      A      $AB00
B600      259  ;   00      8      $BE00      01      B      $AC00
B600      260  ;   00      9      $BF00      01      C      $AD00
B600      261  ;   00      A      --      01      D      $AE00
B600      262  ;   00      B      --      01      E      $AF00
B600      263  ;   00      C      $9D00      01      F      $B000
B600      264  ;   00      D      $9E00      02      0      $B100
B600      265  ;   00      E      $9F00      02      1      $B200
B600      266  ;   00      F      $A000      02      2      $B300
B600      267  ;   01      0      $A100      02      3      $B400
B600      268  ;   01      1      $A200      02      4      $B500
B600      269  ;   01      2      $A300
B600      270  ;
B600      271  ;           phs PAGE08
0800      272  ;
0800      273  ; Start of Boot Stage 1. The first value is used to verify
0800      274  ; this code has been read in correctly. The first time
0800      275  ; this code runs BUFRADRZ+1 will equal 9 ($900).
0800      276  ;
0800      277  BOOTBGN:
0800      278          dfs 1,1
0801      279  ;
0801      280  ;
0801      281  ; Check if first time if BUFRADRZ+1 equals 9.
0801      282  ;
0801 A5 27      283          lda BUFRADRZ+1
0803 C9 09      284          cmp #9
0805 D0 18      285          bne >1
0807      286  ;
0807      287  ;
0807      288  ; Determine address of BOOTFW from slot number.
0807      289  ;
0807 A5 2B      290          lda SLOT16Z
0809      291  ;
0809 4A      292          lsr
080A 4A      293          lsr
080B 4A      294          lsr
080C 4A      295          lsr
080D      296  ;
080D 09 C0      297          ora /BOOTFW
080F 85 3F      298          sta BOOTADRZ+1

```

```

0811      299 ;
0811 A9 5C      300      lda #BOOTFW
0813 85 3E      301      sta BOOTADRZ
0815      302 ;
0815      303 ;
0815      304 ; Set up to read 10 sectors starting at sector 1, and save
0815      305 ; data starting at $BF00 to $B600.
0815      306 ;
0815 18      307      clc
0816      308 ;
0816 AD FE 08    309      lda BOOTADR
0819 6D FF 08    310      adc BOOTPGS
081C 8D FE 08    311      sta BOOTADR
081F      312 ;
081F AE FF 08    313 ^1      ldx BOOTPGS
0822 30 15      314      bmi >2
0824      315 ;
0824 BD 4D 08    316      lda INTRLTBL,X
0827 85 3D      317      sta ROMSECTR
0829      318 ;
0829 CE FF 08    319      dec BOOTPGS
082C      320 ;
082C AD FE 08    321      lda BOOTADR
082F 85 27      322      sta BUFRADRZ+1
0831      323 ;
0831 CE FE 08    324      dec BOOTADR
0834      325 ;
0834 A6 2B      326      ldx SLOT16Z
0836      327 ;
0836 6C 3E 00    328      jmp (BOOTADRZ)
0839      329 ;
0839      330 ;
0839      331 ; BOOTADR will now be at $B500. Next boot phase begins at
0839      332 ; $B700. Initialize the Monitor and begin next boot phase.
0839      333 ;
0839 EE FE 08    334 ^2      inc BOOTADR
083C EE FE 08    335      inc BOOTADR
083F      336 ;
083F 20 89 FE    337      jsr SETKBD
0842 20 93 FE    338      jsr SETVID
0845 20 2F FB    339      jsr INIT
0848      340 ;
0848 A6 2B      341      ldx SLOT16Z
084A      342 ;
084A 6C FD 08    343      jmp (BOOTJMP)
084D      344 ;
084D      345 ;
084D      346 ; Sector interleave remapping table.
084D      347 ;
084D 00 0D 0B    348 INTRLTBL hex 000D0B0907050301
0850 09 07 05
0853 03 01
0855 0E 0C 0A    349      hex 0E0C0A080604020F
0858 08 06 04
085B 02 0F
085D      350 ;
085D      351 BOOTEND:
085D      352 ;
085D      353 ;
085D      354 ; Return to DOS image.
085D      355 ;

```

```

085D          356          phs BOOTCODE+BOOTEND-BOOTBGN
B65D          357          ;
B65D          358          ;
B65D          359          ; The following are patches for APPEND and VERIFY.
B65D          360          ;
B65D          361          PATCHBGN:
B65D          362          ;
B65D          363          APPFLG    dfs 1,0
B65E          364          ;
B65E          365          APNDPTCH:
B65E 20 64 A7 366          jsr LOCBUF
B661 B0 08    367          bcs >1
B663          368          ;
B663 A9 00    369          lda #ZERO
B665 A8       370          tay
B666          371          ;
B666 8D 5D B6 372          sta APPFLG
B669 91 40    373          sta (FILEBUFZ),Y
B66B          374          ;
B66B AD C5 B5 375          ^1    lda RTNCODE
B66E          376          ;
B66E 4C D2 A6 377          jmp DOERROR
B671          378          ;
B671          379          ;
B671          380          ; From DOAPND, exit to HBA84 instead of RWCOMMN2.
B671          381          ;
B671          382          APTCH2:
B671 AD 5D B6 383          lda APPFLG
B674 F0 08    384          beq >1
B676          385          ;
B676 EE BD B5 386          inc RECNUM
B679 D0 03    387          bne >1
B67B          388          ;
B67B EE BE B5 389          inc RECNUM+1
B67E          390          ;
B67E A9 00    391          ^1    lda #ZERO
B680 8D 5D B6 392          sta APPFLG
B683          393          ;
B683          394          ; jmp RWCOMMN2
B683 4C 84 BA 395          jmp HBA84
B686          396          ;
B686          397          ;
B686          398          VFYPTCH:
B686 8D BC B5 399          sta SUBCODE
B689          400          ;
B689 20 A8 A6 401          jsr FMDRVR
B68C 20 EA A2 402          jsr DOCLOSE
B68F          403          ;
B68F 4C 7D A2 404          jmp DOVERIFY
B692          405          ;
B692          406          ;
B692          407          ; From FMDRVR if RTNCODE is FMEOFERR. If FILEPOSN and
B692          408          ; FILEBYTE are equal, copy RECNUMBR and BYTEOFFS to
B692          409          ; RECNUM and BYTOFFST in FM parameter list. Otherwise
B692          410          ; set APPFLG to NEGONE. Exit through FMDRVR2.
B692          411          ;
B692          412          APTCH3:
B692 A0 13    413          ldy #$13
B694          414          ;
B694 B1 42    415          ^1    lda (BUFADRZ),Y
B696 D0 14    416          bne >4

```

```

B698          417 ;
B698 C8        418      iny
B699          419 ;
B699 C0 17     420      cpy #$17
B69B D0 F7     421      bne <1
B69D          422 ;
B69D A0 19     423      ldy #$19
B69F          424 ;
B69F B1 42     425 ^2      lda (BUFADRZ),Y
B6A1 99 A4 B5  426      sta RECNUM-$19,Y
B6A4          427 ;
B6A4 C8        428      iny
B6A5          429 ;
B6A5 C0 1D     430      cpy #$1D
B6A7 D0 F6     431      bne <2
B6A9          432 ;
B6A9 4C BB A6  433 ^3      jmp FMDRVR2
B6AC          434 ;
B6AC A2 FF     435 ^4      ldx #NEGONE
B6AE 8E 5D B6  436      stx APPFLG
B6B1 D0 F6     437      bne <3
B6B3          438 ;
B6B3          439      dfs 29,ZERO
B6D0          440 ;
B6D0 20 58 FC  441      jsr HOME
B6D3          442 ;
B6D3 A9 C2     443      lda #"B"
B6D5 20 ED FD  444      jsr COUT
B6D8          445 ;
B6D8 A9 01     446      lda #1
B6DA 20 DA FD  447      jsr PRBYTE
B6DD          448 ;
B6DD A9 AD     449      lda #"-"
B6DF 20 ED FD  450      jsr COUT
B6E2          451 ;
B6E2 A9 00     452      lda #ZERO
B6E4 20 DA FD  453      jsr PRBYTE
B6E7          454 ;
B6E7 60        455      rts
B6E8          456 ;
B6E8          457 PATCHEND:
B6E8          458 ;
B6E8          459      dfs BOOTVALS-BOOTEND+PATCHEND-PATCHBGN,ZERO
B6FD          460 ;
B6FD          461 PTCHSTOP:
B6FD          462 ;
B6FD          463 ; Return to BOOT code.
B6FD          464 ;
B6FD          465      phs BOOTVALS
08FD          466 ;
08FD          467 VALSBGN:
08FD          468 ;
08FD 00        469 BOOTJMP hex 00      ; Boot Stage 2 lo address
08FE B6        470 BOOTADR hex B6      ; memory high address
08FF 09        471 BOOTPGS hex 09      ; sector to read on track zero
0900          472 ;
0900          473 VALSEND:
0900          474 ;
0900          475 ;
0900          476 ; Return to DOS code.
0900          477 ;

```

```

0900          478          phs PTCHSTOP+VALSEND-VALSBGN
B700          479          ;
B700          480          ; DOS Boot Stage 2 entry.  Set up USRBUF with $B500 for
B700          481          ; track 2 and sector 4.
B700          482          ;
B700 8E E9 B7  483          stx SNUM16
B703 8E F7 B7  484          stx SLOTFND
B706          485          ;
B706 A9 01     486          lda #1
B708 8D F8 B7  487          sta DRVFNDD
B70B 8D EA B7  488          sta DNUM
B70E          489          ;
B70E AD E0 B7  490          lda NumpGS
B711 8D E1 B7  491          sta NSECSRW
B714          492          ;
B714 A9 02     493          lda #2
B716 8D EC B7  494          sta TNUM
B719          495          ;
B719 A9 04     496          lda #4
B71B 8D ED B7  497          sta SNUM
B71E          498          ;
B71E AC E7 B7  499          ldy FRSTBOOT+1
B721 88        500          dey
B722 8C F1 B7  501          sty USRBUF+1
B725          502          ;
B725 A9 01     503          lda #RWTSREAD
B727 8D F4 B7  504          sta CMDCODE
B72A          505          ;
B72A 8A        506          txa
B72B          507          ;
B72B 4A        508          lsr
B72C 4A        509          lsr
B72D 4A        510          lsr
B72E 4A        511          lsr
B72F          512          ;
B72F AA        513          tax
B730          514          ;
B730 A9 00     515          lda #ZERO
B732 9D F8 04  516          sta DRV1TRK,X
B735 9D 78 04  517          sta DRV0TRK,X
B738          518          ;
B738 20 93 B7  519          jsr RWPAGES
B73B          520          ;
B73B A2 FF     521          ldx #NEGONE
B73D 9A        522          txs
B73E 8E EB B7  523          stx VOLEXPT
B741          524          ;
B741 4C C8 BF  525          jmp REBOOT
B744          526          ;
B744          527          ;
B744          528          RESTART:
B744 20 89 FE  529          jsr SETKBD
B747 4C 84 9D  530          jmp DOSSTRT
B74A          531          ;
B74A          532          ;
B74A          533          ; Write DOS on first two tracks of disk.
B74A          534          ;
B74A          535          PUTDOS:
B74A AD E7 B7  536          lda FRSTBOOT+1
B74D 38        537          sec
B74E ED F1 B7  538          sbc USRBUF+1

```



```

B751 8D E1 B7      539      sta NSECSRW
B754              540      ;
B754 AD E7 B7      541      lda FRSTBOOT+1
B757 8D F1 B7      542      sta USRBUF+1
B75A              543      ;
B75A CE F1 B7      544      dec USRBUF+1
B75D              545      ;
B75D A9 02         546      lda #2
B75F 8D EC B7      547      sta TNUM
B762              548      ;
B762 A9 04         549      lda #4
B764 8D ED B7      550      sta SNUM
B767              551      ;
B767 A9 02         552      lda #RWTSWRIT
B769 8D F4 B7      553      sta CMDCODE
B76C              554      ;
B76C 20 93 B7      555      jsr RWPAGES
B76F              556      ;
B76F AD E7 B7      557      lda FRSTBOOT+1
B772 8D FE B6      558      sta PTCHSTOP+1      ; location of BOOTADR
B775              559      ;
B775 18            560      clc
B776 69 09         561      adc #9
B778 8D F1 B7      562      sta USRBUF+1
B77B              563      ;
B77B A9 0A         564      lda #$A
B77D 8D E1 B7      565      sta NSECSRW
B780              566      ;
B780 38            567      sec
B781 E9 01         568      sbc #1
B783 8D FF B6      569      sta PTCHSTOP+2      ; location of BOOTPGS
B786 8D ED B7      570      sta SNUM
B789              571      ;
B789 20 93 B7      572      jsr RWPAGES
B78C              573      ;
B78C 60            574      rts
B78D              575      ;
B78D              576      dfs 6,ZERO
B793              577      ;
B793              578      ;
B793              579      ; Used by Stage 2 boot and FORMAT command.
B793              580      ;
B793              581      RWPAGES:
B793 AD E5 B7      582      lda RWTSPPTR+1
B796 AC E4 B7      583      ldy RWTSPPTR
B799              584      ;
B799 20 B5 B7      585      jsr CALLRWTS
B79C              586      ;
B79C AC ED B7      587      ldy SNUM
B79F              588      ;
B79F 88            589      dey
B7A0 10 07         590      bpl >1
B7A2              591      ;
B7A2 A0 0F         592      ldy #$F
B7A4              593      ;
B7A4 EA            594      nop
B7A5 EA            595      nop
B7A6              596      ;
B7A6 CE EC B7      597      dec TNUM
B7A9              598      ;
B7A9 8C ED B7      599      ^1      sty SNUM

```

```

B7AC          600 ;
B7AC CE F1 B7 601      dec USRBUF+1
B7AF CE E1 B7 602      dec NSECSRW
B7B2 D0 DF    603      bne RWPAGES
B7B4          604 ;
B7B4 60       605      rts
B7B5          606 ;
B7B5          607 ;
B7B5          608 ; Page 3 entry.
B7B5          609 ;
B7B5          610 CALLRWTS:
B7B5 08       611      php
B7B6          612 ;
B7B6 78       613      sei
B7B7          614 ;
B7B7 20 00 BD 615      jsr RWTSENT
B7BA B0 03    616      bcs >1
B7BC          617 ;
B7BC 28       618      plp
B7BD          619 ;
B7BD 18       620      clc
B7BE 60       621      rts
B7BF          622 ;
B7BF 28       623 ^1    plp
B7C0          624 ;
B7C0 38       625      sec
B7C1 60       626      rts
B7C2          627 ;
B7C2          628 ;
B7C2          629 ; Used by FORMAT command to write sectors.
B7C2          630 ;
B7C2          631 RWTSPRMS:
B7C2 AD BC B5 632      lda SUBCODE
B7C5 8D F1 B7 633      sta USRBUF+1
B7C8          634 ;
B7C8 A9 00    635      lda #ZERO
B7CA 8D F0 B7 636      sta USRBUF
B7CD          637 ;
B7CD AD F9 B5 638      lda VOLNUMBR
B7D0 49 FF    639      eor #NEGONE
B7D2 8D EB B7 640      sta VOLEXPT
B7D5          641 ;
B7D5 60       642      rts
B7D6          643 ;
B7D6          644 ;
B7D6          645 ZEROBUFR:
B7D6 A9 00    646      lda #ZERO
B7D8 A8       647      tay
B7D9          648 ;
B7D9 91 42    649 ^1    sta (BUFADRZ),Y
B7DB          650 ;
B7DB C8       651      iny
B7DC D0 FB    652      bne <1
B7DE          653 ;
B7DE 60       654      rts
B7DF          655 ;
B7DF          656 ;
B7DF          657      dfs 1,ZERO      ; not used
B7E0          658 ;
B7E0 1B       659 NUMPGS    hex 1B      ; stage 2 boot number pages
B7E1 00 0A 1B 660 NSECSRW    hex 000A1B ; stage 2 boot sector count

```

```

B7E4      661      ;
B7E4 E8 B7      662      RWTSPPTR adr TBLTYPE      ; address of IOB
B7E6 00 B6      663      FRSTBOOT adr BOOTCODE      ; stage 2 boot address begin
B7E8      664      ;
B7E8      665      ;
B7E8      666      ; RWTS Input/Output Context Block.
B7E8      667      ;
B7E8      668      TBLTYPE      dfs 1,1      ; IOB structure block
B7E9 60      669      SNUM16      hex 60      ; slot * 16
B7EA      670      DNUM      dfs 1,1      ; drive number
B7EB      671      VOLEXPT      dfs 1,ZERO      ; volume number
B7EC      672      TNUM      dfs 1,ZERO      ; track number
B7ED      673      SNUM      dfs 1,9      ; sector number
B7EE FB B7      674      DCTADR      adr DEVTYPE      ; address of DCT table
B7F0 00 9D      675      USRBUF      adr DOSTART      ; user buffer address
B7F2 00 01      676      BYTCNT      hex 0001      ; bytes to read/write
B7F4      677      CMDCODE      dfs 1,1      ; command
B7F5      678      ERRRCODE      dfs 1,ZERO      ; return error code
B7F6      679      VOLFND      dfs 1,ZERO      ; return volume found
B7F7 60      680      SLOTFND      hex 60      ; return slot found
B7F8      681      DRVFND      dfs 1,1      ; return drive found
B7F9      682      ;
B7F9      683      dfs 2,ZERO      ; not used
B7FB      684      ;
B7FB      685      DEVTYPE      dfs 1,ZERO      ; disk device structure block
B7FC      686      PHPERTRK      dfs 1,1      ; phases per track
B7FD EF D8      687      MOTONTIM      hex EFD8      ; motor on time
B7FF      688      ;
B7FF      689      dfs 1,ZERO      ; not used
B800      690      ;
B800      691      ;

```

```
BSAVE SEG02,D1,A$1000,B,L$0D3F
```

```

B800      692      usr SEG02,D1
B800      693      ;
B800      694      ;
B800      695      icl "RWTS1.L,D2"

```

```
LLOAD RWTS1.L,D2,A$4000
```

```

B800          1          ttl "DOS 3.3 Source Code, RWTS1.L"
B800          2          ;
B800          3          ;
B800          4          ; RWTS1.L
B800          5          ;
B800          6          ;
B800          7          obj PAGE10
B800          8          usr
B800          9          ;
B800         10          ;
B800         11          ; The Prenibblize routine converts 256 bytes pointed at by
B800         12          ; BUFADR2Z to 342 6-bit nibbles of the form 00XXXXXX.
B800         13          ;
B800         14          PRENIB16:
B800 A2 00     15          ldx #ZERO
B802 A0 02     16          ldy #2
B804          17          ;
B804          18          ;
B804          19          ; Get next user byte.
B804          20          ;
B804 88        21          ^1      dey
B805          22          ;
B805 B1 3E     23          lda (BUFADR2Z),Y
B807          24          ;
B807          25          ;
B807          26          ; Shift high order two bytes into NBUF2.
B807          27          ;
B807 4A        28          lsr
B808 3E 00 BC  29          rol NBUF2,X
B80B          30          ;
B80B 4A        31          lsr
B80C 3E 00 BC  32          rol NBUF2,X
B80F          33          ;
B80F          34          ;
B80F          35          ; Put low order six bits (shifted left) into NBUF1.
B80F          36          ;
B80F 99 00 BB  37          sta NBUF1,Y
B812          38          ;
B812 E8        39          inx
B813          40          ;
B813 E0 56     41          cpx #NBUF2SIZ
B815 90 ED     42          bcc <1
B817          43          ;
B817          44          ;
B817          45          ; Done yet?
B817          46          ;
B817 A2 00     47          ldx #ZERO
B819          48          ;
B819 98        49          tya
B81A D0 E8     50          bne <1
B81C          51          ;
B81C          52          ;
B81C          53          ; Strip high order two bits of NBUF2.
B81C          54          ;
B81C A2 55     55          ldx #NBUF2SIZ-1
B81E          56          ;
B81E BD 00 BC  57          ^2      lda NBUF2,X
B821 29 3F     58          and #NBUFMASK
B823 9D 00 BC  59          sta NBUF2,X
B826          60          ;

```

```

B826 CA          61          dex
B827 10 F5       62          bpl <2
B829            63          ;
B829 60          64          rts
B82A            65          ;
B82A            66          ;
B82A            67          ; Write subroutine writes prenibbilized data in NBUF1 and
B82A            68          ; NBUF2 to disk. On exit C-flag = 0 for okay and C-flag =
B82A            69          ; 1 for error.
B82A            70          ;
B82A            71          ; Note: these routines are all time critical. Be careful
B82A            72          ; of page boundries and do not remove NOP instructions.
B82A            73          ;
B82A            74          WRITE16:
B82A 38          75          sec                      ; anticipate write protect
B82B            76          ;
B82B 86 27       77          stx TEMP2Z
B82D 8E 78 06    78          stx SLO TABS
B830            79          ;
B830 BD 8D C0    80          lda LATCH,X
B833 BD 8E C0    81          lda DATAIN,X              ; sense write protect
B836            82          ;
B836 30 7C       83          bmi >5
B838            84          ;
B838 AD 00 BC    85          lda NBUF2
B83B 85 26       86          sta TEMPZ
B83D            87          ;
B83D A9 FF       88          lda #SYNCKMARK             ; sync byte
B83F 9D 8F C0    89          sta DATAOUT,X            ; write 1st nibble
B842 1D 8C C0    90          ora STROBE,X
B845            91          ;
B845 48          92          pha
B846 68          93          pla
B847            94          ;
B847 EA          95          nop
B848            96          ;
B848            97          ;
B848            98          ; Write the four sync bytes.
B848            99          ;
B848 A0 04      100          ldy #4
B84A            101          ;
B84A 48          102          ^1 pha
B84B 68          103          pla
B84C            104          ;
B84C 20 B9 B8    105          jsr WNIBL7
B84F            106          ;
B84F 88          107          dey
B850 D0 F8       108          bne <1
B852            109          ;
B852 A9 D5       110          lda #DATMARK1             ; first data mark
B854 20 B8 B8    111          jsr WNIBL9
B857            112          ;
B857 A9 AA       113          lda #DATMARK2             ; second data mark
B859 20 B8 B8    114          jsr WNIBL9
B85C            115          ;
B85C A9 AD       116          lda #DATMARK3             ; third data mark
B85E 20 B8 B8    117          jsr WNIBL9
B861            118          ;
B861 98          119          tya                      ; clear checksum
B862            120          ;
B862 A0 56      121          ldy #NBUF2SIZ              ; NBUF2 index

```

```

B864 D0 03      122      bne >3
B866            123      ;
B866 B9 00 BC   124      ^2      lda NBUF2,Y          ; get prior 6-bit nibble
B869            125      ;
B869 59 FF BB   126      ^3      eor NBUF2-1,Y        ; XOR with current nibble
B86C AA         127      tax
B86D            128      ;
B86D BD 29 BA   129      lda WRNIBL,X
B870            130      ;
B870 A6 27      131      ldx TEMP2Z
B872            132      ;
B872 9D 8D C0   133      sta LATCH,X          ; write nibble
B875 BD 8C C0   134      lda STROBE,X
B878            135      ;
B878 88         136      dey          ; next nibble
B879 D0 EB      137      bne <2
B87B            138      ;
B87B            139      ;
B87B            140      ; Now handle NBUF1.  Get prior nibble.
B87B            141      ;
B87B A5 26      142      lda TEMPZ
B87D            143      ;
B87D EA         144      nop
B87E            145      ;
B87E            146      ;
B87E            147      ; Loop to write out data in NBUF1.
B87E            148      ;
B87E 59 00 BB   149      ^4      eor NBUF1,Y
B881 AA         150      tax
B882            151      ;
B882 BD 29 BA   152      lda WRNIBL,X
B885            153      ;
B885 AE 78 06   154      ldx SLO TABS
B888            155      ;
B888 9D 8D C0   156      sta LATCH,X          ; write nibble
B88B BD 8C C0   157      lda STROBE,X
B88E            158      ;
B88E B9 00 BB   159      lda NBUF1,Y
B891            160      ;
B891 C8         161      iny
B892 D0 EA      162      bne <4
B894            163      ;
B894 AA         164      tax
B895            165      ;
B895 BD 29 BA   166      lda WRNIBL,X
B898            167      ;
B898            168      ;
B898            169      ; Write checksum.
B898            170      ;
B898 A6 27      171      ldx TEMP2Z
B89A            172      ;
B89A 20 BB B8   173      jsr WNIBL
B89D            174      ;
B89D            175      ;
B89D            176      ; Write epilog to data.
B89D            177      ;
B89D A9 DE      178      lda #SLPMARK1          ; bit slip mark 1
B89F 20 B8 B8   179      jsr WNIBL9
B8A2            180      ;
B8A2 A9 AA      181      lda #SLPMARK2          ; bit slip mark 2
B8A4 20 B8 B8   182      jsr WNIBL9

```

```
B8A7      183 ;
B8A7 A9 EB 184      lda #SLPMARK3      ; bit slip mark 3
B8A9 20 B8 B8 185      jsr WNIBL9
B8AC      186 ;
B8AC      187 ;
B8AC      188 ; All done, close up the shop!
B8AC      189 ;
B8AC A9 FF 190      lda #SYNCMARK
B8AE 20 B8 B8 191      jsr WNIBL9
B8B1      192 ;
B8B1      193 ;
B8B1      194 ; Turn off the write mode.
B8B1      195 ;
B8B1 BD 8E C0 196      lda DATAIN,X
B8B4      197 ;
B8B4      198 ;
B8B4      199 ; And back to the read mode.
B8B4      200 ;
B8B4 BD 8C C0 201 ^5      lda STROBE,X
B8B7      202 ;
B8B7 60      203      rts
B8B8      204 ;
B8B8      205 ;
B8B8      206 ; WNIBL9 is 9 clock cycles, then write.
B8B8      207 ;
B8B8      208 WNIBL9:
B8B8 18      209      clc
B8B9      210 ;
B8B9      211 ; WNIBL7 is 7 clock cycles, then write.
B8B9      212 ;
B8B9      213 WNIBL7:
B8B9 48      214      pha
B8BA 68      215      pla
B8BB      216 ;
B8BB      217 ; Write nibble to disk.
B8BB      218 ;
B8BB      219 WNIBL:
B8BB 9D 8D C0 220      sta LATCH,X
B8BE 1D 8C C0 221      ora STROBE,X
B8C1      222 ;
B8C1 60      223      rts
B8C2      224 ;
B8C2      225 ;
B8C2      226 ; The post nibblize routine converts 342 nibbles of the
B8C2      227 ; form 00xxxxxx to eight bit bytes. The nibbles are
B8C2      228 ; stored in NBUF1 and NBUF2, the 8-bit bytes will be
B8C2      229 ; stored at (BUFADR2Z). Before calling this routine
B8C2      230 ; initialize TEMPZ with zero.
B8C2      231 ;
B8C2      232 POSTNB16:
B8C2 A0 00      233      ldy #ZERO
B8C4      234 ;
B8C4 A2 56      235 ^1      ldx #NBUF2SIZ
B8C6      236 ;
B8C6 CA      237 ^2      dex
B8C7 30 FB      238      bmi <1
B8C9      239 ;
B8C9      240 ;
B8C9      241 ; Get byte and shift in low order two bits from NBUF2.
B8C9      242 ;
B8C9 B9 00 BB      243      lda NBUF1,Y
```

```

B8CC      244 ;
B8CC 5E 00 BC 245      lsr NBUF2,X
B8CF      246 ;
B8CF 2A      247      rol
B8D0 5E 00 BC 248      lsr NBUF2,X
B8D3      249 ;
B8D3 2A      250      rol
B8D4      251 ;
B8D4      252 ;
B8D4      253 ; Store in user buffer.
B8D4      254 ;
B8D4 91 3E    255      sta (BUFADR2Z),Y
B8D6      256 ;
B8D6 C8      257      iny
B8D7      258 ;
B8D7 C4 26    259      cpy TEMPZ
B8D9 D0 EB    260      bne <2
B8DB      261 ;
B8DB 60      262      rts
B8DC      263 ;
B8DC      264 ;
B8DC      265 ; READ routine reads a sector from the disk and stores the
B8DC      266 ; data in NBUF1 and NBUF2. Fail after 32 attempts.
B8DC      267 ;
B8DC      268 READ16:
B8DC A0 20    269      ldy #32
B8DE      270 ;
B8DE      271 ;
B8DE      272 ; Get sync bytes.
B8DE      273 ;
B8DE 88      274 ^1      dey
B8DF F0 61    275      beq RDERR
B8E1      276 ;
B8E1      277 ;
B8E1      278 ; Wait until a byte is recieved from the disk drive.
B8E1      279 ;
B8E1 BD 8C C0 280 ^2      lda STROBE,X
B8E4 10 FB    281      bpl <2
B8E6      282 ;
B8E6      283 ;
B8E6      284 ; Byte received, so check for data mark 1.
B8E6      285 ;
B8E6 49 D5    286 ^3      eor #DATMARK1
B8E8 D0 F4    287      bne <1
B8EA      288 ;
B8EA EA      289      nop
B8EB      290 ;
B8EB      291 ;
B8EB      292 ; Get next byte and check for data mark 2.
B8EB      293 ;
B8EB BD 8C C0 294 ^4      lda STROBE,X
B8EE 10 FB    295      bpl <4
B8F0      296 ;
B8F0 C9 AA    297      cmp #DATMARK2
B8F2 D0 F2    298      bne <3
B8F4      299 ;
B8F4 A0 56    300      ldy #NBUF2SIZ
B8F6      301 ;
B8F6      302 ;
B8F6      303 ; Check for data mark 3.
B8F6      304 ;

```



```

B8F6 BD 8C C0    305 ^5      lda STROBE,X
B8F9 10 FB      306      bpl <5
B8FB           307      ;
B8FB C9 AD      308      cmp #DATMARK3
B8FD D0 E7      309      bne <3
B8FF           310      ;
B8FF           311      ;
B8FF           312      ; Initialize checksum to zero and read the data from the
B8FF           313      ; sector.
B8FF           314      ;
B8FF A9 00      315      lda #ZERO
B901           316      ;
B901           317      ;
B901           318      ; Read nibbles into NBUF2.
B901           319      ;
B901 88         320 ^1      dey
B902 84 26      321      sty TEMPZ
B904           322      ;
B904 BC 8C C0   323 ^2      ldY STROBE,X
B907 10 FB      324      bpl <2
B909           325      ;
B909 59 00 BA   326      eor RDNIBL-$96,Y
B90C           327      ;
B90C A4 26      328      ldY TEMPZ
B90E           329      ;
B90E 99 00 BC   330      sta NBUF2,Y
B911           331      ;
B911 D0 EE      332      bne <1
B913           333      ;
B913           334      ;
B913           335      ; Read nibbles into NBUF1.
B913           336      ;
B913 84 26      337 ^3      sty TEMPZ
B915           338      ;
B915 BC 8C C0   339 ^4      ldY STROBE,X
B918 10 FB      340      bpl <4
B91A           341      ;
B91A 59 00 BA   342      eor RDNIBL-$96,Y
B91D           343      ;
B91D A4 26      344      ldY TEMPZ
B91F           345      ;
B91F 99 00 BB   346      sta NBUF1,Y
B922           347      ;
B922 C8         348      iny
B923 D0 EE      349      bne <3
B925           350      ;
B925           351      ;
B925           352      ; Check the checksum byte.
B925           353      ;
B925 BC 8C C0   354 ^5      ldY STROBE,X
B928 10 FB      355      bpl <5
B92A           356      ;
B92A D9 00 BA   357      cmp RDNIBL-$96,Y
B92D D0 13      358      bne RDERR
B92F           359      ;
B92F           360      ;
B92F           361      ; Check for bit slip mark 1.
B92F           362      ;
B92F BD 8C C0   363 ^6      lda STROBE,X
B932 10 FB      364      bpl <6
B934           365      ;

```

```

B934 C9 DE      366      cmp #SLPMARK1
B936 D0 0A      367      bne RDERR
B938           368      ;
B938 EA        369      nop
B939           370      ;
B939           371      ;
B939           372      ; Check for bit slip mark 2.  Bit slip mark 3 is not
B939           373      ; checked.
B939           374      ;
B939 BD 8C C0    375      ^7      lda STROBE,X
B93C 10 FB      376      bpl <7
B93E           377      ;
B93E C9 AA      378      cmp #SLPMARK2
B940 F0 5C      379      beq RDADRX
B942           380      ;
B942 38         381      RDERR    sec
B943 60         382      rts
B944           383      ;
B944           384      ;
B944           385      ; Read address field.
B944           386      ;
B944           387      ; Reads the starting address marks ($D5, $AA, $96),
B944           388      ; address information (volume/track/sector/checksum), and
B944           389      ; bit slip marks ($DE, $AA).  Address information is
B944           390      ; odd/even encoded.  Read up to $400 disk bytes before
B944           391      ; giving up.
B944           392      ;
B944           393      RDADR16:
B944 A0 FC      394      ldy #$FC
B946 84 26      395      sty TEMPZ
B948           396      ;
B948 C8         397      ^0      iny
B949 D0 04      398      bne >1
B94B           399      ;
B94B E6 26      400      inc TEMPZ
B94D F0 F3      401      beq RDERR
B94F           402      ;
B94F           403      ;
B94F           404      ; Read first address mark ($D5).
B94F           405      ;
B94F BD 8C C0    406      ^1      lda STROBE,X
B952 10 FB      407      bpl <1
B954           408      ;
B954 C9 D5      409      ^2      cmp #ADRMARK1
B956 D0 F0      410      bne <0
B958           411      ;
B958 EA        412      nop
B959           413      ;
B959           414      ;
B959           415      ; Read second address mark ($AA).
B959           416      ;
B959 BD 8C C0    417      ^3      lda STROBE,X
B95C 10 FB      418      bpl <3
B95E           419      ;
B95E C9 AA      420      cmp #ADRMARK2
B960 D0 F2      421      bne <2
B962           422      ;
B962 A0 03      423      ldy #3
B964           424      ;
B964           425      ;
B964           426      ; Read third address mark ($96)

```

```
B964          427 ;
B964 BD 8C C0 428 ^4      lda STROBE,X
B967 10 FB    429          bpl <4
B969          430 ;
B969 C9 96    431          cmp #ADRMARK3
B96B D0 E7    432          bne <2
B96D          433 ;
B96D          434 ;
B96D          435 ; Initialize checksum and read the address data field
B96D          436 ; (four bytes).
B96D          437 ;
B96D A9 00    438          lda #ZERO
B96F          439 ;
B96F 85 27    440 ^5      sta TEMP2Z
B971          441 ;
B971          442 ;
B971          443 ; Read 'odd' bit nibble.
B971          444 ;
B971 BD 8C C0 445 ^6      lda STROBE,X
B974 10 FB    446          bpl <6
B976          447 ;
B976 2A       448          rol
B977 85 26    449          sta TEMPZ
B979          450 ;
B979          451 ;
B979          452 ; Read 'even' bit nibble.
B979          453 ;
B979 BD 8C C0 454 ^7      lda STROBE,X
B97C 10 FB    455          bpl <7
B97E          456 ;
B97E          457 ;
B97E          458 ; Merge the two nibbles.
B97E          459 ;
B97E 25 26    460          and TEMPZ
B980          461 ;
B980          462 ;
B980          463 ; Store the data byte, update checksum, and repeat until
B980          464 ; the entire address field is read.
B980          465 ;
B980 99 2C 00 466          sta DATAFNDZ,Y
B983 45 27    467          eor TEMP2Z
B985          468 ;
B985 88       469          dey
B986 10 E7    470          bpl <5
B988          471 ;
B988          472 ;
B988          473 ; Checksum in A-reg must be zero.
B988          474 ;
B988 A8       475          tay
B989 D0 B7    476          bne RDERR
B98B          477 ;
B98B          478 ;
B98B          479 ; Read first bit slip mark ($DE).
B98B          480 ;
B98B BD 8C C0 481 ^8      lda STROBE,X
B98E 10 FB    482          bpl <8
B990          483 ;
B990 C9 DE    484          cmp #SLPMARK1
B992 D0 AE    485          bne RDERR
B994          486 ;
B994 EA       487          nop
```

```

B995          488 ;
B995          489 ; Read second bit slip mark ($AA).
B995          490 ;
B995 BD 8C C0 491 ^9      lda STROBE,X
B998 10 FB    492          bpl <9
B99A          493 ;
B99A C9 AA    494          cmp #SLPMARK2
B99C D0 A4    495          bne RDERR
B99E          496 ;
B99E 18       497 RDADR    clc
B99F 60       498          rts
B9A0          499 ;
B9A0          500 ;
B9A0          501 ; The SEEKABS routine moves the disk head to the desired
B9A0          502 ; track.
B9A0          503 ;
B9A0          504 SEEKABS:
B9A0 86 2B    505          stx SLOT16Z
B9A2          506 ;
B9A2 85 2A    507          sta SEEKTRKZ
B9A4 CD 78 04 508          cmp CURTRK
B9A7 F0 53    509          beq >6
B9A9          510 ;
B9A9 A9 00    511          lda #ZERO
B9AB 85 26    512          sta TEMPZ
B9AD          513 ;
B9AD AD 78 04 514 ^1      lda CURTRK
B9B0 85 27    515          sta TEMP2Z
B9B2          516 ;
B9B2 38       517          sec
B9B3          518 ;
B9B3 E5 2A    519          sbc SEEKTRKZ
B9B5 F0 33    520          beq ISTHERE
B9B7          521 ;
B9B7 B0 07    522          bcs >2
B9B9          523 ;
B9B9 49 FF    524          eor #NEGONE
B9BB          525 ;
B9BB EE 78 04 526          inc CURTRK
B9BE 90 05    527          bcc >3
B9C0          528 ;
B9C0 69 FE    529 ^2      adc #$FE          ; -2
B9C2          530 ;
B9C2 CE 78 04 531          dec CURTRK
B9C5          532 ;
B9C5 C5 26    533 ^3      cmp TEMPZ
B9C7 90 02    534          bcc >4
B9C9          535 ;
B9C9 A5 26    536          lda TEMPZ
B9CB          537 ;
B9CB C9 0C    538 ^4      cmp #OFFTBL-ONTBL
B9CD B0 01    539          bcs >5
B9CF          540 ;
B9CF A8       541          tay
B9D0          542 ;
B9D0 38       543 ^5      sec
B9D1 20 EE B9 544          jsr CHKPOS
B9D4          545 ;
B9D4 B9 11 BA 546          lda ONTBL,Y
B9D7 20 00 BA 547          jsr MSWAIT
B9DA          548 ;

```

```

B9DA A5 27      549      lda TEMP2Z
B9DC           550      ;
B9DC 18         551      clc
B9DD 20 F1 B9   552      jsr CHKPOS2
B9E0           553      ;
B9E0 B9 1D BA   554      lda OFFTBL,Y
B9E3 20 00 BA   555      jsr MSWAIT
B9E6           556      ;
B9E6 E6 26      557      inc TEMPZ
B9E8 D0 C3      558      bne <1
B9EA           559      ;
B9EA 20 00 BA   560      ISTHERE jsr MSWAIT
B9ED           561      ;
B9ED 18         562      clc
B9EE           563      ;
B9EE AD 78 04   564      CHKPOS  lda CURTRK
B9F1           565      ;
B9F1 29 03      566      CHKPOS2 and #3
B9F3 2A         567      rol
B9F4 05 2B      568      ora SLOT16Z
B9F6 AA         569      tax
B9F7           570      ;
B9F7 BD 80 C0   571      lda PHASEOFF,X
B9FA           572      ;
B9FA A6 2B      573      ldx SLOT16Z
B9FC           574      ;
B9FC 60         575      ^6      rts
B9FD           576      ;
B9FD           577      ;
B9FD           578      dfs 3,ZERO
BA00           579      ;
BA00           580      ;
BA00           581      ; Head move delay subroutine delays A-reg * 100 usec. This
BA00           582      ; routine begins on a page boundary.
BA00           583      ;
BA00           584      MSWAIT:
BA00 A2 11      585      ldx #17
BA02           586      ;
BA02 CA         587      ^1      dex                ; delay 86 usec
BA03 D0 FD      588      bne <1
BA05           589      ;
BA05 E6 46      590      inc MONTIME
BA07 D0 02      591      bne >3
BA09           592      ;
BA09 E6 47      593      inc MONTIME+1
BA0B           594      ;
BA0B 38         595      ^3      sec
BA0C E9 01      596      sbc #1
BA0E D0 F0      597      bne MSWAIT
BA10           598      ;
BA10 60         599      rts
BA11           600      ;
BA11           601      ;
BA11           602      ; PHASEON/PHASEOFF tables. These are 100 usec intervals.
BA11           603      ;
BA11 01 30 28   604      ONTBL   hex 01302824201E1D1C1C1C1C1C
BA14 24 20 1E
BA17 1D 1C 1C
BA1A 1C 1C 1C
BA1D 70 2C 26   605      OFFTBL   hex 702C26221F1E1D1C1C1C1C1C
BA20 22 1F 1E

```

```

BA23 1D 1C 1C
BA26 1C 1C 1C
BA29          606 ;
BA29          607 ;
BA29          608 ; Write translate table.
BA29          609 ;
BA29 96 97 9A 610 WRNIBL    hex 96979A9B9D9E9FA6A7ABACADAEAFB2B3
BA2C 9B 9D 9E
BA2F 9F A6 A7
BA32 AB AC AD
BA35 AE AF B2
BA38 B3
BA39 B4 B5 B6 611          hex B4B5B6B7B9BABBBBCBDBEBFCBCDCECFD3
BA3C B7 B9 BA
BA3F BB BC BD
BA42 BE BF CB
BA45 CD CE CF
BA48 D3
BA49 D6 D7 D9 612          hex D6D7D9DADBDCDDDEDFE5E6E7E9EAEBEC
BA4C DA DB DC
BA4F DD DE DF
BA52 E5 E6 E7
BA55 E9 EA EB
BA58 EC
BA59 ED EE EF 613          hex EDEEEFF2F3F4F5F6F7F9FAFBFCFDFEFF
BA5C F2 F3 F4
BA5F F5 F6 F7
BA62 F9 FA FB
BA65 FC FD FE
BA68 FF
BA69          614 ;
BA69          615 ;
BA69          616 ; If not the APPEND command, zero APPFLG.
BA69          617 ;
BA69 AE 5F AA 618 HBA69    ldx CMDINDX
BA6C E0 1C    619          cpx #$1C
BA6E F0 05    620          beq >1
BA70          621 ;
BA70 A2 00    622          ldx #ZERO
BA72 8E 5D B6 623          stx APPFLG
BA75          624 ;
BA75 60       625 ^1      rts
BA76          626 ;
BA76          627 ;
BA76          628 ; Init XMODE (all bits on) and set video to VID80OFF and
BA76          629 ; character set to ALTCHOFF.
BA76          630 ;
BA76 A9 FF    631 HBA76    lda #NEGONE
BA78 8D FB 04 632          sta XMODE
BA7B 8D 0C C0 633          sta VID80OFF
BA7E 8D 0E C0 634          sta ALTCHOFF
BA81          635 ;
BA81 4C 2F FB 636          jmp INIT
BA84          637 ;
BA84          638 ;
BA84          639 ; From APTCH2, exit FILEMGR.
BA84          640 ;
BA84 AD BD B5 641 HBA84    lda RECNUM
BA87 8D E6 B5 642          sta FILEBYTE
BA8A 8D EA B5 643          sta RECNUMBR
BA8D          644 ;

```

```

BA8D BA          645          tsx
BA8E 8E 9B B3    646          stx STKSAVE
BA91          647          ;
BA91 4C 7F B3    648          jmp NOERROR
BA94          649          ;
BA94          650          ;
BA94          651          ; Align the read translate table.
BA94          652          ;
BA94          653          dfs $96-* -PAGESIZE**/PAGESIZE,ZERO
BA96          654          ;
BA96 00 01 98    655  RDNIBL   hex 0001989902039C040506A0A1A2A3A4A5
BA99 99 02 03
BA9C 9C 04 05
BA9F 06 A0 A1
BAA2 A2 A3 A4
BAA5 A5
BAA6 07 08 A8    656          hex 0708A8A9AA090A0B0C0DB0B10E0F1011
BAA9 A9 AA 09
BAAC 0A 0B 0C
BAAF 0D B0 B1
BAB2 0E 0F 10
BAB5 11
BAB6 12 13 B8    657          hex 1213B81415161718191AC0C1C2C3C4C5
BAB9 14 15 16
BABC 17 18 19
BABF 1A C0 C1
BAC2 C2 C3 C4
BAC5 C5
BAC6 C6 C7 C8    658          hex C6C7C8C9CA1BCC1C1D1ED0D1D21FD4D5
BAC9 C9 CA 1B
BACC CC 1C 1D
BACF 1E D0 D1
BAD2 D2 1F D4
BAD5 D5
BAD6 20 21 D8    659          hex 2021D822232425262728E0E1E2E3E429
BAD9 22 23 24
BADC 25 26 27
BADF 28 E0 E1
BAE2 E2 E3 E4
BAE5 29
BAE6 2A 2B E8    660          hex 2A2BE82C2D2E2F303132F0F133343536
BAE9 2C 2D 2E
BAEC 2F 30 31
BAEF 32 F0 F1
BAF2 33 34 35
BAF5 36
BAF6 37 38 F8    661          hex 3738F8393A3B3C3D3E3F
BAF9 39 3A 3B
BAFC 3C 3D 3E
BAFF 3F
BB00          662          ;
BB00          663          ;
BB00          664          ; Nibble buffers.
BB00          665          ;
BB00          666  NBUF1      dfs PAGESIZE,ZERO
BC00          667  NBUF2      dfs NBUF2SIZ,ZERO
BC56          668          ;
BC56          669          ;
BC56          670          ; Write an address field routine. Assume write protect
BC56          671          ; error.
BC56          672          ;

```

```

BC56      673  WRADR16:
BC56 38      674      sec
BC57      675  ;
BC57 BD 8D C0 676      lda LATCH,X
BC5A BD 8E C0 677      lda DATAIN,X
BC5D      678  ;
BC5D 30 5E    679      bmi >2
BC5F      680  ;
BC5F      681  ;
BC5F      682  ; Output sync bytes.
BC5F      683  ;
BC5F A9 FF    684      lda #SYNCMARK
BC61 9D 8F C0 685      sta DATAOUT,X
BC64 DD 8C C0 686      cmp STROBE,X
BC67      687  ;
BC67 48      688      pha
BC68 68      689      pla
BC69      690  ;
BC69 20 C3 BC 691  ^1    jsr WAIT12
BC6C 20 C3 BC 692      jsr WAIT12
BC6F      693  ;
BC6F 9D 8D C0 694      sta LATCH,X
BC72 DD 8C C0 695      cmp STROBE,X
BC75      696  ;
BC75 EA      697      nop
BC76      698  ;
BC76 88      699      dey
BC77 D0 F0    700      bne <1
BC79      701  ;
BC79      702  ;
BC79      703  ; Output the address marks $D5, $AA, and $96.
BC79      704  ;
BC79 A9 D5    705      lda #ADRMARK1
BC7B 20 D5 BC 706      jsr WBYTE9
BC7E      707  ;
BC7E A9 AA    708      lda #ADRMARK2
BC80 20 D5 BC 709      jsr WBYTE9
BC83      710  ;
BC83 A9 96    711      lda #ADRMARK3
BC85 20 D5 BC 712      jsr WBYTE9
BC88      713  ;
BC88      714  ;
BC88      715  ; Output volume, track, sector, and checksum
BC88      716  ;
BC88 A5 41    717      lda VOLUMEZ
BC8A 20 C4 BC 718      jsr WBYTE
BC8D      719  ;
BC8D A5 44    720      lda TRACKZ
BC8F 20 C4 BC 721      jsr WBYTE
BC92      722  ;
BC92 A5 3F    723      lda SECTORZ
BC94 20 C4 BC 724      jsr WBYTE
BC97      725  ;
BC97 A5 41    726      lda VOLUMEZ
BC99 45 44    727      eor TRACKZ
BC9B 45 3F    728      eor SECTORZ
BC9D 48      729      pha
BC9E      730  ;
BC9E 4A      731      lsr
BC9F      732  ;
BC9F 05 3E    733      ora ODDBITSZ

```



```

BCA1 9D 8D C0    734          sta LATCH,X
BCA4 BD 8C C0    735          lda STROBE,X
BCA7            736          ;
BCA7 68          737          pla
BCA8            738          ;
BCA8 09 AA      739          ora #ODDBITS
BCAA 20 D4 BC    740          jsr WBYTE11
BCAD            741          ;
BCAD            742          ;
BCAD            743          ; Output slip bit marks $DE, $AA, and $EB.
BCAD            744          ;
BCAD A9 DE      745          lda #SLPMARK1
BCAF 20 D5 BC    746          jsr WBYTE9
BCB2            747          ;
BCB2 A9 AA      748          lda #SLPMARK2
BCB4 20 D5 BC    749          jsr WBYTE9
BCB7            750          ;
BCB7 A9 EB      751          lda #SLPMARK3
BCB9 20 D5 BC    752          jsr WBYTE9
BCBC            753          ;
BCBC 18          754          clc
BCBD            755          ;
BCBD BD 8E C0    756          ^2  lda DATAIN,X
BCC0 BD 8C C0    757          lda STROBE,X
BCC3            758          ;
BCC3 60          759  WAIT12  rts
BCC4            760          ;
BCC4            761          ;
BCC4            762          ; Write a byte as two four bit nibbles to the disk.
BCC4            763          ;
BCC4            764  WBYTE:
BCC4 48          765          pha
BCC5            766          ;
BCC5 4A          767          lsr
BCC6            768          ;
BCC6 05 3E      769          ora ODDBITSZ
BCC8            770          ;
BCC8            771          ;
BCC8            772          ; Write the odd bits.
BCC8            773          ;
BCC8 9D 8D C0    774          sta LATCH,X
BCCB DD 8C C0    775          cmp STROBE,X
BCCE            776          ;
BCCE 68          777          pla
BCCF            778          ;
BCCF EA          779          nop
BCD0 EA          780          nop
BCD1 EA          781          nop
BCD2            782          ;
BCD2 09 AA      783          ora #ODDBITS
BCD4            784          ;
BCD4 EA          785  WBYTE11  nop
BCD5            786          ;
BCD5 EA          787  WBYTE9   nop
BCD6            788          ;
BCD6 48          789          pha
BCD7 68          790          pla
BCD8            791          ;
BCD8            792          ;
BCD8            793          ; Write the even bits.
BCD8            794          ;

```

```
BCD8 9D 8D C0    795          sta LATCH,X
BCDB DD 8C C0    796          cmp STROBE,X
BCDE             797      ;
BCDE 60          798          rts
BCDF             799      ;
BCDF             800      ;
BCDF             801      ; Force RWTS to a page boundry (33 bytes unused).
BCDF             802      ;
BCDF             803          dfs PAGE_SIZE--PAGE_SIZE**/PAGE_SIZE,ZERO
BD00             804      ;
BD00             805      ;
BD00             806          icl "RWTS2.L"

LLOAD RWTS2.L,A$4000
```

```

BD00          1          ttl "DOS 3.3 Source Code, RWTS2.L"
BD00          2          ;
BD00          3          ;
BD00          4          ; RWTS2.L
BD00          5          ;
BD00          6          ;
BD00          7          ; RWTS entry point.  Upon entry, A-reg & Y-reg point to the
BD00          8          ; IOB.
BD00          9          ;
BD00         10         RWTSENT:
BD00 84 48      11             sty IOBADR
BD02 85 49      12             sta IOBADR+1
BD04          13          ;
BD04          14          ;
BD04          15          ; Set up for one disk head recalibrate and four seeks.
BD04          16          ;
BD04 A0 02      17             ldy #2
BD06 8C F8 06   18             sty RECALCNT
BD09          19          ;
BD09 A0 04      20             ldy #4
BD0B 8C F8 04   21             sty SEEKCNT
BD0E          22          ;
BD0E          23          ;
BD0E          24          ; Get the slot number times 16.
BD0E          25          ;
BD0E A0 01      26             ldy #SNUM16-TBLTYPE
BD10          27          ;
BD10 B1 48      28             lda (IOBADR),Y
BD12 AA         29             tax
BD13          30          ;
BD13          31          ;
BD13          32          ; See if the slot number has changed.
BD13          33          ;
BD13 A0 0F      34             ldy #SLOTFND-TBLTYPE
BD15          35          ;
BD15 D1 48      36             cmp (IOBADR),Y
BD17 F0 1B      37             beq >3
BD19          38          ;
BD19          39          ;
BD19          40          ; Not the same slot so turn off old drive and update
BD19          41          ; SLOTFND.
BD19          42          ;
BD19 8A         43             txa
BD1A 48         44             pha
BD1B          45          ;
BD1B B1 48      46             lda (IOBADR),Y
BD1D AA         47             tax
BD1E          48          ;
BD1E 68         49             pla
BD1F 48         50             pha
BD20          51          ;
BD20 91 48      52             sta (IOBADR),Y
BD22          53          ;
BD22 BD 8E C0   54             lda DATAIN,X
BD25          55          ;
BD25          56          ;
BD25          57          ; Delay until data is constant for 8 reads.
BD25          58          ;
BD25 A0 08      59 ^1         ldy #8
BD27          60          ;

```

```

BD27 BD 8C C0      61      lda STROBE,X
BD2A              62      ;
BD2A DD 8C C0      63      ^2      cmp STROBE,X
BD2D D0 F6         64      bne <1
BD2F              65      ;
BD2F 88           66      dey
BD30 D0 F8         67      bne <2
BD32              68      ;
BD32 68           69      pla
BD33 AA           70      tax
BD34              71      ;
BD34              72      ;
BD34              73      ; Start the motor up.  Data must change within 8 reads.
BD34              74      ;
BD34 BD 8E C0      75      ^3      lda DATAIN,X
BD37 BD 8C C0      76      lda STROBE,X
BD3A              77      ;
BD3A A0 08         78      ldy #8
BD3C              79      ;
BD3C BD 8C C0      80      ^4      lda STROBE,X
BD3F 48           81      pha
BD40              82      ;
BD40 68           83      pla
BD41 48           84      pha
BD42 68           85      pla
BD43              86      ;
BD43 8E F8 05      87      stx SLOTSAV
BD46              88      ;
BD46 DD 8C C0      89      cmp STROBE,X
BD49 D0 03         90      bne >5
BD4B              91      ;
BD4B 88           92      dey
BD4C D0 EE         93      bne <4
BD4E              94      ;
BD4E 08           95      ^5      php
BD4F              96      ;
BD4F BD 89 C0      97      lda MOTORON,X
BD52              98      ;
BD52              99      ;
BD52             100      ; Move necessary pointers to page zero.
BD52             101      ;
BD52 A0 06         102      ldy #DCTADR-TBLTYPE
BD54             103      ;
BD54 B1 48         104      ^6      lda (IOBADR),Y
BD56 99 36 00      105      sta DEVCTBL-DCTADR-TBLTYPE,Y
BD59             106      ;
BD59 C8           107      iny
BD5A             108      ;
BD5A C0 0A         109      cpy #BYTCNT-TBLTYPE
BD5C D0 F6         110      bne <6
BD5E             111      ;
BD5E             112      ;
BD5E             113      ; Get other parameters.  Update DRVFN and turn motor on.
BD5E             114      ;
BD5E A0 03         115      ldy #MOTONTIM-DEVTYPE-1
BD60             116      ;
BD60 B1 3C         117      lda (DEVCTBL),Y
BD62 85 47         118      sta MONTIME+1
BD64             119      ;
BD64 A0 02         120      ldy #DNUM-TBLTYPE
BD66             121      ;

```

```

BD66 B1 48      122      lda (IOBADR),Y
BD68            123      ;
BD68 A0 10      124      ldy #DRVFND-TBLTYPE
BD6A            125      ;
BD6A D1 48      126      cmp (IOBADR),Y
BD6C F0 06      127      beq >7
BD6E            128      ;
BD6E 91 48      129      sta (IOBADR),Y
BD70            130      ;
BD70 28         131      plp
BD71            132      ;
BD71 A0 00      133      ldy #ZERO
BD73            134      ;
BD73 08         135      php
BD74            136      ;
BD74            137      ;
BD74            138      ; Rotate drive number into carry.  If C-flag = 0, drive 2.
BD74            139      ; If C-flag = 1, drive 1.
BD74            140      ;
BD74 6A         141      ^7      ror
BD75 90 05      142      bcc >8
BD77            143      ;
BD77 BD 8A C0   144      lda DRV0EN,X
BD7A B0 03      145      bcs >9
BD7C            146      ;
BD7C BD 8B C0   147      ^8      lda DRV1EN,X
BD7F            148      ;
BD7F            149      ;
BD7F            150      ; Save which drive is being used.
BD7F            151      ;
BD7F 66 35      152      ^9      ror DRIVNO
BD81            153      ;
BD81 28         154      plp
BD82 08         155      php
BD83            156      ;
BD83 D0 0B      157      bne >2
BD85            158      ;
BD85            159      ;
BD85            160      ; Wait 100 usec for old drive's timing capacitor to
BD85            161      ; discharge.
BD85            162      ;
BD85 A0 07      163      ldy #7
BD87            164      ;
BD87 20 00 BA    165      ^1      jsr MSWAIT
BD8A            166      ;
BD8A 88         167      dey
BD8B D0 FA      168      bne <1
BD8D            169      ;
BD8D AE F8 05    170      ldx SLOTSAV
BD90            171      ;
BD90 A0 04      172      ^2      ldy #TNUM-TBLTYPE
BD92            173      ;
BD92 B1 48      174      lda (IOBADR),Y
BD94 20 5A BE    175      jsr MYSEEK
BD97            176      ;
BD97 28         177      plp
BD98 D0 11      178      bne >5
BD9A            179      ;
BD9A A4 47      180      ldy MONTIME+1
BD9C 10 0D      181      bpl >5
BD9E            182      ;

```

```

BD9E      183 ;
BD9E      184 ; Wait for motor to come up to speed.
BD9E      185 ;
BD9E A0 12 186 ^3      ldy #18
BDA0      187 ;
BDA0 88   188 ^4      dey
BDA1 D0 FD 189      bne <4
BDA3      190 ;
BDA3 E6 46 191      inc MONTIME
BDA5 D0 F7 192      bne <3
BDA7      193 ;
BDA7 E6 47 194      inc MONTIME+1
BDA9 D0 F3 195      bne <3
BDAB      196 ;
BDAB      197 ;
BDAB      198 ; Disk is now up to speed. If not format operation then
BDAB      199 ; position the head over the requested track. If C-flag =
BDAB      200 ; 0, then write. If C-Flag = 1, then read.
BDAB      201 ;
BDAB A0 0C 202 ^5      ldy #CMDCODE-TBLTYPE
BDAD      203 ;
BDAD B1 48 204      lda (IOBADR),Y
BDAF F0 5A 205      beq GALLDONE
BDB1      206 ;
BDB1 C9 04 207      cmp #RWTSFRMT
BDB3 F0 58 208      beq FORMDSK
BDB5      209 ;
BDB5 6A    210      ror
BDB6 08    211      php                      ; save R/W status
BDB7 B0 03 212      bcs >6
BDB9      213 ;
BDB9      214 ;
BDB9      215 ; A write operation, so must prenibblize the data first.
BDB9      216 ;
BDB9 20 00 B8 217      jsr PRENIB16
BDBC      218 ;
BDBC      219 ;
BDBC      220 ; Set up for a maximum of 48 retries.
BDBC      221 ;
BDBC A0 30 222 ^6      ldy #48
BDBE 8C 78 05 223      sty RETRYCNT
BDC1      224 ;
BDC1 AE F8 05 225 ^7      ldx SLOTSAV
BDC4      226 ;
BDC4 20 44 B9 227      jsr RDADR16
BDC7 90 24 228      bcc >2
BDC9      229 ;
BDC9 CE 78 05 230 ^8      dec RETRYCNT
BDCC 10 F3 231      bpl <7
BDCE      232 ;
BDCE      233 ;
BDCE      234 ; Recalibrate the disk head. Assume the disk head is at
BDCE      235 ; track 48 (A-reg = 2 * 48).
BDCE      236 ;
BDCE AD 78 04 237 ^9      lda CURTRK
BDD1 48    238      pha
BDD2      239 ;
BDD2 A9 60 240      lda #96
BDD4 20 95 BE 241      jsr SETTRK
BDD7      242 ;
BDD7 CE F8 06 243      dec RECALCNT

```

```

BDDA F0 28      244      beq >3
BDDC              245      ;
BDDC A9 04      246      lda #4
BDDE 8D F8 04   247      sta SEEKCNT
BDE1              248      ;
BDE1 A9 00      249      lda #ZERO
BDE3 20 5A BE   250      jsr MYSEEK
BDE6              251      ;
BDE6 68         252      pla
BDE7              253      ;
BDE7 20 5A BE   254      ^1      jsr MYSEEK
BDEA 4C BC BD   255      jmp <6
BDED              256      ;
BDED              257      ;
BDED              258      ; An address field has now been read.  Now check for
BDED              259      ; desired track, sector, and volume.
BDED              260      ;
BDED A4 2E      261      ^2      ldY TRKFNDZ
BDEF CC 78 04   262      cpy CURTRK
BDF2 F0 1C      263      beq >5
BDF4              264      ;
BDF4              265      ;
BDF4              266      ; Save destination track value.
BDF4              267      ;
BDF4 AD 78 04   268      lda CURTRK
BDF7 48         269      pha
BDF8              270      ;
BDF8 98         271      tya
BDF9 20 95 BE   272      jsr SETTRK
BDFC              273      ;
BDFC 68         274      pla
BDFD              275      ;
BDFD CE F8 04   276      dec SEEKCNT
BE00              277      ;
BE00 D0 E5      278      bne <1
BE02 F0 CA      279      beq <9
BE04              280      ;
BE04              281      ;
BE04              282      ; Bad drive error.
BE04              283      ;
BE04 68         284      ^3      pla
BE05              285      ;
BE05 A9 40      286      lda #RWDRVERR
BE07              287      ;
BE07 28         288      ^4      plp
BE08              289      ;
BE08 4C 48 BE   290      jmp >8
BE0B              291      ;
BE0B              292      ;
BE0B F0 39      293      GALLDONE beq >7
BE0D              294      ;
BE0D 4C AF BE   295      FORMDSK jmp DSKFORM
BE10              296      ;
BE10              297      ;
BE10              298      ; Drive is on the right track.  Now check for a volume
BE10              299      ; mismatch.
BE10              300      ;
BE10 A0 03      301      ^5      ldY #VOLEXPT-TBLTYPE
BE12              302      ;
BE12 B1 48      303      lda (IOBADR),Y      ; get desired volume
BE14 48         304      pha

```

```
BE15          305 ;
BE15          306 ;
BE15          307 ; Save volume actually found in RWTS IOB.
BE15          308 ;
BE15 A5 2F    309         lda VOLFNDZ
BE17          310 ;
BE17 A0 0E    311         ldy #VOLFND-TBLTYPE
BE19          312 ;
BE19 91 48    313         sta (IOBADR),Y
BE1B          314 ;
BE1B          315 ;
BE1B          316 ; If volume specified was zero, no error.
BE1B          317 ;
BE1B 68       318         pla
BE1C F0 08    319         beq >6
BE1E          320 ;
BE1E          321 ; Otherwise, check for a volume mismatch error.
BE1E          322 ;
BE1E C5 2F    323         cmp VOLFNDZ
BE20 F0 04    324         beq >6
BE22          325 ;
BE22 A9 20    326         lda #RWVOLERR
BE24 D0 E1    327         bne <4
BE26          328 ;
BE26          329 ;
BE26          330 ; Now check for the correct sector.
BE26          331 ;
BE26 A0 05    332 ^6         ldy #SNUM-TBLTYPE
BE28          333 ;
BE28 B1 48    334         lda (IOBADR),Y
BE2A          335 ;
BE2A          336 ;
BE2A          337 ; Convert to a "soft" sector number by applying the
BE2A          338 ; software interleave table.
BE2A          339 ;
BE2A A8       340         tay
BE2B          341 ;
BE2B B9 B8 BF 342         lda INTRLEAV,Y
BE2E          343 ;
BE2E          344 ;
BE2E          345 ; Are we at that sector yet?
BE2E          346 ;
BE2E C5 2D    347         cmp SECFNDZ
BE30 D0 97    348         bne <8
BE32          349 ;
BE32          350 ;
BE32          351 ; At correct sector, so see if we are doing a read or a
BE32          352 ; write.
BE32          353 ;
BE32 28       354         plp
BE33 90 1C    355         bcc >9
BE35          356 ;
BE35          357 ;
BE35          358 ; Reading, so read in the 256 bytes of data that follow.
BE35          359 ;
BE35 20 DC B8 360         jsr READ16
BE38 08       361         php
BE39 B0 8E    362         bcs <8
BE3B          363 ;
BE3B 28       364         plp
BE3C          365 ;
```



```

BE3C          366 ;
BE3C          367 ; Convert the nibbles to bytes. First, initialize TEMPZ.
BE3C          368 ;
BE3C A2 00     369         ldx #ZERO
BE3E 86 26     370         stx TEMPZ
BE40          371 ;
BE40 20 C2 B8  372         jsr POSTNB16
BE43          373 ;
BE43 AE F8 05  374         ldx SLOTSAV
BE46          375 ;
BE46 18        376 ^7      clc
BE47          377 ;
BE47 24 48     378         bit IOBADR          ; BIT zero page variable
BE49          379         dfs !-1             ; backup compiler one byte
BE48          380 ;
BE48 38        381 ^8      sec                ; opcode skipped by BIT
BE49          382 ;
BE49 A0 0D     383         ldy #ERRRCODE-TBLTYPE
BE4B          384 ;
BE4B 91 48     385         sta (IOBADR),Y
BE4D          386 ;
BE4D BD 88 C0  387         lda MOTOROFF,X
BE50          388 ;
BE50 60        389         rts
BE51          390 ;
BE51          391 ;
BE51          392 ; Write the data (already nibblized) to the following data
BE51          393 ; sector. On exit C-flag = 0 for okay and C-flag = 1 for
BE51          394 ; error.
BE51          395 ;
BE51 20 2A B8  396 ^9      jsr WRITE16
BE54 90 F0     397         bcc <7
BE56          398 ;
BE56 A9 10     399         lda #RWPROTER
BE58 B0 EE     400         bcs <8
BE5A          401 ;
BE5A          402 ;
BE5A          403 ; MYSEEK is the seek routine. It seeks track 'N' in slot
BE5A          404 ; X/16.
BE5A          405 ;
BE5A          406 ; If DRIVNO is negative - drive 0
BE5A          407 ; If DRIVNO is positive - drive 1
BE5A          408 ;
BE5A          409 MYSEEK:
BE5A 48        410         pha
BE5B          411 ;
BE5B A0 01     412         ldy #PHPERTRK-DEVTYPE
BE5D          413 ;
BE5D B1 3C     414         lda (DEVCTBL),Y
BE5F 6A        415         ror
BE60 68        416         pla
BE61 90 08     417         bcc SEEK1
BE63          418 ;
BE63 0A        419         asl
BE64 20 6B BE  420         jsr SEEK1
BE67          421 ;
BE67 4E 78 04  422         lsr CURTRK
BE6A          423 ;
BE6A 60        424         rts
BE6B          425 ;
BE6B          426 SEEK1:

```

```

BE6B 85 2A      427      sta SEEKTRKZ
BE6D           428      ;
BE6D 20 8E BE   429      jsr XTOY
BE70           430      ;
BE70 B9 78 04   431      lda DRV0TRK,Y
BE73           432      ;
BE73 24 35      433      bit DRIVNO
BE75 30 03      434      bmi >1
BE77           435      ;
BE77 B9 F8 04   436      lda DRV1TRK,Y
BE7A           437      ;
BE7A 8D 78 04   438      ^1 sta CURTRK
BE7D           439      ;
BE7D A5 2A      440      lda SEEKTRKZ
BE7F           441      ;
BE7F 24 35      442      bit DRIVNO
BE81 30 05      443      bmi >2
BE83           444      ;
BE83 99 F8 04   445      sta DRV1TRK,Y
BE86           446      ;
BE86 10 03      447      bpl >3
BE88           448      ;
BE88 99 78 04   449      ^2 sta DRV0TRK,Y
BE8B           450      ;
BE8B 4C A0 B9   451      ^3 jmp SEEKABS
BE8E           452      ;
BE8E           453      ;
BE8E           454      ; Y-reg = X-reg / 16.
BE8E           455      ;
BE8E           456      XTOY:
BE8E 8A         457      txa
BE8F           458      ;
BE8F 4A         459      lsr
BE90 4A         460      lsr
BE91 4A         461      lsr
BE92 4A         462      lsr
BE93           463      ;
BE93 A8         464      tay
BE94           465      ;
BE94 60         466      rts
BE95           467      ;
BE95           468      ;
BE95           469      ; This routine sets the slot dependant track location.
BE95           470      ;
BE95           471      SETTRK:
BE95 48         472      pha
BE96           473      ;
BE96 A0 02      474      ldy #DNUM-TBLTYPE
BE98           475      ;
BE98 B1 48      476      lda (IOBADR),Y
BE9A 6A         477      ror
BE9B           478      ;
BE9B 66 35      479      ror DRIVNO
BE9D           480      ;
BE9D 20 8E BE   481      jsr XTOY
BEA0           482      ;
BEA0 68         483      pla
BEA1 0A         484      asl
BEA2           485      ;
BEA2 24 35      486      bit DRIVNO
BEA4 30 05      487      bmi >1

```

```
BEA6          488 ;
BEA6 99 F8 04 489      sta DRV1TRK,Y
BEA9 10 03     490      bpl >2
BEAB          491 ;
BEAB 99 78 04 492 ^1      sta DRV0TRK,Y
BEAE          493 ;
BEAE 60        494 ^2      rts
BEAF          495 ;
BEAF          496 ;
BEAF          497 ; This is the disk formatter routine.
BEAF          498 ;
BEAF          499 DSKFORM:
BEAF A0 03     500      ldy #VOLEXPT-TBLTYPE
BEB1          501 ;
BEB1 B1 48     502      lda (IOBADR),Y
BEB3 85 41     503      sta VOLUMEZ
BEB5          504 ;
BEB5          505 ;
BEB5          506 ; Save odd bits in zero page (for time critical section).
BEB5          507 ;
BEB5 A9 AA     508      lda #ODDBITS
BEB7 85 3E     509      sta ODDBITSZ
BEB9          510 ;
BEB9 A0 56     511      ldy #NBUF2SIZ
BEBB          512 ;
BEBB          513 ;
BEBB          514 ; Set up to start a track zero.
BEBB          515 ;
BEBB A9 00     516      lda #ZERO
BEDD 85 44     517      sta TRACKZ
BEBF          518 ;
BEBF          519 ;
BEBF          520 ; Zero secondary buffer.
BEBF          521 ;
BEBF 99 FF BB 522 ^1      sta NBUF2-1,Y
BEC2          523 ;
BEC2 88        524      dey
BEC3 D0 FA     525      bne <1
BEC5          526 ;
BEC5          527 ;
BEC5          528 ; Zero primary buffer.
BEC5          529 ;
BEC5 99 00 BB 530 ^2      sta NBUF1,Y
BEC8          531 ;
BEC8 88        532      dey
BEC9 D0 FA     533      bne <2
BECB          534 ;
BECB          535 ;
BECB          536 ; Recalibrate the disk head. Assume the disk head is at
BECB          537 ; track 40 (A-reg = 2 * 40).
BECB          538 ;
BECB A9 50     539      lda #80
BEDD 20 95 BE 540      jsr SETTRK
BED0          541 ;
BED0          542 ;
BED0          543 ; Start with 40 bytes of self sync bytes.
BED0          544 ;
BED0 A9 28     545      lda #40
BED2 85 45     546      sta SYNCNT
BED4          547 ;
BED4          548 ;
```

```

BED4          549      ; Go to the track and format it.
BED4          550      ;
BED4 A5 44     551      ^3      lda TRACKZ
BED6 20 5A BE  552          jsr MYSEEK
BED9          553      ;
BED9 20 0D BF  554          jsr DISKF2
BEDC          555      ;
BEDC          556      ;
BEDC          557      ; Initialize in case of disk I/O error.
BEDC          558      ;
BEDC A9 08     559          lda #RWINITER
BEDE B0 24     560          bcs >5
BEE0          561      ;
BEE0 A9 30     562          lda #48
BEE2 8D 78 05  563          sta RETRYCNT
BEE5          564      ;
BEE5          565      ;
BEE5          566      ; Verify the track just formatted.
BEE5          567      ;
BEE5 38        568      ^4      sec
BEE6          569      ;
BEE6 CE 78 05  570          dec RETRYCNT
BEE9 F0 19     571          beq >5
BEEB          572      ;
BEEB          573      ;
BEEB          574      ; Read the address field. On return C-flag = 1 for error.
BEEB          575      ;
BEEB 20 44 B9  576          jsr RDADR16
BEEE B0 F5     577          bcs <4
BEF0          578      ;
BEF0 A5 2D     579          lda SECFNDZ
BEF2 D0 F1     580          bne <4
BEF4          581      ;
BEF4          582      ;
BEF4          583      ; Read the data and see if it's ok.
BEF4          584      ;
BEF4 20 DC B8  585          jsr READ16
BEF7 B0 EC     586          bcs <4
BEF9          587      ;
BEF9          588      ;
BEF9          589      ; Everything okay so move on to the next track.
BEF9          590      ;
BEF9 E6 44     591          inc TRACKZ
BEFB          592      ;
BEFB          593      ;
BEFB          594      ; At the last track yet?
BEFB          595      ;
BEFB A5 44     596          lda TRACKZ
BEFD C9 23     597          cmp #LASTTRACK
BEFF 90 D3     598          bcc <3
BF01          599      ;
BF01          600      ;
BF01          601      ; No error, so clear C-flag.
BF01          602      ;
BF01 18        603          clc
BF02 90 05     604          bcc >6
BF04          605      ;
BF04          606      ;
BF04          607      ; Store error code in IOB.
BF04          608      ;
BF04 A0 0D     609      ^5      ldy #ERRRCODE-TBLTYPE

```

```
BF06 91 48      610      sta (IOBADR),Y
BF08           611      ;
BF08 38         612      sec
BF09           613      ;
BF09           614      ;
BF09           615      ; Turn the disk drive off.
BF09           616      ;
BF09 BD 88 C0   617      ^6      lda MOTOROFF,X
BF0C           618      ;
BF0C 60         619      rts
BF0D           620      ;
BF0D           621      ;
BF0D           622      ; Format the current track.  Begin with sector zero.
BF0D           623      ;
BF0D           624      DISKF2:
BF0D A9 00      625      lda #ZERO
BF0F 85 3F      626      sta SECTORZ
BF11           627      ;
BF11           628      ;
BF11           629      ; Begin track with 128 sync bytes before writing an
BF11           630      ; address field.
BF11           631      ;
BF11 A0 80      632      ldy #128
BF13 D0 02      633      bne >2
BF15           634      ;
BF15 A4 45      635      ^1      ldy SYNCNT
BF17           636      ;
BF17 20 56 BC   637      ^2      jsr WRADR16
BF1A B0 6B      638      bcs DELAY12
BF1C           639      ;
BF1C           640      ;
BF1C           641      ; Write a data field.
BF1C           642      ;
BF1C 20 2A B8   643      jsr WRITE16
BF1F B0 66      644      bcs DELAY12
BF21           645      ;
BF21           646      ;
BF21           647      ; Increment the sector number and see if it is 16 yet.
BF21           648      ;
BF21 E6 3F      649      inc SECTORZ
BF23           650      ;
BF23 A5 3F      651      lda SECTORZ
BF25 C9 10      652      cmp #16
BF27 90 EC      653      bcc <1
BF29           654      ;
BF29           655      ;
BF29           656      ; Reset sector number to 15.
BF29           657      ;
BF29 A0 0F      658      ldy #15
BF2B 84 3F      659      sty SECTORZ
BF2D           660      ;
BF2D           661      ;
BF2D           662      ; Mark the current track as formatted.
BF2D           663      ;
BF2D A9 30      664      lda #48
BF2F 8D 78 05   665      sta RETRYCNT
BF32           666      ;
BF32 99 A8 BF   667      ^3      sta SECMAP,Y
BF35           668      ;
BF35 88         669      dey
BF36 10 FA      670      bpl <3
```

```

BF38          671 ;
BF38 A4 45    672      ldy SYNCNT
BF3A          673 ;
BF3A 20 87 BF 674 ^4      jsr DELAY12
BF3D 20 87 BF 675      jsr DELAY12
BF40 20 87 BF 676      jsr DELAY12
BF43          677 ;
BF43 48      678      pha
BF44 68      679      pla
BF45          680 ;
BF45 EA      681      nop
BF46          682 ;
BF46 88      683      dey
BF47 D0 F1   684      bne <4
BF49          685 ;
BF49 20 44 B9 686      jsr RDADR16
BF4C B0 23   687      bcs >4
BF4E          688 ;
BF4E A5 2D   689      lda SECFNDZ
BF50 F0 15   690      beq >2
BF52          691 ;
BF52 A9 10   692      lda #16
BF54 C5 45   693      cmp SYNCNT
BF56          694 ;
BF56 A5 45   695      lda SYNCNT
BF58 E9 01   696      sbc #1
BF5A 85 45   697      sta SYNCNT
BF5C          698 ;
BF5C C9 05   699      cmp #5
BF5E B0 11   700      bcs >4
BF60          701 ;
BF60 38      702      sec
BF61 60      703      rts
BF62          704 ;
BF62 20 44 B9 705 ^1      jsr RDADR16
BF65 B0 05   706      bcs >3
BF67          707 ;
BF67 20 DC B8 708 ^2      jsr READ16
BF6A 90 1C   709      bcc >6
BF6C          710 ;
BF6C CE 78 05 711 ^3      dec RETRYCNT
BF6F D0 F1   712      bne <1
BF71          713 ;
BF71 20 44 B9 714 ^4      jsr RDADR16
BF74 B0 0B   715      bcs >5
BF76          716 ;
BF76 A5 2D   717      lda SECFNDZ
BF78 C9 0F   718      cmp #15
BF7A D0 05   719      bne >5
BF7C          720 ;
BF7C 20 DC B8 721      jsr READ16
BF7F 90 8C   722      bcc DISKF2
BF81          723 ;
BF81 CE 78 05 724 ^5      dec RETRYCNT
BF84 D0 EB   725      bne <4
BF86          726 ;
BF86 38      727      sec
BF87          728 ;
BF87 60      729 DELAY12 rts
BF88          730 ;
BF88          731 ;

```

```

BF88      732      ; Mark the map as found.
BF88      733      ;
BF88 A4 2D      734      ^6      ldy SECFNDZ
BF8A      735      ;
BF8A B9 A8 BF   736      lda SECMAP,Y
BF8D 30 DD      737      bmi <3
BF8F      738      ;
BF8F A9 FF      739      lda #NEGONE
BF91 99 A8 BF   740      sta SECMAP,Y
BF94      741      ;
BF94 C6 3F      742      dec SECTORZ
BF96 10 CA      743      bpl <1
BF98      744      ;
BF98 A5 44      745      lda TRACKZ
BF9A D0 0A      746      bne >7
BF9C      747      ;
BF9C A5 45      748      lda SYNCNT
BF9E C9 10      749      cmp #16
BFA0 90 E5      750      bcc DELAY12
BFA2      751      ;
BFA2 C6 45      752      dec SYNCNT
BFA4 C6 45      753      dec SYNCNT
BFA6      754      ;
BFA6 18      755      ^7      clc
BFA7 60      756      rts
BFA8      757      ;
BFA8      758      ;
BFA8      759      ; SECMAP is used to mark initialized sectors.
BFA8      760      ;
BFA8      761      SECMAP      dfs 16,ZERO
BFB8      762      ;
BFB8      763      ;
BFB8      764      ; Interleave remapping table.
BFB8      765      ;
BFB8 00 0D 0B   766      INTRLEAV hex 000D0B09070503010E0C0A080604020F
BFBB 09 07 05
BFBE 03 01 0E
BFC1 0C 0A 08
BFC4 06 04 02
BFC7 0F
BFC8      767      ;
BFC8      768      ;
BFC8      769      ; DOS 3.3 patches. WARNING! These patches are addressed
BFC8      770      ; by object code and should not be moved without carefull
BFC8      771      ; thought.
BFC8      772      ;
BFC8      773      REBOOT:
BFC8 20 93 FE   774      jsr SETVID
BFCB      775      ;
BFCB AD 81 C0   776      lda ROM2WE
BFCE AD 81 C0   777      lda ROM2WE
BFD1      778      ;
BFD1      779      ;
BFD1      780      ; The following will cause a reload of the language card
BFD1      781      ; whether it is desired or not.
BFD1      782      ;
BFD1 A9 00      783      lda #ZERO
BFD3 8D 00 E0   784      sta BASCLD
BFD6      785      ;
BFD6 20 76 BA   786      jsr HBA76
BFD9      787      ;

```

```
BFD9 4C 44 B7 788      jmp RESTART
BFDC          789      ;
BFDC          790      ;
BFDC          791  AJUSTBYT:
BFDC 8D 63 AA 792      sta TEMP1
BFDF 8D 70 AA 793      sta BYTVAL
BFE2 8D 71 AA 794      sta BYTVAL+1
BFE5          795      ;
BFE5 60        796      rts
BFE6          797      ;
BFE6          798      ;
BFE6          799  SETWARM:
BFE6 20 5B A7 800      jsr RSET0
BFE9          801      ;
BFE9 8C B7 AA 802      sty RUNINTRC
BFEC          803      ;
BFEC 60        804      rts
BFED          805      ;
BFED          806      ;
BFED          807  DISKFULL:
BFED 20 7E AE 808      jsr SAVFMW
BFF0          809      ;
BFF0 AE 9B B3 810      ldx STKSAVE
BFF3 9A        811      txs
BFF4          812      ;
BFF4 20 16 A3 813      jsr CLOSEALL
BFF7          814      ;
BFF7 BA        815      tsx
BFF8 8E 9B B3 816      stx STKSAVE
BFFB          817      ;
BFFB A9 09     818      lda #EMSOFF09-EMSGOFFS
BFFD 4C 85 B3 819      jmp SETERROR
C000          820      ;
C000          821      ;
```

BSAVE SEG03,D1,A\$1000,B,L\$0800

```
C000          822      usr SEG03,D1
C000          823      ;
C000          824      ;
CD D2

C000          825      dcm "CD D2"
C000          826      ;
C000          827      ;
C000          828      stt "DOS 3.3 Symbol Table"
C000          829      ;
C000          830      ;
C000          831      end 111
```

*** End of Assembly

Symbol List starts at 0x7800, ends at 0x9592, used 0x1D92, remaining 0x1C9A

Symbols unsorted:

CH	0024	BUFRADRZ	0026	TEMPZ	0026	TEMP2Z	0027	BASEZ	0028
SEEKTRKZ	002A	SLOT16Z	002B	DATAFNDZ	002C	SECFNDZ	002D	TRKFNDZ	002E
VOLFNDZ	002F	PROMPT	0033	DRIVNO	0035	CSWL	0036	KSWL	0038
ROMTEMPZ	003C	DEVCTBL	003C	ROMSECTR	003D	BOOTADRZ	003E	BUFADR2Z	003E
ODDBITSZ	003E	SECTORZ	003F	ROMDATA	0040	FILEBUFZ	0040	ROMTRACK	0041
VOLUMEZ	0041	BUFADRZ	0042	TRACKZ	0044	OPRND	0044	SYNCNT	0045
MONTIME	0046	IOBADR	0048	INTLOMEM	004A	INTHIMEM	004C	ASPGMST	0067
ASVARS	0069	ASSTRS	006F	ASHIMEM	0073	ASRUN	0076	ASPEND	00AF
INTSTRT	00CA	INTVEND	00CC	PROTECT	00D6	ASONERR	00D8	INTRUN	00D9
INTEGER	0000	ZERO	0000	FLASHSPC	0060	ASCIMASK	007F	ASCIFLAG	0080
CTRLD	0084	LINEFEED	008A	RETURN	008D	SPACE	00A0	LWRCASE	00E0
DEFLTVOL	00FE	NEGONE	00FF	LNSPERPG	0016	NAMESIZE	001E	LASTRACK	0023
INTBASIC	0020	FPBASIC	004C	PWRUPBYT	00A5	INTACTV	0000	FPOACTV	0040
FPAACTV	0080	STATE0	0000	STATE1	0001	STATE2	0002	STATE3	0003
STATE4	0004	STATE5	0005	STATE6	0006	READMODE	0001	MONOMASK	0010
MONIMASK	0020	MONCMASK	0040	MONMASK	007F	MONFLAG	0080	TXTFLTYP	0000
INTFLTYP	0001	ASFLTYP	0002	BINFLTYP	0004	SFILETYP	0008	RECFLTYP	0010
AFILETYP	0020	BFILETYP	0040	LOCKMASK	007F	LOCKFLAG	0080	DELETFLG	00FF
NWFLMSK	0001	PGCMDMSK	0002	MXFLEMSK	0004	SLNUMMSK	0008	FLNM2MSK	0010
FLNM1MSK	0020	CMDOPMSK	0040	FNOPMSK	0080	AKEYMASK	0001	BKEYMASK	0002
RKEYMASK	0004	LKEYMASK	0008	SKEYMASK	0010	DKEYMASK	0020	VKEYMASK	0040
MKEYMASK	0080	VTOCMASK	0002	DATAMASK	0040	TSBFRMSK	0080	FMOPENCD	0001
FMCLOSED	0002	FMREADCD	0003	FMWRITCD	0004	FMDELECD	0005	FMCATACD	0006
FMLOCKCD	0007	FMUNLKCD	0008	FMRENMC	0009	FMPOSICD	000A	FMINITCD	000B
FMVERICD	000C	FMNOOPSC	0000	FMRW01SC	0001	FMRWNBSC	0002	FMPOS1SC	0003
FMPOSNSC	0004	FMNOERR	0000	FMLNAERR	0001	FMOPTERR	0002	FMSUBERR	0003
FMPROTER	0004	FMEOFERR	0005	FMNOFLER	0006	FMVOLERR	0007	FMDIOERR	0008
FMFULLER	0009	FMLOCKER	000A	RWTSSEEK	0000	RWTSREAD	0001	RWTSWRIT	0002
RWTSFRMT	0004	RWNOERR	0000	RWINITER	0008	RWPROTER	0010	RWVOLERR	0020
RWDRVERR	0040	RWREADER	0080	TSTRKOFF	0001	TSSECOFF	0002	TSRECOFF	0005
TSLSTOFF	000C	VTOCTRK	0011	VTOCENSZ	0023	VTOCEND	00F5	ALOCFRWD	0001
ALOCBKWD	00FF	NBUFMASK	003F	NBUF2SIZ	0056	ODDBITS	00AA	ADRMARK1	00D5
ADRMARK2	00AA	ADRMARK3	0096	DATMARK1	00D5	DATMARK2	00AA	DATMARK3	00AD
SLPMARK1	00DE	SLPMARK2	00AA	SLPMARK3	00EB	SYNCMARK	00FF	PAGESIZE	0100
STACK	0100	INPUT	0200	NBUF2BT	0300	RDNIBLBT	036C	DOSRST	03D0
SOFTEV	03F2	PWREDUP	03F4	CURTRK	0478	SEEKCNT	04F8	XMODE	04FB
DRV0TRK	0478	DRV1TRK	04F8	RETRYCNT	0578	SLOTSAV	05F8	SLOTABS	0678
RECALCNT	06F8	PAGE08	0800	BOOTVALS	08FD	ASRAMWRM	0C3C	ASRAMRST	0CF2
ASRAMCLR	0E65	ASRAMNEW	0FD4	PAGE10	1000	ASRAMERR	1067	DOSFBADR	9AA6
MEMTOP	C000	VID80OFF	C00C	ALTCHOFF	C00E	RAM2WP	C080	ROM2WE	C081
PHASEOFF	C080	PHASEON	C081	MOTOROFF	C088	MOTORON	C089	DRV0EN	C08A
DRV1EN	C08B	STROBE	C08C	LATCH	C08D	DATAIN	C08E	DATAOUT	C08F
ASROMWRM	D43C	ASROMRST	D4F2	ASROMCLR	D665	ASROMNEW	D7D2	ASROMERR	D865
BASCLD	E000	BASWRM	E003	INTERR	E3E3	IRUN	E836	OLDBRK	FA59
INIT	FB2F	HOME	FC58	WAIT	FCA8	RDKEY	FD0C	PRBYTE	FDDA
COUT	FDED	SETKBD	FE89	INPORT	FE8B	SETVID	FE93	OUTPORT	FE95
IORTS	FF58	MON	FF65	BOOTFW	C05C	BOOTFW2	C05D	FNDADDR	C083
FNDDATA	C0A6	DATABUFR	9AA6	TSBUFFER	9BA6	WORKAREA	9CA6	TSFRSTTS	9CA6
TSCURRTS	9CA8	WAFLAGS	9CAA	TSCURDAT	9CAB	SECATOFF	9CAD	BYCATOFF	9CAE
MAXTSECR	9CAF	SECFRSTS	9CB1	SECLASTS	9CB3	SECLSTRD	9CB5	SECRSIZE	9CB7
SECRPOST	9CB9	BYSECOFF	9CBB	RECDLNGH	9CBD	RECURNUM	9CBF	BYRECOFF	9CC1
SECFILEN	9CC3	SECALOTR	9CC5	CURALOTR	9CC6	SECFRETR	9CC7	WAFILTYP	9CCB
WASLTNUM	9CCC	WADRVNUM	9CCD	WAVOLNUM	9CCE	WATRKNUM	9CCF	FILNAMBF	9CD3
WABUFADR	9CF1	TSBUFADR	9CF3	DABUFADR	9CF5	NXTFNADR	9CF7	BUFREND	9CF9
DOSTART	9D00	DBUFP	9D00	DOSKBD	9D02	DOSVID	9D04	PFNADR	9D06
SFNADR	9D08	LDRNGLEN	9D0A	LOADADR	9D0C	FMPARMS	9D0E	CSWSTADR	9D10

CMDTBL	9D1E	CMDINIT	9D1E	CMDLOAD	9D20	CMDSAVE	9D22	CMDRUN	9D24
CMDCHAIN	9D26	CMDDELETE	9D28	CMDLOCK	9D2A	CMDUNLCK	9D2C	CMDCLOSE	9D2E
CMDREAD	9D30	CMDEXEC	9D32	CMDWRITE	9D34	CMDPOSTN	9D36	CMDOPEN	9D38
CMDAPND	9D3A	CMDRENAM	9D3C	CMDCAT	9D3E	CMDMON	9D40	CMDNOMON	9D42
CMDPRNUM	9D44	CMDINNUM	9D46	CMDMXFLS	9D48	CMDFP	9D4A	CMDINT	9D4C
CMDBSAVE	9D4E	CMDBLOAD	9D50	CMDBRUN	9D52	CMDVERFY	9D54	CHNADR	9D56
RUNADR	9D58	BASERR	9D5A	BASCOLD	9D5C	BASWARM	9D5E	ASFTREL	9D60
INTTBL	9D62	FPTBL	9D6C	RAMASTBL	9D78	DOSSTRT	9D84	DOSWARM	9DBF
INITDOS	9DD1	FRSTIME	9DEA	TSTPEND	9E45	PAG3CODE	9E51	PAG3BGN	03D0
PAG3END	0400	KBDINTRC	9E81	RDCHAR	9EBA	VIDINTRC	9EBD	SAVREG	9ED1
CSWST0	9EEB	CSWST1	9F12	CSWST1.1	9F15	CSWST2	9F23	CSWST2.1	9F27
CSWST3	9F2F	CSWST4	9F52	CSWST4.1	9F5B	CSWST5	9F61	CSWST6	9F71
RUNDONE	9F78	SCNXIT	9F83	ECHOC	9F95	ECHOO	9F99	ECHOI	9F9D
ECHO	9FA4	DOSXIT	9FB3	REGRST	9FBA	CMDCOUT	9FC5	CRLF	9FC8
PARSE	9FCD	GETFRST	A000	CMPUTIDX	A01B	CHKFRST	A092	BLKNAME\$	A095
BLKNAME2	A097	FNDNONAM	A0A0	SETDFLTS	A0D1	GETNXT	A0E8	PROCMD	A17A
DOCMD	A180	GNXTCHR	A193	FLSHCMDL	A1A4	CLRFMP	A1AE	GETNUM	A1B9
MUL2	A1FE	DOPRNUM	A229	DOINNUM	A22E	DOMON	A233	DONOMON	A23D
DOMXFLS	A251	DODELETE	A263	DOLOCK	A271	DOUNLOCK	A275	DOVERIFY	A27D
DORENAME	A281	DOAPND	A298	DOOPEN	A2A3	CMDHNDLR	A2A8	CMDHNDL2	A2AA
CLOSOPN	A2C8	DOCLOSE	A2EA	CLOSFREE	A2FC	CLOSEALL	A316	DOBSAVE	A331
DOBLOAD	A35D	DOBRUN	A38E	DOSAVE	A397	OPENTST	A3D5	WRT2BYT	A3E0
RWRANGE	A3FF	RWRANGE2	A40A	DOLOAD	A413	DOLOAD2	A416	RWSETUP	A471
RD2BYT	A47A	PTOOLRG	A4AB	SELBASIC	A4B1	DORUN	A4D1	DORUN2	A4DC
DOSIRUN	A4E5	DOCHAIN	A4F0	DOSARUN	A4FC	DOSARUN2	A506	DOWRITE	A510
DOREAD	A51B	RWCOMMON	A526	RWCOMMN2	A546	DOINIT	A54F	DOCAT	A56E
DOFP	A57A	DOINT	A59E	SETBASIC	A5B2	DOEXEC	A5C6	DOPSTION	A5DD
WRITBYTE	A60E	READBYTE	A626	ISBASRUN	A65E	CLSWARM	A679	EXECRD	A682
RDTEXT	A68C	PNTEXEC	A69D	FMDRVR	A6A8	FMDRVR2	A6BB	CSYNERR	A6C4
NOBUF	A6C8	PTOOBIG	A6CC	FMISMTCH	A6D0	DOERROR	A6D2	PRTERORR	A702
COPYPARM	A71A	COPYNAME	A743	COPYPTRS	A74E	RSET0	A75B	LOCBUF	A764
FRSTBUF	A792	PNTNXTBF	A79A	GFBFNAM	A7AA	ISEXCBUF	A7AF	CHKTYPE	A7C4
INITBUFS	A7D4	FREENAME	A7E5	INITPTRS	A851	DOSCMDS	A884	KWRDPRMS	A909
PPARMS	A941	PARMBITS	A94B	KWRANGE	A955	ERRMSG\$	A971	ERRMSG0	A971
ERRMSG1	A974	ERRMSG2	A98A	ERRMSG3	A995	ERRMSG4	A9A4	ERRMSG5	A9AF
ERRMSG6	A9BD	ERRMSG7	A9CC	ERRMSG8	A9D5	ERRMSG9	A9DE	ERRMSGA	A9E9
ERRMSGB	A9F5	ERRMSGC	AA09	ERRMSGD	AA1B	ERRMSG\$E	AA2C	EMSGOFF\$	AA3F
EMSOFF00	AA3F	EMSOFF01	AA40	EMSOFF02	AA41	EMSOFF03	AA42	EMSOFF04	AA43
EMSOFF05	AA44	EMSOFF06	AA45	EMSOFF07	AA46	EMSOFF08	AA47	EMSOFF09	AA48
EMSOFF10	AA49	EMSOFF11	AA4A	EMSOFF12	AA4B	EMSOFF13	AA4C	EMSOFF14	AA4D
EMSOFF15	AA4E	BUFADR	AA4F	CURSTAT	AA51	CSWSTATE	AA52	CSWLSAV	AA53
KSWLSAV	AA55	MAXFILES	AA57	SSAVE	AA59	XSAVE	AA5A	YSAVE	AA5B
ASAVE	AA5C	CMDLNIDX	AA5D	MONFLGS	AA5E	CMDINDX	AA5F	LOADLEN	AA60
PENDCMD	AA62	TEMP1	AA63	KYWRDIDX	AA64	KYWRDFND	AA65	VOLVAL	AA66
DRVAL	AA68	SLOTVAL	AA6A	LENVAL	AA6C	RECVAL	AA6E	BYTVAL	AA70
ADRVAL	AA72	MONVAL	AA74	FNAME	AA75	SFNAME	AA93	MXFLS	AAB1
CTLD	AAB2	EXFLG	AAB3	EXCBUF	AAB4	WHCBASIC	AAB6	RUNINTRC	AAB7
PGMNAME	AAB8	RWTSPADR	AAC1	VTOCPADR	AAC3	DIRPADR	AAC5	FMFTBL	AAC9
FMRSUB	AAE5	FMWSUB	AAF1	FMEXT	AAFD	FILEMNGR	AB06	OPNHNDLR	AB22
CMNOPN	AB28	INITFMW	ABDC	CLSHNDLR	AC06	RENHNDLR	AC3A	RDHNDLR	AC58
WRTHNDLR	AC70	POSRD1B	AC87	RD1BYTE	AC8A	POSRDR	AC93	RDRANGE	AC96
RDABYTE	ACA8	POSWRT1B	ACBB	WRT1BYTE	ACBE	POSWRTR	ACC7	WRTRANGE	ACCA
WRTBYTE	ACDA	LCKHNDLR	ACEF	UNLKHNDL	ACF6	POSHNDLR	AD12	VFYHNDLR	AD18
DELHNDLR	AD2B	FREESECT	AD89	CATHNDLR	AD98	SKIPLN	AE2F	PRTDEC	AE42
LDFMW	AE6A	SAVFMW	AE7E	INITHNDL	AE8E	SELBUF	AF08	SELTSBUF	AF0C
SELDABUF	AF10	CHKBUF	AF1D	CHKTS	AF34	SETUPRW	AF3A	PREPRWTS	AF4B
RDTSLIST	AF5E	OLDRWTS	AFB5	RDDASEC	AFDC	PREPDATA	AFE4	RDVTOC	AFF7
WRTVTOC	AFFB	RDDIRSEC	B011	WRTDIRSC	B037	PRWTSDIR	B045	RWTS DRV R	B052
SETCMDCD	B058	RDNXTDA	B0B6	ADDATA	B134	INCREC	B15B	INCPOS	B194
MOVRANG	B1A2	DECRNG	B1B5	LCDIRENT	B1C9	COPYFNAM	B21C	NXTDIREN	B230
ALLOCSEC	B244	RLSALLC	B2C3	RORBITMP	B2DD	CALPOSN	B300	LNOTAVL	B35F

RANGERR	B363	RANGERR2	B367	ENDATA	B36F	FNOTFND	B373	DSKFULL	B377
FILELOCK	B37B	NOERROR	B37F	SETERROR	B385	DIRTS	B397	STKSAVE	B39B
DIRINDX	B39C	CNTR	B39D	ALLCFLG	B39E	FREEMASK	B3A0	DECTBL	B3A4
FTTBL	B3A7	DISKVOL	B3AF	DSKVOLND	B3BB	VTCSB	B3BB	FRSTTS	B3BC
DOSRLS	B3BE	DSKVOL	B3C1	NUMTSENT	B3E2	NXTTOALC	B3EB	ALLCDIR	B3EC
NUMTRKS	B3EF	NUMSCTRS	B3F0	BYTPRSEC	B3F1	BITMAP	B3F3	DIRSECBF	B4BB
TSNXTDIR	B4BC	TSTRACK	B4C6	TSSECTOR	B4C7	FILTYP	B4C8	FILNAME	B4C9
FILESIZE	B4E7	FMOPCOD	B5BB	SUBCODE	B5BC	RECNUM	B5BD	BYTOFFST	B5BF
VOLUME	B5BF	DRIVE	B5C0	BYTRANGE	B5C1	SLOT	B5C1	FILETYPE	B5C2
DATADR	B5C3	DATBYTE	B5C3	FNADR	B5C3	RTNCODE	B5C5	WBADR	B5C7
TSLSTADR	B5C9	DATASADR	B5CB	FTSTS	B5D1	FTSS	B5D2	CURTSTS	B5D3
CURTSS	B5D4	FLAGS	B5D5	CURDATS	B5D6	CURDAS	B5D7	DIRSECIX	B5D8
DIRBYTIX	B5D9	SECPERTS	B5DA	RELSFRST	B5DC	RELSLAST	B5DE	RELSLRD	B5E0
SECTLEN	B5E2	FILEPOSN	B5E4	FILEBYTE	B5E6	OPNRCLN	B5E8	RECNUMBR	B5EA
BYTEOFFS	B5EC	SECCNT	B5EE	NEXTSECR	B5F0	CURTRACK	B5F1	SECBTMAP	B5F2
FTYPE	B5F6	SLOT16	B5F7	DRVNUMBR	B5F8	VOLNUMBR	B5F9	TRKNUMBR	B5FA
FMWAEND	B5FE	BOOTCODE	B600	BOOTBGN	0800	INTRLTBL	084D	BOOTEND	085D
PATCHBGN	B65D	APPFLG	B65D	APNDPTCH	B65E	APTCH2	B671	VFYPTCH	B686
APTCH3	B692	PATCHEND	B6E8	PTCHSTOP	B6FD	VALSBGN	08FD	BOOTJMP	08FD
BOOTADR	08FE	BOOTPGS	08FF	VALSEND	0900	RESTART	B744	PUTDOS	B74A
RWPAGES	B793	CALLRWTS	B7B5	RWTSPRMS	B7C2	ZEROBUFR	B7D6	NUMPGS	B7E0
NSECSRW	B7E1	RWTSPPTR	B7E4	FRSTBOOT	B7E6	TBLTYPE	B7E8	SNUM16	B7E9
DNUM	B7EA	VOLEXPT	B7EB	TNUM	B7EC	SNUM	B7ED	DCTADR	B7EE
USRBUF	B7F0	BYTCNT	B7F2	CMDCODE	B7F4	ERRRCODE	B7F5	VOLFND	B7F6
SLOTFND	B7F7	DRVFND	B7F8	DEVTYPE	B7FB	PHPERTRK	B7FC	MOTONTIM	B7FD
PRENIB16	B800	WRITE16	B82A	WNIBL9	B8B8	WNIBL7	B8B9	WNIBL	B8BB
POSTNB16	B8C2	READ16	B8DC	RDERR	B942	RDADR16	B944	RDADRX	B99E
SEEKABS	B9A0	ISTHERE	B9EA	CHKPOS	B9EE	CHKPOS2	B9F1	MSWAIT	BA00
ONTBL	BA11	OFFTBL	BA1D	WRNIBL	BA29	HBA69	BA69	HBA76	BA76
HBA84	BA84	RDNIBL	BA96	NBUF1	BB00	NBUF2	BC00	WRADR16	BC56
WAIT12	BCC3	WBYTE	BCC4	WBYTE11	BCD4	WBYTE9	BCD5	RWTSENT	BD00
GALLDONE	BE0B	FORMDSK	BE0D	MYSEEK	BE5A	SEEK1	BE6B	XTOY	BE8E
SETTRK	BE95	DSKFORM	BEAF	DISKF2	BF0D	DELAY12	BF87	SECMAP	BFA8
INTRLEAV	BFB8	REBOOT	BFC8	AJUSTBYT	BFDC	SETWARM	BFE6	DISKFULL	BFED

Symbols alphabetically sorted:

ADDDATA	B134	ADRMARK1	00D5	ADRMARK2	00AA	ADRMARK3	0096	ADRVAL	AA72
AFILETYP	0020	AJUSTBYT	BFDC	AKEYMASK	0001	ALLCDIR	B3EC	ALLCFLG	B39E
ALLOCSEC	B244	ALOCBKWD	00FF	ALOCFRWD	0001	ALTCHOFF	C00E	APNDPTCH	B65E
APPFLG	B65D	APTCH2	B671	APTCH3	B692	ASAVE	AA5C	ASCIFLAG	0080
ASCIMASK	007F	ASFLTYP	0002	ASFTREL	9D60	ASHIMEM	0073	ASONERR	00D8
ASPEND	00AF	ASPGMST	0067	ASRAMCLR	0E65	ASRAMERR	1067	ASRAMNEW	0FD4
ASRAMRST	0CF2	ASRAMWRM	0C3C	ASROMCLR	D665	ASROMERR	D865	ASROMNEW	D7D2
ASROMRST	D4F2	ASROMWRM	D43C	ASRUN	0076	ASSTRS	006F	ASVARS	0069
BASCLD	E000	BASCOLD	9D5C	BASERR	9D5A	BASEZ	0028	BASWARM	9D5E
BASWRM	E003	BFILETYP	0040	BINFLTYP	0004	BITMAP	B3F3	BKEYMASK	0002
BLKNAME2	A097	BLKNAMES	A095	BOOTADR	08FE	BOOTADRZ	003E	BOOTBGN	0800
BOOTCODE	B600	BOOTEND	085D	BOOTFW	C05C	BOOTFW2	C05D	BOOTJMP	08FD
BOOTPGS	08FF	BOOTVALS	08FD	BUFADR	AA4F	BUFADR2Z	003E	BUFADRZ	0042
BUFRADRZ	0026	BUFREND	9CF9	BYCATOFF	9CAE	BYRECOFF	9CC1	BYSECOFF	9CBB
BYTCNT	B7F2	BYTEOFFS	B5EC	BYTOFFST	B5BF	BYTPRSEC	B3F1	BYTRANGE	B5C1
BYTVAL	AA70	CALLRWTS	B7B5	CALPOSN	B300	CATHNDLR	AD98	CH	0024
CHKBUF	AF1D	CHKFRST	A092	CHKPOS	B9EE	CHKPOS2	B9F1	CHKTS	AF34
CHKTYPE	A7C4	CHNADR	9D56	CLOSEALL	A316	CLOSFREE	A2FC	CLOSOPN	A2C8
CLRFMP	A1AE	CLSHNDLR	AC06	CLSWARM	A679	CMDAPND	9D3A	CMDBLOAD	9D50
CMDBRUN	9D52	CMDBSAVE	9D4E	CMDCAT	9D3E	CMDCHAIN	9D26	CMDCLOSE	9D2E
CMDCODE	B7F4	CMDCOUT	9FC5	CMDDELET	9D28	CMDEXEC	9D32	CMDFP	9D4A
CMDHNDL2	A2AA	CMDHNDLR	A2A8	CMDINDX	AA5F	CMDINIT	9D1E	CMDINNUM	9D46

CMDINT	9D4C	CMDLNIDX	AA5D	CMDLOAD	9D20	CMDLOCK	9D2A	CMDMON	9D40
CMDMXFLS	9D48	CMDNOMON	9D42	CMDOPEN	9D38	CMDOPMSK	0040	CMDPOSTN	9D36
CMDPRNUM	9D44	CMDREAD	9D30	CMDRENAM	9D3C	CMDRUN	9D24	CMDSAVE	9D22
CMDTBL	9D1E	CMDUNLCK	9D2C	CMDVERFY	9D54	CMDWRITE	9D34	CMNOPN	AB28
CMPUTIDX	A01B	CNTR	B39D	COPYFNAM	B21C	COPYNAME	A743	COPYPARM	A71A
COPYPTRS	A74E	COUT	FD0D	CRLF	9FC8	CSWL	0036	CSWLSAV	AA53
CSWST0	9EEB	CSWST1	9F12	CSWST1.1	9F15	CSWST2	9F23	CSWST2.1	9F27
CSWST3	9F2F	CSWST4	9F52	CSWST4.1	9F5B	CSWST5	9F61	CSWST6	9F71
CSWSTADR	9D10	CSWSTATE	AA52	CSYNERR	A6C4	CTLD	AAB2	CTRLD	0084
CURALOTR	9CC6	CURDAS	B5D7	CURDATS	B5D6	CURSTAT	AA51	CURTRACK	B5F1
CURTRK	0478	CURTSS	B5D4	CURTSTS	B5D3	DABUFADR	9CF5	DATABUFR	9AA6
DATADR	B5C3	DATAFNDZ	002C	DATAIN	C08E	DATAMASK	0040	DATAOUT	C08F
DATASADR	B5CB	DATBYTE	B5C3	DATMARK1	00D5	DATMARK2	00AA	DATMARK3	00AD
DBUFP	9D00	DCTADR	B7EE	DECRNG	B1B5	DECTBL	B3A4	DEFLTVOL	00FE
DELAY12	BF87	DELETFLG	00FF	DELHNDLR	AD2B	DEVCTBL	003C	DEVTYPE	B7FB
DIRBYTIX	B5D9	DIRINDX	B39C	DIRPADR	AAC5	DIRSECBF	B4BB	DIRSECIX	B5D8
DIRTS	B397	DISKF2	BF0D	DISKFULL	BFED	DISKVOL	B3AF	DKEYMASK	0020
DNUM	B7EA	DOAPND	A298	DOBLOAD	A35D	DOBRUN	A38E	DOBSAVE	A331
DOCAT	A56E	DOCHAIN	A4F0	DOCLOSE	A2EA	DOCMD	A180	DODELETE	A263
DOERROR	A6D2	DOEXEC	A5C6	DOFP	A57A	DOINIT	A54F	DOINNUM	A22E
DOINT	A59E	DOLOAD	A413	DOLOAD2	A416	DOLOCK	A271	DOMON	A233
DOMXFLS	A251	DONOMON	A23D	DOOPEN	A2A3	DOPRNUM	A229	DOPSTION	A5DD
DOREAD	A51B	DORENAME	A281	DORUN	A4D1	DORUN2	A4DC	DOSARUN	A4FC
DOSARUN2	A506	DOSAVE	A397	DOSCMDS	A884	DOSFBADR	9AA6	DOSIRUN	A4E5
DOSKBD	9D02	DOSRLS	B3BE	DOSRST	03D0	DOSSTRT	9D84	DOSTART	9D00
DOSVID	9D04	DOSWARM	9DBF	DOSXIT	9FB3	DOUNLOCK	A275	DOVERIFY	A27D
DOWRITE	A510	DRIVE	B5C0	DRIVNO	0035	DRV0EN	C08A	DRV0TRK	0478
DRV1EN	C08B	DRV1TRK	04F8	DRVAL	AA68	DRVFND	B7F8	DRVNUMBR	B5F8
DSKFORM	BEAF	DSKFULL	B377	DSKVOL	B3C1	DSKVOLND	B3BB	ECHO	9FA4
ECHOC	9F95	ECHOI	9F9D	ECHOO	9F99	EMSGOFFS	AA3F	EMSOFF00	AA3F
EMSOFF01	AA40	EMSOFF02	AA41	EMSOFF03	AA42	EMSOFF04	AA43	EMSOFF05	AA44
EMSOFF06	AA45	EMSOFF07	AA46	EMSOFF08	AA47	EMSOFF09	AA48	EMSOFF10	AA49
EMSOFF11	AA4A	EMSOFF12	AA4B	EMSOFF13	AA4C	EMSOFF14	AA4D	EMSOFF15	AA4E
ENDATA	B36F	ERRMSG0	A971	ERRMSG1	A974	ERRMSG2	A98A	ERRMSG3	A995
ERRMSG4	A9A4	ERRMSG5	A9AF	ERRMSG6	A9BD	ERRMSG7	A9CC	ERRMSG8	A9D5
ERRMSG9	A9DE	ERRMSGA	A9E9	ERRMSGB	A9F5	ERRMSGC	AA09	ERRMSGD	AA1B
ERRMSGGE	AA2C	ERRMSGGS	A971	ERRRCODE	B7F5	EXCBUF	AAB4	EXECRD	A682
EXFLG	AAB3	FILEBUFZ	0040	FILEBYTE	B5E6	FILELOCK	B37B	FILEMNGR	AB06
FILEPOSN	B5E4	FILESIZE	B4E7	FILETYPE	B5C2	FILNAMBF	9CD3	FILNAME	B4C9
FILTYP	B4C8	FLAGS	B5D5	FLASHSPC	0060	FLNM1MSK	0020	FLNM2MSK	0010
FLSHCMDL	A1A4	FMCATACD	0006	FMCLOSCD	0002	FMDELECD	0005	FMDIOERR	0008
FMDRVR	A6A8	FMDRVR2	A6BB	FMEOFERR	0005	FMEXT	AAFD	FMFTBL	AAC9
FMFULLER	0009	FMINITCD	000B	FMISMTCH	A6D0	FMLNAERR	0001	FMLOCKCD	0007
FMLOCKER	000A	FMNOERR	0000	FMNOFLER	0006	FMNOOPSC	0000	FMOPCOD	B5BB
FMOPENCD	0001	FMOPTERR	0002	FMPARMS	9D0E	FMPOS1SC	0003	FMPOSICD	000A
FMPOSNSC	0004	FMPROTER	0004	FMREADCD	0003	FMRENMCD	0009	FMRSUB	AAE5
FMRW01SC	0001	FMRWNBSC	0002	FMSUBERR	0003	FMUNLKCD	0008	FMVERICD	000C
FMVOLERR	0007	FMWAEND	B5FE	FMWRITCD	0004	FMWSUB	AAF1	FNADR	B5C3
FNAME	AA75	FNDADDR	C083	FNDDATA	C0A6	FNDNONAM	A0A0	FNOPMSK	0080
FNOTFND	B373	FORMDSK	BE0D	FPAACTV	0080	FPBASIC	004C	FPOACTV	0040
FPTBL	9D6C	FREEMASK	B3A0	FREENAME	A7E5	FREESECT	AD89	FRSTBOOT	B7E6
FRSTBUF	A792	FRSTIME	9DEA	FRSTTS	B3BC	FTSS	B5D2	FTSTS	B5D1
FTTBL	B3A7	FTYPE	B5F6	GALLDONE	BE0B	GETFRST	A000	GETNUM	A1B9
GETNXT	A0E8	GFBFNAM	A7AA	GNXTCHR	A193	HBA69	BA69	HBA76	BA76
HBA84	BA84	HOME	FC58	INCPOS	B194	INCREC	B15B	INIT	FB2F
INITBUFS	A7D4	INITDOS	9DD1	INITFMW	ABDC	INITHNDL	AE8E	INITPTRS	A851
INPORT	FE8B	INPUT	0200	INTACTV	0000	INTBASIC	0020	INTEGER	0000
INTERR	E3E3	INTFLTYP	0001	INTHIMEM	004C	INTLOMEM	004A	INTRLEAV	BFB8
INTRLTBL	084D	INTRUN	00D9	INTSTRT	00CA	INTTBL	9D62	INTVEND	00CC
IOBADR	0048	IORTS	FF58	IRUN	E836	ISBASRUN	A65E	ISEXCBUF	A7AF
ISTHERE	B9EA	KBDINTRC	9E81	KSWL	0038	KSWLSAV	AA55	KWRANGE	A955

KWRDPRMS	A909	KYWRDFND	AA65	KYWRDIDX	AA64	LASTRACK	0023	LATCH	C08D
LCDIRENT	B1C9	LCKHNDLR	ACEF	LDFMW	AE6A	LDRNGLEN	9D0A	LENVAL	AA6C
LINEFEED	008A	LKEYMASK	0008	LNOTAVL	B35F	LNSPERPG	0016	LOADADR	9D0C
LOADLEN	AA60	LOCBUF	A764	LOCKFLAG	0080	LOCKMASK	007F	LWRCASE	00E0
MAXFILES	AA57	MAXTSECR	9CAF	MEMTOP	C000	MKEYMASK	0080	MON	FF65
MONCMASK	0040	MONFLAG	0080	MONFLGS	AA5E	MONIMASK	0020	MONMASK	007F
MONOMASK	0010	MONTIME	0046	MONVAL	AA74	MOTONTIM	B7FD	MOTOROFF	C088
MOTORON	C089	MOVRANG	B1A2	MSWAIT	BA00	MUL2	A1FE	MXFLEMSK	0004
MXFLS	AAB1	MYSEEK	BE5A	NAMESIZE	001E	NBUF1	BB00	NBUF2	BC00
NBUF2BT	0300	NBUF2SIZ	0056	NBUFMASK	003F	NEGONE	00FF	NEXTSECR	B5F0
NOBUF	A6C8	NOERROR	B37F	NSECSRW	B7E1	NUMPGS	B7E0	NUMSCTRS	B3F0
NUMTRKS	B3EF	NUMTSENT	B3E2	NWFLMSK	0001	NXTDIREN	B230	NXTFNADR	9CF7
NXTTOALC	B3EB	ODDBITS	00AA	ODDBITSZ	003E	OFFTBL	BA1D	OLDBRK	FA59
OLDRWTS	AFB5	ONTBL	BA11	OPENTST	A3D5	OPNHNDLR	AB22	OPNRCLEN	B5E8
OPRND	0044	OUTPORT	FE95	PAG3BGN	03D0	PAG3CODE	9E51	PAG3END	0400
PAGE08	0800	PAGE10	1000	PAGESIZE	0100	PARMBITS	A94B	PARSE	9FCD
PATCHBGN	B65D	PATCHEND	B6E8	PENDCMD	AA62	PFNADR	9D06	PGCMDMSK	0002
PGMNAME	AAB8	PHASEOFF	C080	PHASEON	C081	PHPERTRK	B7FC	PNTEXEC	A69D
PNTNXTBF	A79A	POSHNDLR	AD12	POSRD1B	AC87	POSRDR	AC93	POSTNB16	B8C2
POSWRT1B	ACBB	POSWRTR	ACC7	PPARMS	A941	PRBYTE	FDDA	PRENIB16	B800
PREPDATA	AFE4	PREPRWTS	AF4B	PROCMD	A17A	PROMPT	0033	PROTECT	00D6
PRTDEC	AE42	PRTERORR	A702	PRWTSDIR	B045	PTCHSTOP	B6FD	PTOOBIG	A6CC
PTOOLRG	A4AB	PUTDOS	B74A	PWREDUP	03F4	PWRUPBYT	00A5	RAM2WP	C080
RAMASTBL	9D78	RANGERR	B363	RANGERR2	B367	RD1BYTE	AC8A	RD2BYT	A47A
RDABYTE	ACA8	RDADR16	B944	RDADRX	B99E	RDCHAR	9EBA	RDDASEC	AFDC
RDDIRSEC	B011	RDERR	B942	RDHNDLR	AC58	RDKEY	FD0C	RDNIBL	BA96
RDNIBLBT	036C	RDNXTDA	B0B6	RDRANGE	AC96	RDTEXT	A68C	RDTSLIST	AF5E
RDVTOC	AFF7	READ16	B8DC	READBYTE	A626	READMODE	0001	REBOOT	BFC8
RECALCNT	06F8	RECDLNH	9CBD	RECFLTYP	0010	RECNUM	B5BD	RECNUMBR	B5EA
RECURNUM	9CBF	RECVAL	AA6E	REGRST	9FBA	RELSFRST	B5DC	RELSLAST	B5DE
RELSLRD	B5E0	RENHNDLR	AC3A	RESTART	B744	RETRYCNT	0578	RETURN	008D
RKEYMASK	0004	RLSALLC	B2C3	ROM2WE	C081	ROMDATA	0040	ROMSECTR	003D
ROMTEMPZ	003C	ROMTRACK	0041	RORBITMP	B2DD	RSET0	A75B	RTNCODE	B5C5
RUNADR	9D58	RUNDONE	9F78	RUNINTRC	AAB7	RWCOMMN2	A546	RWCOMMON	A526
RWDRVERR	0040	RWINITER	0008	RWNOERR	0000	RWPAGES	B793	RWPROTER	0010
RWRANGE	A3FF	RWRANGE2	A40A	RWREADER	0080	RWSETUP	A471	RWTSDRVR	B052
RWTSENT	BD00	RWTSFRMT	0004	RWTSPADR	AAC1	RWTSPPTR	B7E4	RWTSPRMS	B7C2
RWTSREAD	0001	RWTSSEEK	0000	RWTSWRIT	0002	RWVOLERR	0020	SAVFMW	AE7E
SAVREG	9ED1	SCNXIT	9F83	SECALOTR	9CC5	SECATOFF	9CAD	SECBTMAP	B5F2
SECCNT	B5EE	SECFILEN	9CC3	SECFNDZ	002D	SECFRETR	9CC7	SECFRSTS	9CB1
SECLASTS	9CB3	SECLSTRD	9CB5	SECMAP	BFA8	SECPERTS	B5DA	SECRPOST	9CB9
SECRSIZE	9CB7	SECTLEN	B5E2	SECTORZ	003F	SEEK1	BE6B	SEEKABS	B9A0
SEEKCNT	04F8	SEEKTRKZ	002A	SELBASIC	A4B1	SELBUF	AF08	SELDABUF	AF10
SELTSBUF	AF0C	SETBASIC	A5B2	SETCMDCD	B058	SETDFLTS	A0D1	SETERROR	B385
SETKBD	FE89	SETTRK	BE95	SETUPRW	AF3A	SETVID	FE93	SETWARM	BFE6
SFILETYP	0008	SFNADR	9D08	SFNAME	AA93	SKEYMASK	0010	SKIPLN	AE2F
SLNUMMSK	0008	SLOT	B5C1	SLOT16	B5F7	SLOT16Z	002B	SLOTABS	0678
SLOTFND	B7F7	SLOTSAV	05F8	SLOTVAL	AA6A	SLPMARK1	00DE	SLPMARK2	00AA
SLPMARK3	00EB	SNUM	B7ED	SNUM16	B7E9	SOFTEV	03F2	SPACE	00A0
SSAVE	AA59	STACK	0100	STATE0	0000	STATE1	0001	STATE2	0002
STATE3	0003	STATE4	0004	STATE5	0005	STATE6	0006	STKSAVE	B39B
STROBE	C08C	SUBCODE	B5BC	SYNCMARK	00FF	SYNCNT	0045	TBLTYPE	B7E8
TEMP1	AA63	TEMP2Z	0027	TEMPZ	0026	TNUM	B7EC	TRACKZ	0044
TRKFNDZ	002E	TRKNUMBR	B5FA	TSBFRRMSK	0080	TSBUFADR	9CF3	TSBUFFER	9BA6
TSCURDAT	9CAB	TSCURRTS	9CA8	TSFRSTTS	9CA6	TSLSTADR	B5C9	TSLSTOFF	000C
TSNXTDIR	B4BC	TSRECOFF	0005	TSSECOFF	0002	TSSECTOR	B4C7	TSTPEND	9E45
TSTRACK	B4C6	TSTRKOFF	0001	TXTFLTYP	0000	UNLKHNDL	ACF6	USRBUF	B7F0
VALSBGN	08FD	VALSEND	0900	VFYHNDLR	AD18	VFYPTCH	B686	VID80OFF	C00C
VIDINTRC	9EBD	VKEYMASK	0040	VOLEXPT	B7EB	VOLFND	B7F6	VOLFNDZ	002F
VOLNUMBR	B5F9	VOLUME	B5BF	VOLUMEZ	0041	VOLVAL	AA66	VTOCEND	00F5
VTOCENSZ	0023	VTOCMASK	0002	VTOCPADR	AAC3	VTOCSB	B3BB	VTOCTRK	0011

WABUFADR	9CF1	WADRVNUM	9CCD	WAFILTYP	9CCB	WAFLAGS	9CAA	WAIT	FCA8
WAIT12	BCC3	WASLTNUM	9CCC	WATRKNUM	9CCF	WAVOLNUM	9CCE	WBADR	B5C7
WBYTE	BCC4	WBYTE11	BCD4	WBYTE9	BCD5	WHCBASIC	AAB6	WNIBL	B8BB
WNIBL7	B8B9	WNIBL9	B8B8	WORKAREA	9CA6	WRADR16	BC56	WRITBYTE	A60E
WRITE16	B82A	WRNIBL	BA29	WRT1BYTE	ACBE	WRT2BYT	A3E0	WRTBYTE	ACDA
WRTDIRSC	B037	WRTHNDLR	AC70	WRTRANGE	ACCA	WRTVTOC	AFFB	XMODE	04FB
XSAVE	AA5A	XTOY	BE8E	YSAVE	AA5B	ZERO	0000	ZEROBUFR	B7D6

Symbols numerically sorted:

ZERO	0000	TXTFLTYP	0000	STATE0	0000	RWTSSEEK	0000	RWNOERR	0000
INTEGER	0000	INTACTV	0000	FMNOOPSC	0000	FMNOERR	0000	TSTRKOFF	0001
STATE1	0001	RWTSREAD	0001	READMODE	0001	NWFLMSK	0001	INTFLTYP	0001
FMRW01SC	0001	FMOPENCD	0001	FMLNAERR	0001	ALOCFRWD	0001	AKEYMASK	0001
VTOCMASK	0002	TSSECOFF	0002	STATE2	0002	RWTSWRIT	0002	PGCMDMSK	0002
FMRWNBSC	0002	FMOPTERR	0002	FMCLOSCD	0002	BKEYMASK	0002	ASFLTYP	0002
STATE3	0003	FMSUBERR	0003	FMREADCD	0003	FMPOS1SC	0003	STATE4	0004
RWTSFRMT	0004	RKEYMASK	0004	MXFLEMSK	0004	FMWRITCD	0004	FMPROTER	0004
FMPOSNSC	0004	BINFLTYP	0004	TSRECOFF	0005	STATE5	0005	FMEOFERR	0005
FMDELECD	0005	STATE6	0006	FMNOFLER	0006	FMCATACD	0006	FMVOLERR	0007
FMLOCKCD	0007	SLNUMMSK	0008	SFILETYP	0008	RWINITER	0008	LKEYMASK	0008
FMUNLKCD	0008	FMDIOERR	0008	FMRENMCD	0009	FMFULLER	0009	FMPOSICD	000A
FMLOCKER	000A	FMINITCD	000B	TSLSTOFF	000C	FMVERICD	000C	SKEYMASK	0010
RWPROTER	0010	RECFLTYP	0010	MONOMASK	0010	FLNM2MSK	0010	VTOCTRK	0011
LNSPERPG	0016	NAMESIZE	001E	RWVOLERR	0020	MONIMASK	0020	INTBASIC	0020
FLNM1MSK	0020	DKEYMASK	0020	AFILETYP	0020	VTOCENSZ	0023	LASTRACK	0023
CH	0024	TEMPZ	0026	BUFRADRZ	0026	TEMP2Z	0027	BASEZ	0028
SEEKTRKZ	002A	SLOT16Z	002B	DATAFNDZ	002C	SECFNDZ	002D	TRKFNDZ	002E
VOLFNDZ	002F	PROMPT	0033	DRIVNO	0035	CSWL	0036	KSWL	0038
ROMTEMPZ	003C	DEVCTBL	003C	ROMSECTR	003D	ODDBITSZ	003E	BUFADR2Z	003E
BOOTADRZ	003E	SECTORZ	003F	NBUFMASK	003F	VKEYMASK	0040	RWDRVERR	0040
ROMDATA	0040	MONCMASK	0040	FPOACTV	0040	FILEBUFZ	0040	DATAMASK	0040
CMDOPMSK	0040	BFILETYP	0040	VOLUMEZ	0041	ROMTRACK	0041	BUFADRZ	0042
TRACKZ	0044	OPRND	0044	SYNCNT	0045	MONTIME	0046	IOBADR	0048
INTLOMEM	004A	INTHIMEM	004C	FPBASIC	004C	NBUF2SIZ	0056	FLASHSPC	0060
ASPGMST	0067	ASVARS	0069	ASSTRS	006F	ASHIMEM	0073	ASRUN	0076
MONMASK	007F	LOCKMASK	007F	ASCIMASK	007F	TSBFRMSK	0080	RWREADER	0080
MONFLAG	0080	MKEYMASK	0080	LOCKFLAG	0080	FPAACTV	0080	FNOPMSK	0080
ASCIFLAG	0080	CTRLD	0084	LINEFEED	008A	RETURN	008D	ADRMARK3	0096
SPACE	00A0	PWRUPBYT	00A5	SLPMARK2	00AA	ODDBITS	00AA	DATMARK2	00AA
ADRMARK2	00AA	DATMARK3	00AD	ASPEND	00AF	INTSTRT	00CA	INTVEND	00CC
DATMARK1	00D5	ADRMARK1	00D5	PROTECT	00D6	ASONERR	00D8	INTRUN	00D9
SLPMARK1	00DE	LWRCASE	00E0	SLPMARK3	00EB	VTOCEND	00F5	DEFLTVOL	00FE
SYNCKMARK	00FF	NEGONE	00FF	DELETF LG	00FF	ALOCBKWD	00FF	STACK	0100
PAGESIZE	0100	INPUT	0200	NBUF2BT	0300	RDNIBLBT	036C	PAG3BGN	03D0
DOSRST	03D0	SOFTEV	03F2	PWREDUP	03F4	PAG3END	0400	DRV0TRK	0478
CURTRK	0478	SEEKCNT	04F8	DRV1TRK	04F8	XMODE	04FB	RETRYCNT	0578
SLOTSAV	05F8	SLOTABS	0678	RECALCNT	06F8	PAGE08	0800	BOOTBGN	0800
INTRLTBL	084D	BOOTEND	085D	VALSBGN	08FD	BOOTVALS	08FD	BOOTJMP	08FD
BOOTADR	08FE	BOOTPGS	08FF	VALSEND	0900	ASRAMWRM	0C3C	ASRAMRST	0CF2
ASRAMCLR	0E65	ASRAMNEW	0FD4	PAGE10	1000	ASRAMERR	1067	DOSFBADR	9AA6
DATABUFR	9AA6	TSBUFFER	9BA6	WORKAREA	9CA6	TSFRSTTS	9CA6	TSCURRTS	9CA8
WAFLAGS	9CAA	TSCURDAT	9CAB	SECATOFF	9CAD	BYCATOFF	9CAE	MAXTSECR	9CAF
SECFRSTS	9CB1	SECLASTS	9CB3	SECLSTRD	9CB5	SECRSIZE	9CB7	SECRPOST	9CB9
BYSECOFF	9CBB	RECDLNH	9CBD	RECURNUM	9CBF	BYRECOFF	9CC1	SECFILEN	9CC3
SECALOTR	9CC5	CURALOTR	9CC6	SECFRETR	9CC7	WAFILTYP	9CCB	WASLTNUM	9CCC
WADRVNUM	9CCD	WAVOLNUM	9CCE	WATRKNUM	9CCF	FILNAMBF	9CD3	WABUFADR	9CF1
TSBUFADR	9CF3	DABUFADR	9CF5	NXTFNADR	9CF7	BUFREND	9CF9	DOSTART	9D00
DBUFP	9D00	DOSKBD	9D02	DOSVID	9D04	PFNADR	9D06	SFNADR	9D08

LDRNGLEN	9D0A	LOADADR	9D0C	FMPARMS	9D0E	CSWSTADR	9D10	CMDTBL	9D1E
CMDINIT	9D1E	CMDLOAD	9D20	CMDSAVE	9D22	CMDRUN	9D24	CMDCHAIN	9D26
CMDDELET	9D28	CMDLOCK	9D2A	CMDUNLCK	9D2C	CMDCLOSE	9D2E	CMDREAD	9D30
CMDEXEC	9D32	CMDWRITE	9D34	CMDPOSTN	9D36	CMDOPEN	9D38	CMDAPND	9D3A
CMDRENAM	9D3C	CMDCAT	9D3E	CMDMON	9D40	CMDNOMON	9D42	CMDPRNUM	9D44
CMDINNUM	9D46	CMDMXFLS	9D48	CMDFP	9D4A	CMDINT	9D4C	CMDBSAVE	9D4E
CMDBLOAD	9D50	CMDBRUN	9D52	CMDVERFY	9D54	CHNADR	9D56	RUNADR	9D58
BASERR	9D5A	BASCOLD	9D5C	BASWARM	9D5E	ASFTREL	9D60	INTTBL	9D62
FPTBL	9D6C	RAMASTBL	9D78	DOSSTR	9D84	DOSWARM	9DBF	INITDOS	9DD1
FRSTIME	9DEA	TSTPEND	9E45	PAG3CODE	9E51	KBDINTRC	9E81	RDCHAR	9EBA
VIDINTRC	9EBD	SAVREG	9ED1	CSWST0	9EEB	CSWST1	9F12	CSWST1.1	9F15
CSWST2	9F23	CSWST2.1	9F27	CSWST3	9F2F	CSWST4	9F52	CSWST4.1	9F5B
CSWST5	9F61	CSWST6	9F71	RUNDONE	9F78	SCNXIT	9F83	ECHOC	9F95
ECHOO	9F99	ECHOI	9F9D	ECHO	9FA4	DOSXIT	9FB3	REGRST	9FBA
CMDCOUT	9FC5	CRLF	9FC8	PARSE	9FCD	GETFRST	A000	CMPUTIDX	A01B
CHKFRST	A092	BLKNAME	A095	BLKNAME2	A097	FNDNONAM	A0A0	SETDFLTS	A0D1
GETNXT	A0E8	PROCMD	A17A	DOCMD	A180	GNXTCHR	A193	FLSHCMDL	A1A4
CLRFMP	A1AE	GETNUM	A1B9	MUL2	A1FE	DOPRNUM	A229	DOINNUM	A22E
DOMON	A233	DONOMON	A23D	DOMXFLS	A251	DODELETE	A263	DOLOCK	A271
DOUNLOCK	A275	DOVERIFY	A27D	DORENAME	A281	DOAPND	A298	DOOPEN	A2A3
CMDHNDLR	A2A8	CMDHNDL2	A2AA	CLOSOPN	A2C8	DOCLOSE	A2EA	CLOSFREE	A2FC
CLOSEALL	A316	DOBSAVE	A331	DOBLOAD	A35D	DOBRUN	A38E	DOSAVE	A397
OPENTST	A3D5	WRT2BYT	A3E0	RWRANGE	A3FF	RWRANGE2	A40A	DOLOAD	A413
DOLOAD2	A416	RWSETUP	A471	RD2BYT	A47A	PTOOLRG	A4AB	SELBASIC	A4B1
DORUN	A4D1	DORUN2	A4DC	DOSIRUN	A4E5	DOCHAIN	A4F0	DOSARUN	A4FC
DOSARUN2	A506	DOWRITE	A510	DOREAD	A51B	RWCOMMON	A526	RWCOMMN2	A546
DOINIT	A54F	DOCAT	A56E	DOFP	A57A	DOINT	A59E	SETBASIC	A5B2
DOEXEC	A5C6	DOPSTION	A5DD	WRITBYTE	A60E	READBYTE	A626	ISBASRUN	A65E
CLSWARM	A679	EXECRD	A682	RDTEXT	A68C	PNTEXEC	A69D	FMDRVR	A6A8
FMDRVR2	A6BB	CSYNERR	A6C4	NOBUF	A6C8	PTOOBIG	A6CC	FMISMTCH	A6D0
DOERROR	A6D2	PRTERORR	A702	COPYPARM	A71A	COPYNAME	A743	COPYPTRS	A74E
RSET0	A75B	LOCBUF	A764	FRSTBUF	A792	PNTNXTBF	A79A	GFBFNAM	A7AA
ISEXCBUF	A7AF	CHKTYPE	A7C4	INITBUFS	A7D4	FRENAME	A7E5	INITPTRS	A851
DOSCMDS	A884	KWRDPRMS	A909	PPARMS	A941	PARMBITS	A94B	KWRANGE	A955
ERRMSG	A971	ERRMSG0	A971	ERRMSG1	A974	ERRMSG2	A98A	ERRMSG3	A995
ERRMSG4	A9A4	ERRMSG5	A9AF	ERRMSG6	A9BD	ERRMSG7	A9CC	ERRMSG8	A9D5
ERRMSG9	A9DE	ERRMSGA	A9E9	ERRMSGB	A9F5	ERRMSGC	AA09	ERRMSGD	AA1B
ERRMSG	AA2C	EMSOFF00	AA3F	EMSGOFFS	AA3F	EMSOFF01	AA40	EMSOFF02	AA41
EMSOFF03	AA42	EMSOFF04	AA43	EMSOFF05	AA44	EMSOFF06	AA45	EMSOFF07	AA46
EMSOFF08	AA47	EMSOFF09	AA48	EMSOFF10	AA49	EMSOFF11	AA4A	EMSOFF12	AA4B
EMSOFF13	AA4C	EMSOFF14	AA4D	EMSOFF15	AA4E	BUFADR	AA4F	CURSTAT	AA51
CSWSTATE	AA52	CSWLSAV	AA53	KSWLSAV	AA55	MAXFILES	AA57	SSAVE	AA59
XSAVE	AA5A	YSAVE	AA5B	ASAVE	AA5C	CMDLNIDX	AA5D	MONFLGS	AA5E
CMDINDX	AA5F	LOADLEN	AA60	PENDCMD	AA62	TEMP1	AA63	KYWRDIDX	AA64
KYWRDFND	AA65	VOLVAL	AA66	DRVAL	AA68	SLOTVAL	AA6A	LENVAL	AA6C
RECVAL	AA6E	BYTVAL	AA70	ADRVAL	AA72	MONVAL	AA74	FNAME	AA75
SFNAME	AA93	MXFLS	AAB1	CTLD	AAB2	EXFLG	AAB3	EXCBUF	AAB4
WHCBASIC	AAB6	RUNINTRC	AAB7	PGMNAME	AAB8	RWTSPADR	AAC1	VTCPADR	AAC3
DIRPADR	AAC5	FMFTBL	AAC9	FMRSUB	AAE5	FMWSUB	AAF1	FMEXT	AAFD
FILEMNGR	AB06	OPNHNDLR	AB22	CMNOPN	AB28	INITFMW	ABDC	CLSHNDLR	AC06
RENHNDLR	AC3A	RDHNDLR	AC58	WRTHNDLR	AC70	POSRD1B	AC87	RD1BYTE	AC8A
POSRRDR	AC93	RDRANGE	AC96	RDABYTE	ACA8	POSWRT1B	ACBB	WRT1BYTE	ACBE
POSWRTR	ACC7	WRTRANGE	ACCA	WRTBYTE	ACDA	LCKHNDLR	ACEF	UNLKHNDL	ACF6
POSHNDLR	AD12	VFYHNDLR	AD18	DELHNDLR	AD2B	FREESECT	AD89	CATHNDLR	AD98
SKIPLN	AE2F	PRTDEC	AE42	LDFMW	AE6A	SAVFMW	AE7E	INITHNDL	AE8E
SELBUF	AF08	SELTSBUF	AF0C	SELDABUF	AF10	CHKBUF	AF1D	CHKTS	AF34
SETUPRW	AF3A	PREPRWTS	AF4B	RDTSLIST	AF5E	OLDRWTS	AFB5	RDDASEC	AFDC
PREPDATA	AFF4	RDVTOC	AFF7	WRTVTOC	AFFB	RDDIRSEC	B011	WRTDIRSC	B037
PRWTSRDR	B045	RWTSRDRVR	B052	SETCMDCD	B058	RDNXTDA	B0B6	ADDATA	B134
INCREC	B15B	INCPOS	B194	MOVRANG	B1A2	DECRNG	B1B5	LCDIRENT	B1C9
COPYFNAM	B21C	NXTDIREN	B230	ALLOCSEC	B244	RLSALLC	B2C3	RORBITMP	B2DD

CALPOSN	B300	LNOTAVL	B35F	RANGERR	B363	RANGERR2	B367	ENDATA	B36F
FNOTFND	B373	DSKFULL	B377	FILELOCK	B37B	NOERROR	B37F	SETERROR	B385
DIRTS	B397	STKSAVE	B39B	DIRINDX	B39C	CNTR	B39D	ALLCFLG	B39E
FREEMASK	B3A0	DECTBL	B3A4	FTTBL	B3A7	DISKVOL	B3AF	VTOCSB	B3BB
DSKVOLND	B3BB	FRSTTS	B3BC	DOSRLS	B3BE	DSKVOL	B3C1	NUMTSENT	B3E2
NXTTOALC	B3EB	ALLCDIR	B3EC	NUMTRKS	B3EF	NUMSCTRS	B3F0	BYTPRSEC	B3F1
BITMAP	B3F3	DIRSECBF	B4BB	TSNXTDIR	B4BC	TSTRACK	B4C6	TSSECTOR	B4C7
FILTYP	B4C8	FILNAME	B4C9	FILESIZE	B4E7	FMOPCOD	B5BB	SUBCODE	B5BC
RECNUM	B5BD	VOLUME	B5BF	BYTOFFST	B5BF	DRIVE	B5C0	SLOT	B5C1
BYTRANGE	B5C1	FILETYPE	B5C2	FNADR	B5C3	DATBYTE	B5C3	DATADR	B5C3
RTNCODE	B5C5	WBADR	B5C7	TSLSTADR	B5C9	DATASADR	B5CB	FTSTS	B5D1
FTSS	B5D2	CURTSTS	B5D3	CURTSS	B5D4	FLAGS	B5D5	CURDATS	B5D6
CURDAS	B5D7	DIRSECIX	B5D8	DIRBYTIX	B5D9	SECPERTS	B5DA	RELSFRST	B5DC
RELSLAST	B5DE	RELSLRD	B5E0	SECTLEN	B5E2	FILEPOSN	B5E4	FILEBYTE	B5E6
OPNRCLN	B5E8	RECNUMBR	B5EA	BYTEOFFS	B5EC	SECCNT	B5EE	NEXTSECR	B5F0
CURTRACK	B5F1	SECBTMAP	B5F2	FTYPE	B5F6	SLOT16	B5F7	DRVNUMBR	B5F8
VOLNUMBR	B5F9	TRKNUMBR	B5FA	FMWAEND	B5FE	BOOTCODE	B600	PATCHBGN	B65D
APPFLG	B65D	APNDPTCH	B65E	APTCH2	B671	VFYPTCH	B686	APTCH3	B692
PATCHEND	B6E8	PTCHSTOP	B6FD	RESTART	B744	PUTDOS	B74A	RWPAGES	B793
CALLRWTS	B7B5	RWTSPRMS	B7C2	ZEROBUFR	B7D6	NUMPGS	B7E0	NSECSRW	B7E1
RWTSPPTR	B7E4	FRSTBOOT	B7E6	TBLTYPE	B7E8	SNUM16	B7E9	DNUM	B7EA
VOLEXPT	B7EB	TNUM	B7EC	SNUM	B7ED	DCTADR	B7EE	USRBUF	B7F0
BYTCNT	B7F2	CMDCODE	B7F4	ERRRCODE	B7F5	VOLFND	B7F6	SLOTFND	B7F7
DRVFND	B7F8	DEVTYPE	B7FB	PHPERTRK	B7FC	MOTONTIM	B7FD	PRENIB16	B800
WRITE16	B82A	WNIBL9	B8B8	WNIBL7	B8B9	WNIBL	B8BB	POSTNB16	B8C2
READ16	B8DC	RDERR	B942	RDADR16	B944	RDADR16	B99E	SEEKABS	B9A0
ISTHERE	B9EA	CHKPOS	B9EE	CHKPOS2	B9F1	MSWAIT	BA00	ONTBL	BA11
OFFTBL	BA1D	WRNIBL	BA29	HBA69	BA69	HBA76	BA76	HBA84	BA84
RDNIBL	BA96	NBUF1	BB00	NBUF2	BC00	WRADR16	BC56	WAIT12	BCC3
WBYTE	BCC4	WBYTE11	BCD4	WBYTE9	BCD5	RWTSENT	BD00	GALLDONE	BE0B
FORMDSK	BE0D	MYSEEK	BE5A	SEEK1	BE6B	XTOY	BE8E	SETTRK	BE95
DSKFORM	BEAF	DISKF2	BF0D	DELAY12	BF87	SECMAP	BFA8	INTRLEAV	BFB8
REBOOT	BFC8	AJUSTBYT	BFDC	SETWARM	BFE6	DISKFULL	BFED	MEMTOP	C000
VID80OFF	C00C	ALTCHOFF	C00E	BOOTFW	C05C	BOOTFW2	C05D	RAM2WP	C080
PHASEOFF	C080	ROM2WE	C081	PHASEON	C081	FNDADDR	C083	MOTOROFF	C088
MOTORON	C089	DRV0EN	C08A	DRV1EN	C08B	STROBE	C08C	LATCH	C08D
DATAIN	C08E	DATAOUT	C08F	FNDDATA	C0A6	ASROMWRM	D43C	ASROMRST	D4F2
ASROMCLR	D665	ASROMNEW	D7D2	ASROMERR	D865	BASCLD	E000	BASWRM	E003
INTERR	E3E3	IRUN	E836	OLDBRK	FA59	INIT	FB2F	HOME	FC58
WAIT	FCA8	RDKEY	FD0C	PRBYTE	FDDA	COUT	FDED	SETKBD	FE89
INPORT	FE8B	SETVID	FE93	OUTPORT	FE95	IORTS	FF58	MON	FF65