

!A

LLOAD INCL.L,A\$4000

LLOAD CFFA.L,A\$4000

*** End of Pass 1

LLOAD INCL.L,A\$4000

LLOAD SLOT.L,A\$4000

LLOAD ROM1.L,A\$4000

LLOAD ROM2.L,A\$4000

LLOAD ROM3.L,A\$4000

LLOAD ROM4.L,A\$4000

LLOAD ROM5.L,A\$4000

LLOAD CFFA.L,A\$4000

*** End of Pass 2

```
0800      1          ttl "CFFA Firmware Source Code, CFFA.L"
0800      2          src "CFFA.L"
0800      3      ;
0800      4      ;
0800      5      ; CFFA.L
0800      6      ;
0800      7      ;
0800      8      ; CFFA Firmware Source Code
0800      9      ;
0800     10      ; 2024 February 14
0800     11      ;
0800     12      ;
0800     13      ; DOS 4.5, Build 06
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          icl "INCL.L"
```

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

```
0800      1          ttl "CFFA Firmware Source Code, INCL.L"
0800      2      ;
0800      3      ;
0800      4      ; INCL.L
0800      5      ;
0800      6      ;
0800      7      ; Page Zero variables.
0800      8      ;
0026      9  BUFRADRZ epz $26          ; boot buffer pointer
002B     10  SLOT16Z epz $2B          ; boot slot16
002C     11  DRIVEZ epz $2C          ; IOCB drive
002D     12  SECTORZ epz $2D          ; IOCB sector
002E     13  TRACKZ epz $2E          ; IOCB track
002F     14  VOLUMEZ epz $2F          ; IOCB volume
0800     15  ;
003C     16  COMMANDZ epz $3C          ; IOCB command
003D     17  ROMSECTR epz $3D          ; boot sector
003E     18  GENPTR epz $3E          ; general data pointer
0800     19  ;
004A     20  IOBADR epz $4A          ; RWTS IOCB pointer
0800     21  ;
00EE     22  DATPTR epz $EE          ; Disk Address Table pointer
0800     23  ;
0800     24          enz
0800     25  ;
0800     26  ;
0800     27  ; Version and Build parameters.
0800     28  ;
0005     29  CFFAVRSN equ $05          ; CFFA version number
0006     30  CFFABLD equ $06          ; CFFA build number
0800     31  ;
0033     32  VRSN3.3 equ $33          ; DOS 3.3 version number
0800     33  ;
0041     34  VRSN4.1 equ $41          ; DOS 4.1 version number
0046     35  BLD4.1 equ $46          ; DOS 4.1 build number
0800     36  ;
0043     37  VRSN4.3 equ $43          ; DOS 4.3 version number
0008     38  BLD4.3 equ $08          ; DOS 4.3 build number
0800     39  ;
0045     40  VRSN4.5 equ $45          ; DOS 4.5 version number
0006     41  BLD4.5 equ $06          ; DOS 4.5 build number
0800     42  ;
0800     43  ;
0800     44  ; Standard parameters.
0800     45  ;
0000     46  ZERO equ $00          ; zero
0018     47  NAMESIZE equ $18          ; length of filename
00FF     48  NEGONE equ $FF          ; negative one
0800     49  ;
0800     50  ;
0800     51  ; ASCII parameters.
0800     52  ;
007F     53  ASCIMASK equ $7F          ; mask to remove bit 7
0080     54  ASCIFLAG equ $80          ; flag to set bit 7
0088     55  LARROW equ $88          ; left arrow
008D     56  RETURN equ $8D          ; carriage return
00A0     57  SPACE equ $A0          ; space character
0800     58  ;
0800     59  ;
0800     60  ; Delay time values for CFWAIT routine.
```

```
0800      61 ;
0004      62 WAIT100U equ $04           ; 100 usec delay
0011      63 WAIT001M equ $11         ; 1 msec delay
007C      64 WAIT040M equ $7C         ; 40 msec delay
00C5      65 WAIT100M equ $C5         ; 100 msec delay
0800      66 ;
0800      67 ;
0800      68 ; Minimum and maximum values for variables.
0800      69 ;
0004      70 MAXIMTRK equ $04           ; max boot IMAGE track
0800      71 ;
0001      72 MINIMAG equ $01           ; minimum image value
0008      73 MAXIMAG equ $08           ; maximum image value
0800      74 ;
0003      75 VOLXINC equ $03           ; volume increment, X-reg
0030      76 VOLYINC equ $30           ; volume increment, Y-reg
0800      77 ;
0020      78 MAXCFSEC equ $20           ; max CFFA sectors
0030      79 MAXCFTRK equ $30           ; max CFFA tracks
0800      80 ;
0800      81 ;
0800      82 ; Variable masks.
0800      83 ;
0007      84 SLOTMASK equ $07           ; slot number mask
000F      85 LBAMASK equ $0F           ; bits 27:24 mask
000F      86 NIBLMASK equ $0F         ; lower nibble mask
000F      87 SECMASKL equ $0F         ; lower sector mask
0010      88 SECMASKH equ $10         ; upper sector mask
003F      89 TRKMASK equ $3F           ; pseudo track 0 mask
0800      90 ;
0800      91 ;
0800      92 ; CFFA lengths and offsets.
0800      93 ;
0005      94 WAITMLEN equ $05
0014      95 IDSRLLEN equ $14
0014      96 IDSRLOFF equ $14
0800      97 ;
0300      98 VOLSIZE equ MAXCFTRK*MAXCFSEC/2
0800      99 ;
0800     100 ;
0800     101 ; Disk interface ROM offsets.
0800     102 ;
005C     103 ROMENTRY equ $5C
00FE     104 VRSNOFF equ $FE
0800     105 ;
0800     106 ;
0800     107 ; RWTS processing commands.
0800     108 ;
0800     109 ; The RDLBACMD and WRLBACMD both utilize the IOCB in order
0800     110 ; to read or write the CF card using LBA number directly.
0800     111 ; The LBA number consists of four bytes, with LBA0 the LSB
0800     112 ; and LBA3 the MSB. DVTS maps to LBA as follows: sector
0800     113 ; for LBA0, track for LBA1, volume for LBA2, and drive
0800     114 ; for LBA3. Once the user has completed processing using
0800     115 ; the IOCB to read or write the CF card using LBA number,
0800     116 ; the IOCB should be restored to a useable function, where
0800     117 ; drive = 1 and volume = track = sector = zero.
0800     118 ;
0000     119 SEEKCMD equ $00           ; seek to DVTS
0001     120 READCMD equ $01           ; read DVTS, sectors 0:1F
0002     121 WRITCMD equ $02           ; write DVTS, sectors 0:1F
```

```

0004      122  FMTCMD    equ $04          ; format DVTS, sectors 0:1F
0011      123  READCMD2 equ $11          ; read DVTS 512 bytes, 0:1F
0012      124  WRITCMD2 equ $12          ; write DVTS 512 bytes, 0:1F
0020      125  RSETCMD   equ $20          ; reset CFFA
0021      126  RDLBACMD  equ $21          ; read LBA, 512 bytes
0022      127  WRLBACMD  equ $22          ; write LBA, 512 bytes
0030      128  RDIDCMD   equ $30          ; read CFFA ID, 512 bytes
0800      129  ;
0800      130  ;
0800      131  ; RTWS processing errors.
0800      132  ;
0030      133  RWSYNERR  equ $30          ; RWS syntax error (new)
0800      134  ;
0800      135  ;
0800      136  ; Indexes for RWS IOCB variables.
0800      137  ;
0001      138  SLOTNDX   equ $01          ; slot16
0002      139  DRVNDX    equ $02          ; drive
0003      140  VOLNDX     equ $03          ; volume, or LBA1
0004      141  TRKNDX     equ $04          ; track, or LBA2
0005      142  SECNDX     equ $05          ; sector, or LBA3
0008      143  BUFRNDX    equ $08          ; buffer address (L/H)
000A      144  PHASNDX    equ $0A          ; half-phases per track
000B      145  XFERNDX    equ $0B          ; transfer byte count
000C      146  CMDNDX     equ $0C          ; command
000D      147  ERRNDX     equ $0D          ; return status
000E      148  LVOLNDX    equ $0E          ; return last volume
000F      149  LSLTNDX    equ $0F          ; return last slot
0010      150  LDRVNDX    equ $10          ; return last drive
0800      151  ;
0800      152  ;
0800      153  ; CFFA processing commands.
0800      154  ;
0020      155  ATA_READ   equ $20          ; read 512 bytes from one LBA
0030      156  ATA_WRIT   equ $30          ; write 512 bytes to one LBA
00EC      157  ATA_ID     equ $EC          ; read identity drive
0800      158  ;
0800      159  ;
0800      160  ; CFFA processing errors.
0800      161  ;
0000      162  NOERROR    equ $00          ; no IOCB processing error
0800      163  ;
0001      164  BUSYERR    equ $01          ; CFBUSY error
0002      165  RDYERR     equ $02          ; CFREADY error
0003      166  STATERR    equ $03          ; CFSTATUS error
0800      167  ;
0004      168  LBAERR     equ $04          ; LOADLBA error
0008      169  CMDERR     equ $08          ; LOADCMD error
000C      170  CHKERR     equ $0C          ; CFCHECK error
0800      171  ;
0080      172  RSTERR     equ $80          ; CFRESET error
0090      173  RD1ERR     equ $90          ; RD1PAGE error
00A0      174  WR1ERR     equ $A0          ; WR1PAGE error
00B0      175  FMTERR     equ $B0          ; FMTVOL error
00C0      176  IDERR      equ $C0          ; READID error
00D0      177  RD2ERR     equ $D0          ; RD2PAGE error
00E0      178  WR2ERR     equ $E0          ; WR2PAGE error
00F0      179  UNKERR     equ $F0          ; unknown command error
0800      180  ;
0800      181  ;
0800      182  ; Signature bytes.

```

```
0800      183      ;
00C4      184      SIGVAL1    equ  $C4
00B8      185      SIGVAL2    equ  $B8
0090      186      SIGVAL3    equ  $90
00ED      187      SIGVAL4    equ  $ED
0800      188      ;
0800      189      ;
0800      190      ; System architecture.
0800      191      ;
0100      192      PAGESIZE    equ  $100          ; size of 6502 page
0100      193      STACK      equ  $100          ; 6502 stack
0800      194      ;
0110      195      RSTRTSAV    equ  $110          ; RESTART save address
0800      196      ;
0112      197      MESSAGE    equ  $112          ; message number
0113      198      VERSION    equ  $113          ; DOS version number
0800      199      ;
0114      200      TEMPERR     equ  $114          ; temporary error status
0115      201      LCRAM       equ  $115          ; state of LC RAM if enabled
0800      202      ;
0116      203      STKCODE     equ  $116          ; exit routines
0800      204      ;
0800      205      ;
0800      206      ; DOS intercepts and addresses.
0800      207      ;
03D0      208      DOSWARM     equ  $3D0          ; DOS warm start entry
03D3      209      DOSCOLD     equ  $3D3          ; DOS cold start entry
03D6      210      CALLFM      equ  $3D6          ; DOS file manager entry
03D9      211      RWTS        equ  $3D9          ; RWTS manager entry
03DC      212      GETFMCB     equ  $3DC          ; DOS FM IO block pointer
03E1      213      RDCLKVSN    equ  $3E1          ; copy CLK/VSN to #Y/A bufr
03E3      214      GETIOCB     equ  $3E3          ; RWTS IO block pointer
03E8      215      PRTERADR     equ  $3E8          ; print error mesg # in X-reg
03EA      216      HOOKDOS     equ  $3EA          ; DOS reconnect entry
03ED      217      XFERADR     equ  $3ED          ; for DOXFER routine
03EF      218      AUTOBRK     equ  $3EF          ; break handler entry
03F2      219      AUTORSET    equ  $3F2          ; reset handler entry
03F4      220      PWRSTATE    equ  $3F4          ; power up byte
03F5      221      USRAHAND     equ  $3F5          ; ampersand handler entry
03F8      222      USRYHAND     equ  $3F8          ; ctrl-Y handler address
03FB      223      NMASKIRQ     equ  $3FB          ; nonmaskable IRQ address
03FE      224      MASKIRQ     equ  $3FE          ; maskable IRQ address
0800      225      ;
04FB      226      XMODE       equ  $4FB          ; video firmware mode
0800      227      ;
08FF      228      BOOTPGS     equ  $8FF          ; DOS boot page parameter
0800      229      ;
0800      230      ;
0800      231      ; Public slot variables.
0800      232      ;
0478      233      CFSLOT      equ  $478          ; CFFA slot number
04F8      234      CFSLOT16    equ  $4F8          ; slot number * 16
0578      235      LBA0        equ  $578          ; LBA 07:00 (sector)
05F8      236      LBA1        equ  $5F8          ; LBA 15:08 (track)
0678      237      LBA2        equ  $678          ; LBA 23:16 (volume)
06F8      238      LBA3        equ  $6F8          ; LBA 27:24 (drive)
0778      239      PAGE        equ  $778          ; 0 = page 1, >0 = page 2
07F8      240      CFPAGECX    equ  $7F8          ; slot memory page (MSLOT)
0800      241      ;
0800      242      ;
0800      243      ; Private slot variables using CFSLOT for index.
```

```
0800      244 ;
0800      245 ; DOS 3.3 initializes 0x478,X & 0x4F8,X (DRV0TRK, DRV1TRK)
0800      246 ; and not 0x678,X & 0x6F8,X to zero before calling RWPAGES
0800      247 ; to load into memory the rest of DOS 3.3. Therefore, the
0800      248 ; variables ERRSTAT and CFERROR are still safe to use.
0800      249 ;
0800      250 ; DOS 4.X initializes 0x478,X, 0x4F8,X, 0x678,X and 0x6F8,X
0800      251 ; (DRV0PHAS, DRV1PHAS) to zero but does not use SAVEADRL or
0800      252 ; SAVEADRH, so DRIVE, VOLUME, DOSVRSN, and DOSNBUF1 are all
0800      253 ; protected.
0800      254 ;
0478      255 ERRSTAT equ $478 ; error status value
04F8      256 CFERROR equ $4F8 ; last CF error
0578      257 DRIVE equ $578 ; drive
05F8      258 VOLUME equ $5F8 ; volume/DOS image
0678      259 SAVEADRL equ $678 ; save low address
06F8      260 SAVEADRH equ $6F8 ; save high address
0778      261 DOSVRSN equ $778 ; DOS version now booted
07F8      262 DOSNBUF1 equ $7F8 ; DOS nibble buffer 1 page
0800      263 ;
0800      264 ;
0800      265 ; DOS 3.3 addresses.
0800      266 ;
9E41      267 FTMOD equ $9E41 ; DOS 3.3 FRSTIME mod
A0D9      268 SDFMOD equ $A0D9 ; DOS 3.3 SETDFLTS mod
A955      269 KWRANGE equ $A955 ; DOS 3.3 DWRANGE range table
AA65      270 KYWRDFND equ $AA65 ; DOS 3.3 keyword parameter
AA66      271 VOLVAL equ $AA66 ; DOS 3.3 volume parameter
AD9D      272 CATHMOD1 equ $AD9D ; DOS 3.3 CATHNDLR mod 1
AE16      273 CATHMOD2 equ $AE16 ; DOS 3.3 CATHNDLR mod 2
B202      274 LCDMOD equ $B202 ; DOS 3.3 LCDIRENT mod
B5F9      275 VOLNUMBR equ $B5F9 ; DOS 3.3 File Mngr parameter
B706      276 SETDRIVE equ $B706 ; DOS 3.3 Boot Stage 2
B744      277 RESTART equ $B744 ; DOS 3.3 RESTART routine
B7B5      278 CALLRWTS equ $B7B5 ; DOS 3.3 CALLRWTS routine
B7EB      279 VOLEXPT equ $B7EB ; DOS 3.3 RWTS IOCB VOLEXPT
BB00      280 NBUF1BUF equ $BB00 ; DOS 3.3 nibble bufr address
0800      281 ;
0800      282 ;
0800      283 ; DOS 4.1 routine addresses
0800      284 ;
BED9      285 INITADR equ $BED9 ; address found in INITDOS
BFF8      286 INITDOS equ $BFF8 ; location containing INITADR
BFFB      287 DISKTBL equ $BFFB ; Disk Address Table offset
0800      288 ;
0800      289 ;
0800      290 ; DOS 4.3/5 routine addresses
0800      291 ;
BFF0      292 BLDVRSN equ $BFF0 ; DOS version number
BFF1      293 BLDNMBR equ $BFF1 ; DOS build number
0800      294 ;
BFF2      295 MNGDISK equ $BFF2 ; manage Disk Table routine
0800      296 ;
BFFC      297 BCFGNDX equ $BFFC ; BOOTCFG Table offset
BFFD      298 NBUF1ADR equ $BFFD ; NBUF1 MSB page
0800      299 ;
0800      300 ;
0800      301 ; Memory page addresses.
0800      302 ;
0800      303 PAGE08 equ $0800 ; memory address of page 0x08
0900      304 PAGE09 equ $0900 ; memory address of page 0x09
```

```
BF00      305 PAGEBF      equ $BF00      ; memory address of page 0xBF
C000      306 PAGEC0      equ $C000      ; memory address of page 0xC0
C800      307 PAGEC8      equ $C800      ; memory address of page 0xC8
D000      308 PAGED0      equ $D000      ; memory address of page 0xD0
0800      309 ;
0800      310 ;
0800      311 ; Apple //e configuration soft switches.
0800      312 ;
C00C      313 VID800FF equ $C00C      ; disable 80 column video
C00E      314 ALTCHOFF equ $C00E      ; enable normal character set
C012      315 RDLCRAM  equ $C012      ; state if LC RAM is enabled
0800      316 ;
0800      317 ;
0800      318 ; CFFA registers using CFSLOT16 as a slot index.
0800      319 ;
C080      320 ATADATAH equ $C080      ; read/write high data reg
C081      321 SETCSMSK equ $C081      ; ignore pre-fetch register
C082      322 CLRCSMSK equ $C082      ; honor pre-fetch register
C086      323 ATADEVCT equ $C086      ; CFFA device register
C086      324 ATASTAT2 equ $C086      ; alternate status register
C088      325 ATADATAL equ $C088      ; read/write low data register
C089      326 ATAERROR equ $C089      ; error register
C08A      327 ATASECCT equ $C08A      ; sector count register
C08B      328 ATASECTR equ $C08B      ; LBA 07:00 register
C08C      329 ATACYLNL equ $C08C      ; LBA 15:08 register
C08D      330 ATACYLNH equ $C08D      ; LBA 23:16 register
C08E      331 ATAHEAD  equ $C08E      ; LBA 27:24; LBA/CHS select
C08F      332 ATACMD   equ $C08F      ; command register
C08F      333 ATASTAT  equ $C08F      ; status register
0800      334 ;
0800      335 ;
0800      336 ; CFFA partition offsets used to calculate LBA.
0800      337 ;
0018      338 IMAGSIZE equ $0018      ; size of DOS image
0800      339 ;
0010      340 OFFSET0   equ $0010      ; DOS images
0100      341 OFFSET1   equ $0100      ; volume partition
0800      342 ;
0800      343 ;
0800      344 ; Memory configuration soft switches.
0800      345 ;
C082      346 ROM2WP     equ $C082      ; enable ROM, wrt protect RAM
C083      347 RAM2WE     equ $C083      ; write enable RAM, bank 2
C08B      348 RAM1WE     equ $C08B      ; write enable RAM, bank 1
0800      349 ;
0800      350 ;
0800      351 ; ROM control address.
0800      352 ;
CFFF      353 CLRROM     equ $CFFF      ; deselect ROM memory
0800      354 ;
0800      355 ;
0800      356 ; Addresses for standard ROM routines.
0800      357 ;
FB2F      358 INIT       equ $FB2F      ; terminal init
FC58      359 HOME       equ $FC58      ; clear screen
FD8E      360 CROUT      equ $FD8E      ; print return
FDED      361 COUT       equ $FDED      ; print character
FE84      362 SETNORM    equ $FE84      ; set normal text
FE89      363 SETVID     equ $FE89      ; set video
FE93      364 SETKBD     equ $FE93      ; set keyboard
FF69      365 MONITOR    equ $FF69      ; enter monitor
```



```
0800          366 ;  
0800          367 ;  
0800          368      icl "SLOT.L"
```

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

```

0800          1          ttl "CFFA Firmware Source Code, SLOT.L"
0800          2          ;
0800          3          ;
0800          4          ; SLOT.L
0800          5          ;
0800          6          ;
C000          7          org PAGEC0
C000          8          ;
C000          9          obj PAGE09
C000         10          usr
C000         11          ;
C000         12          ;
C000         13          ; Signature bytes for CFFA card firmware. Image value can
C000         14          ; range from 1 to 8.
C000         15          ;
C000 69 20     16  CFBOOT   adc #$20
C002 A0 00     17          ldy #$00
C004 A2 03     18          ldx #$03
C006 86 3C     19          stx COMMANDZ
C008          20          ;
C008 2C FF CF  21          bit CLRROM          ; deselect ROM memory
C00B          22          ;
C00B AD 3C C8  23          lda DFLTBOOT        ; get default image value
C00E D0 10     24          bne USBOOT          ; always taken
C010          25          ;
C010          26          ;
C010          27          ; Connect CFFA to DOS.
C010          28          ;
C010 2C FF CF  29  ROMHOOK bit CLRROM          ; deselect ROM memory
C013          30          ;
C013 A0 00     31          ldy #ZERO          ; select connect to DOS flag
C015 F0 0D     32          beq TOGGLE          ; always taken
C017          33          ;
C017          34          dfs 1,NEGONE
C018          35          ;
C018          36          ;
C018          37          ; Disconnect CFFA from DOS.
C018          38          ;
C018 2C FF CF  39  ROMUHOOK bit CLRROM          ; deselect ROM memory
C01B          40          ;
C01B A0 FF     41          ldy #NEGONE          ; select disconnect from DOS
C01D D0 05     42          bne TOGGLE          ; always taken
C01F          43          ;
C01F          44          dfs 1,NEGONE
C020          45          ;
C020          46          ;
C020          47          ; User Boot entry. A-reg contains boot image value.
C020          48          ;
C020 A2 00     49  USBOOT   ldx #ZERO          ; set value for drive number
C022 F0 0C     50          beq VOLBOOT        ; always taken
C024          51          ;
C024          52          ;
C024 20 41 C8  53  TOGGLE   jsr GETSLOT        ; get slot number
C027          54          ;
C027 4C 14 CB  55          jmp DOTOGGLE        ; toggle CFFA hook into DOS
C02A          56          ;
C02A          57          ;
C02A 20 41 C8  58  VOLBOOT2 jsr GETSLOT        ; get slot number
C02D          59          ;
C02D 4C D0 C8  60          jmp DOBOOT          ; boot selected DOS

```

```

C030      61 ;
C030      62 ;
C030      63 ; Volume boot entry.  A-reg contains volume number, X-reg
C030      64 ; contains drive number, drive > 0 for valid volume boot.
C030      65 ;
C030 78    66 VOLBOOT sei                ; disable interrupts
C031      67 ;
C031 2C FF CF 68                bit CLRROM        ; deselect ROM memory
C034      69 ;
C034 86 2C    70                stx DRIVEZ        ; save DOS drive
C036 85 2F    71                sta VOLUMEZ       ; save DOS volume
C038      72 ;
C038 18      73                clc                ; clear C-flag
C039 90 EF    74                bcc VOLBOOT2      ; always taken
C03B      75 ;
C03B      76 ;
C03B      77 ; DOS 3.X RWTS entry.
C03B      78 ;
C03B      79 ; On entry A-reg and Y-reg point to the IOCB.
C03B      80 ;
C03B 2C FF CF 81 DISKRWTS bit CLRROM        ; deselect ROM memory
C03E      82 ;
C03E EA      83                nop
C03F      84 ;
C03F 20 3D C8 85                jsr GETSLOT2      ; get slot number
C042      86 ;
C042 A0 01    87                ldy #SLOTNDX      ; get slot index
C044      88 ;
C044 D1 4A    89                cmp (IOBADR),Y    ; compare to CFFA slot
C046 F0 0A    90                beq CFRWTS2       ; branch if the same
C048      91 ;
C048 4C 83 C8 92                jmp ENTRDOS      ; enter DOS 3.3 RWTS
C04B      93 ;
C04B      94 ;
C04B      95 ; CFFA RWTS entry.
C04B      96 ;
C04B      97 ; On entry A-reg and Y-reg point to the IOCB.
C04B      98 ;
C04B      99 ; DOS 4.X sets the X-reg to SLOT*16, and IOBADR and
C04B     100 ; BUFADR2Z are initialized.
C04B     101 ;
C04B 2C FF CF 102 CFRWTS bit CLRROM        ; deselect ROM memory
C04E     103 ;
C04E EA      104                nop
C04F     105 ;
C04F 20 3D C8 106                jsr GETSLOT2      ; get slot number
C052     107 ;
C052 A0 0F    108 CFRWTS2 ldy #LSLTNDX      ; get found slot index
C054     109 ;
C054 91 4A    110                sta (IOBADR),Y    ; save found slot
C056     111 ;
C056 18      112                clc                ; clear C-flag
C057 90 0B    113                bcc CFRWTS3      ; always taken
C059     114 ;
C059     115 ;
C059     116                dfs ROMENTRY-*)&NEGONE,ZERO
C05C     117 ;
C05C     118 ;
C05C 78      119 ROMBOOT sei                ; disable interrupts
C05D     120 ;
C05D 2C FF CF 121                bit CLRROM        ; deselect ROM memory

```

```

C060      122 ;
C060 EA    123      nop
C061      124 ;
C061 4C 9E C9 125      jmp BOOTVOL      ; continue boot stage 2
C064      126 ;
C064      127 ;
C064      128 ; Let CFFA handle this call.
C064      129 ;
C064      130 ; A) Process using IOCB management in DVTS mode:
C064      131 ;
C064      132 ;     1) For booting an image, drive = 0
C064      133 ;         volume = 1:8
C064      134 ;         track  = 0:3
C064      135 ;         sector = 0:15
C064      136 ;
C064      137 ;     2) For processing a volume, drive > 0
C064      138 ;         volume = 0:255
C064      139 ;         track  = 0:47
C064      140 ;         sector = 0:31
C064      141 ;
C064      142 ; B) Process using IOCB management in LBA mode:
C064      143 ;
C064      144 ;     LBA3 = (27:24) drive
C064      145 ;     LBA2 = (23:16) volume
C064      146 ;     LBA1 = (15:08) track
C064      147 ;     LBA0 = (07:00) sector
C064      148 ;
C064      149 ;     When LBA mode processing is complete, the IOCB should
C064      150 ;     be reconfigured as follows:
C064      151 ;
C064      152 ;         drive  = 1
C064      153 ;         volume = 0
C064      154 ;         track  = 0
C064      155 ;         sector = 0
C064      156 ;
C064      157 ; C) Valid IOCB commands for CF data processing:
C064      158 ;
C064      159 ;     CMD  Description
C064      160 ;     ---  -----
C064      161 ;     00  DVTS, seek
C064      162 ;     01  DVTS, read 256 bytes
C064      163 ;     02  DVTS, write 256 bytes
C064      164 ;     04  DVTS, format volume
C064      165 ;     11  DVTS, read 512 bytes
C064      166 ;     12  DVTS, write 512 bytes
C064      167 ;     20  CF, reset CF
C064      168 ;     *21  LBA, read 512 bytes
C064      169 ;     *22  LBA, write 512 bytes
C064      170 ;     30  CF, read ID 512 bytes
C064      171 ;
C064      172 ;     * See paragraph B) above once processing is complete.
C064      173 ;
C064      174 ;
C064      175 ; General entry used by all versions of DOS.
C064      176 ;
C064 A0 02    177 CFRWTS3  ldy #DRVNDX      ; get drive index
C066      178 ;
C066 B1 4A    179      lda (IOBADR),Y      ; get IOCB DNUM
C068 85 2C    180      sta DRIVEZ        ; save drive
C06A 8D F8 06 181      sta LBA3          ; save LBA3
C06D      182 ;

```

```

C06D A0 10      183      ldy #LDRVNDX      ; get found drive index
C06F           184      ;
C06F 91 4A      185      sta (IOBADR),Y      ; save found drive
C071           186      ;
C071 A0 03      187      ldy #VOLNDX      ; get volume index
C073           188      ;
C073 B1 4A      189      lda (IOBADR),Y      ; get IOCB VOLEXPT
C075 85 2F      190      sta VOLUMEZ      ; save volume
C077 8D 78 06    191      sta LBA2      ; save LBA2
C07A           192      ;
C07A A0 0E      193      ldy #LVOLNDX      ; get found volume index
C07C           194      ;
C07C 91 4A      195      sta (IOBADR),Y      ; save found volume
C07E           196      ;
C07E A0 04      197      ldy #TRKNDX      ; get track index
C080           198      ;
C080 B1 4A      199      lda (IOBADR),Y      ; get IOCB TNUM
C082 8D F8 05    200      sta LBA1      ; save LBA1
C085           201      ;
C085 29 3F      202      and #TRKMASK      ; remove pseudo track 0
C087 85 2E      203      sta TRACKZ      ; save track
C089           204      ;
C089 C8          205      iny      ; set sector index
C08A           206      ;
C08A B1 4A      207      lda (IOBADR),Y      ; get IOCB SNUM
C08C 85 3D      208      sta ROMSECTR      ; save sector
C08E 8D 78 05    209      sta LBA0      ; save LBA0
C091           210      ;
C091 29 0F      211      and #SECMASKL      ; mask lower sector
C093 85 2D      212      sta SECTORZ      ; save lower sector
C095           213      ;
C095 A0 08      214      ldy #BUFRNDX      ; get buffer index
C097           215      ;
C097 B1 4A      216      lda (IOBADR),Y      ; get IOCB USRBUF
C099 85 26      217      sta BUFRADRZ      ; save lower address
C09B           218      ;
C09B C8          219      iny      ; set buffer index ++
C09C           220      ;
C09C B1 4A      221      lda (IOBADR),Y      ; get IOCB USRBUF+1
C09E 85 27      222      sta BUFRADRZ+1      ; save upper address
C0A0           223      ;
C0A0 A0 0C      224      ldy #CMDNDX      ; get command index
C0A2           225      ;
C0A2 B1 4A      226      lda (IOBADR),Y      ; get IOCB CMDCODE
C0A4 85 3C      227      sta COMMANDZ      ; save command
C0A6           228      ;
C0A6 C9 20      229      cmp #RSETCMD      ; test for CF commands
C0A8 90 08      230      bcc >1      ; process DVTS commands
C0AA           231      ;
C0AA F0 2F      232      beq >4      ; process Reset command
C0AC           233      ;
C0AC C9 30      234      cmp #RDIDCMD      ; test for ID command
C0AE F0 2B      235      beq >4      ; process ID command
C0B0           236      ;
C0B0 D0 24      237      bne >3      ; process the LBA commands
C0B2           238      ;
C0B2 20 58 C9    239      ^1      jsr CHKDRVZ      ; check if DRIVEZ is zero
C0B5 F0 24      240      beq >4      ; branch if booting an image
C0B7           241      ;
C0B7 CD 3A C8    242      cmp CFMAXDRV      ; range check drive value
C0BA 90 09      243      bcc >2      ; branch if less

```

```

C0BC          244 ;
C0BC D0 25    245      bne >5          ; error if branch
C0BE          246 ;
C0BE A5 2F    247      lda VOLUMEZ      ; range check remaining
C0C0 CD 3B C8 248      cmp CFREMOVOL    ; volume value
C0C3 B0 1E    249      bcs >5          ; error if branch
C0C5          250 ;
C0C5 C6 2C    251      ^2      dec DRIVEZ      ; align to 0:CFMAXDRV-1
C0C7          252 ;
C0C7 A5 2E    253      lda TRACKZ       ; recall track
C0C9 C9 30    254      cmp #MAXCFTRK    ; range check track value
C0CB B0 16    255      bcs >5          ; error if branch
C0CD          256 ;
C0CD A5 3D    257      lda ROMSECTR     ; recall unmasked sector
C0CF C9 20    258      cmp #MAXCFSEC    ; range check sector value
C0D1 B0 10    259      bcs >5          ; error if branch
C0D3          260 ;
C0D3 20 DE CB 261      jsr DVTS2LBA      ; convert DVTS to LBA and PAGE
C0D6          262 ;                      with ROMSECTR in A-reg
C0D6          263 ;
C0D6 20 30 CC 264      ^3      jsr CHKCF LBA      ; range check the LBA
C0D9 B0 08    265      bcs >5          ; error if branch
C0DB          266 ;
C0DB 20 39 CD 267      ^4      jsr DOFCMD      ; do the CFFA I/O
C0DE B0 05    268      bcs >6          ; error if branch
C0E0          269 ;
C0E0 A9 00    270      lda #NOERROR     ; no processing error
C0E2          271 ;
C0E2 2C 00 00 272      bit *-*          ; skip over error number
C0E5          273      dfs !-2
C0E3          274 ;
C0E3 A9 30    275      ^5      lda #RWSYNERR    ; get RWTS syntax error
C0E5          276 ;
C0E5 A0 0D    277      ^6      ldy #ERRNDX      ; get return status index
C0E7          278 ;
C0E7 91 4A    279      sta (IOBADR),Y      ; save IOCB ERRCODE
C0E9          280 ;
C0E9 AE 78 04 281      ldx CFSLOT          ; get CF slot index
C0EC          282 ;
C0EC 9D F8 04 283      sta CFERROR,X      ; save CF error
C0EF          284 ;
C0EF 08       285      php              ; save processor status
C0F0          286 ;
C0F0 4C 1D 01 287      jmp EXIT3          ; exit CFFA
C0F3          288 ;
C0F3          289 ;
C0F3 2C FF CF 290      MODOS3      bit CLRROM
C0F6          291 ;
C0F6 EA       292      nop
C0F7          293 ;
C0F7 20 41 C8 294      jsr GETSLOT
C0FA          295 ;
C0FA 4C 9F CB 296      jmp DOMODOS3
C0FD          297 ;
C0FD          298 ;
C0FD          299      dfs VRSNOFF-*&NEGONE,ZERO
C0FE          300 ;
C0FE          301 ;
C0FE 05       302      byt CFFAVRSN      ; version number
C0FF 06       303      byt CFFABLD      ; build number
C100          304 ;

```

```
C100          305 ;
C100          306 ; Copyright (c) 2023 April 04
C100          307 ; by
C100          308 ; Walland Philip Vrbancic Jr
C100          309 ;
C100          310 ; All Rights Reserved
C100          311 ;
C100          312 ;
```

```
BSAVE CFFA_SLOT_BUILD56,A$0900,B,L$0100
```

```
C100          313          usr CFFA_SLOT_BUILD56
C100          314 ;
C100          315 ;
C100          316          icl "ROM1.L"
```

```
LLOAD ROM1.L,A$4000
```

```

C100          1          ttl "CFFA Firmware Source Code, ROM1.L"
C100          2          ;
C100          3          ;
C100          4          ; ROM1.L
C100          5          ;
C100          6          ;
C800          7          org PAGEC8
C800          8          obj PAGE09
C800          9          usr
C800         10          ;
C800         11          ;
C800         12          ; Firmware Data. Data established at install time and
C800         13          ; updated by the CFFA Volume Manager.
C800         14          ;
C800         15          ; An error message is generated at boot time if the CF
C800         16          ; card does not match the serial number CFSRLBUF.
C800         17          ;
C800 C4       18          ROMSIG    byt SIGVAL1          ; signature bytes for ROM code
C801 B8       19          byt SIGVAL2
C802 90       20          byt SIGVAL3
C803 ED       21          byt SIGVAL4
C804         22          ;
C804         23          CFSRLBUF  dfs IDSRLLEN,ZERO    ; CF card serial number
C818         24          ;
C818         25          CFNAME    dfs 24,ZERO          ; CF name at install time
C830         26          ;
C830         27          CFDATE    dfs 6,ZERO           ; install date and time
C836         28          ;
C836         29          CFLBANUM  dfs 4,ZERO           ; last valid LBA on CF card
C83A         30          ;
C83A         31          CFMAXDRV  dfs 1,ZERO           ; maximum number of drives
C83B         32          CFREMVOL  dfs 1,ZERO           ; remaining volumes
C83C         33          ;
C83C         34          DFLTBOOT  dfs 1,6              ; default boot image, 1:8
C83D         35          ;
C83D         36          ;
C83D         37          ; Copy EXIT code to the stack and extract slot page from
C83D         38          ; the stack. Exit with SLOT*16 in A-reg.
C83D         39          ;
C83D 84 4A    40          GETSLOT2  sty IOBADR           ; save low address of IOCB
C83F 85 4B    41          sta IOBADR+1         ; save high address of IOCB
C841         42          ;
C841 A2 0B    43          GETSLOT   ldx #EXITLEN-1      ; get exit code size
C843         44          ;
C843 BD 60 C8 45          ^1        lda EXIT,X         ; get exit code
C846 9D 16 01 46          sta STKCODE,X       ; save to the stack
C849         47          ;
C849 CA      48          dex
C84A 10 F7    49          bpl <1
C84C         50          ;
C84C BA      51          tsx                  ; get the stack pointer
C84D         52          ;
C84D BD 02 01 53          lda STACK+2,X       ; get return high address
C850 8D F8 07 54          sta CFPAGECX       ; save CF slot page
C853         55          ;
C853 29 07    56          and #SLOTMASK       ; get physical slot number
C855 8D 78 04 57          sta CFSLOT         ; save CF slot number/index
C858         58          ;
C858 0A      59          asl                  ; multiply by 16
C859 0A      60          asl

```



```

C85A 0A          61          asl
C85B 0A          62          asl
C85C             63          ;
C85C 8D F8 04    64          sta CFSLOT16          ; save SLOT*16 number
C85F             65          ;
C85F 60          66          rts                  ; return to caller
C860             67          ;
C860             68          ;
C860             69          ; Exit routines that are copied to the stack.
C860             70          ;
C860             71          EXIT:
C860             72          phs STKCODE
0116             73          ;
0116             74          EXITBGN:
0116             75          ;
0116             76          ;
0116             77          ; Code to exit RWTS, enable interrupts, and jump to an
0116             78          ; address in EXITADR.
0116             79          ;
0116 58          80          EXIT1      cli          ; enable interrupts
0117             81          ;
0117             82          ;
0117             83          ; Code to exit the CFFA and jump to a DOS 3.X address.
0117             84          ;
0117 2C FF CF    85          EXIT2      bit CLRROM          ; deselect ROM memory
011A             86          ;
011A 4C 00 00    87          EXITADR   jmp *-*          ; jump to address
011D             88          ;
011D             89          ;
011D             90          ; Code to exit the CFFA and return to the caller.
011D             91          ;
011D 2C FF CF    92          EXIT3      bit CLRROM          ; deselect ROM memory
0120             93          ;
0120 28          94          plp                  ; recall processor status
0121             95          ;
0121 60          96          rts                  ; return to caller
0122             97          ;
0122             98          ;
000C             99          EXITLEN   equ *-EXITBGN
0122            100          ;
0122            101          phs EXIT+EXITLEN
C86C             102          ;
C86C             103          ;
C86C 20 99 C8    104          EXITERR   jsr PRTMESGS          ; print message
C86F             105          ;
C86F A9 A0       106          lda #SPACE          ; initialize A-reg
C871             107          ;
C871 A2 69       108          ldx #MONITOR          ; get MONITOR address
C873 A0 FF       109          ldy /MONITOR
C875             110          ;
C875             111          ;
C875             112          ; Deselect ROM memory and exit RWTS.
C875             113          ;
C875 8E 1B 01    114          EXITRWTS  stx EXITADR+1          ; save address
C878 8C 1C 01    115          sty EXITADR+2
C87B             116          ;
C87B AE F8 04    117          ldx CFSLOT16          ; get SLOT*16
C87E 86 2B       118          stx SLOT16Z          ; restore zpage value
C880             119          ;
C880 4C 16 01    120          jmp EXIT1          ; exit CFFA
C883             121          ;

```

```

C883      122 ;
C883      123 ; Deselect ROM memory and jump to a DOS 3.X address.
C883      124 ;
C883 AE 78 04 125 ENTRDOS ldx CFSLOT ; get CF slot index
C886      126 ;
C886 BC 78 06 127 ldy SAVEADRL,X ; get saved LSB address
C889 BD F8 06 128 lda SAVEADRH,X ; get saved MSB address
C88C      129 ;
C88C 8C 1B 01 130 sty EXITADR+1 ; save address
C88F 8D 1C 01 131 sta EXITADR+2
C892      132 ;
C892 A4 4A 133 ldy IOBADR ; get IOCB LSB pointer
C894 A5 4B 134 lda IOBADR+1 ; get IOCB MSB pointer
C896      135 ;
C896 4C 17 01 136 jmp EXIT2 ; exit CFFA
C899      137 ;
C899      138 ;
C899      139 ; Print the selected 3-byte text message string. Fall into
C899      140 ; PRTMSG. Enable ROM in order to use COUT.
C899      141 ;
C899 8D 82 C0 142 PRTMSGs sta ROM2WP ; enable ROM
C89C      143 ;
C89C A0 01 144 ldy #MSG1B-MESGS ; print two returns
C89E 20 BB C8 145 jsr PRTMSG
C8A1      146 ;
C8A1 BC 9F CF 147 ldy TEXTS,X ; print first string
C8A4 20 BB C8 148 jsr PRTMSG
C8A7      149 ;
C8A7 A9 20 150 lda #SPACE&ASCIMASK ; print a leading space
C8A9      151 ;
C8A9 BC A0 CF 152 ldy TEXTS+1,X ; print second string
C8AC 20 BF C8 153 jsr PRTMSG1 ; print a leading space
C8AF      154 ;
C8AF A9 20 155 lda #SPACE&ASCIMASK ; print a leading space
C8B1      156 ;
C8B1 BC A1 CF 157 ldy TEXTS+2,X ; get third string index
C8B4 F0 03 158 beq >1 ; done if zero, so branch
C8B6      159 ;
C8B6 20 BF C8 160 jsr PRTMSG1 ; print third string
C8B9      161 ;
C8B9 A0 00 162 ^1 ldy #MSG1A-MESGS ; print period and 2 returns
C8BB      163 ;
C8BB      164 ;
C8BB B9 33 CF 165 PRTMSG lda MSGS,Y ; get string data
C8BE      166 ;
C8BE C8 167 iny
C8BF      168 ;
C8BF 48 169 PRTMSG1 pha ; save on stack
C8C0      170 ;
C8C0 09 80 171 ora #ASCIFLAG ; set bit 7 on
C8C2      172 ;
C8C2 20 ED FD 173 jsr COUT ; output the data
C8C5      174 ;
C8C5 68 175 pla ; recall data
C8C6 10 F3 176 bpl PRTMSG ; bit 7 clear if branch
C8C8      177 ;
C8C8 18 178 clc ; clear C-flag
C8C9      179 ;
C8C9 60 180 rts ; return to caller
C8CA      181 ;
C8CA      182 ;

```

```

C8CA      183 ; Call the DISKADRS Table manager to set or restore a
C8CA      184 ; Table entry that corresponds to a slot number found in
C8CA      185 ; the X-reg.
C8CA      186 ;
C8CA 18    187 CLRDISK   clc                      ; restore DISKADRS Table entry
C8CB      188 ;
C8CB B0 00 189          bcs  *+2
C8CD      190          dfs  !-1
C8CC      191 ;
C8CC 38    192 SETDISK   sec                      ; set DISKADRS Table entry
C8CD      193 ;
C8CD 6C F2 BF 194          jmp  (MNGDISK)
C8D0      195 ;
C8D0      196 ;
C8D0      197 ; Phase 2 of the Boot Process.
C8D0      198 ;
C8D0      199 ; DOS Partition
C8D0      200 ;     Image (volume)      1:8
C8D0      201 ;     Drive          0
C8D0      202 ;     LBA            n:n+(IMAGSIZE-1)
C8D0      203 ;
C8D0      204 ;     n = ( ( Image-1 ) * IMAGSIZE ) + OFFSET0
C8D0      205 ;
C8D0      206 ;     OFFSET0 = 0x10
C8D0      207 ;     IMAGSIZE = 0x18
C8D0      208 ;
C8D0      209 ;
C8D0      210 ; Volume Partition
C8D0      211 ;     Sector          0:31
C8D0      212 ;     Track          0:47
C8D0      213 ;     Volume        0:255
C8D0      214 ;     Drive          1:CFMAXDRV
C8D0      215 ;
C8D0      216 ;
C8D0      217 ; Entry to boot from DOS Partition or Volume Partition.
C8D0      218 ; Delay 500 msec in case the CFFA RESET jumper is enabled.
C8D0      219 ;
C8D0 D8    220 DOBOOT    cld                      ; disable decimal mode
C8D1      221 ;
C8D1 AD 12 C0 222          lda  RDLGRAM          ; get state of LC RAM
C8D4 8D 15 01 223          sta  LGRAM          ; save the state
C8D7      224 ;
C8D7 8D 82 C0 225          sta  ROM2WP          ; enable ROM routines
C8DA      226 ;
C8DA A2 FF    227          ldx  #NEGONE          ; get initial stack value
C8DC      228 ;
C8DC 8E FB 04 229          stx  XMODE          ; set video firmware mode
C8DF 8E 0C C0 230          stx  VID80OFF        ; disable 80 column video
C8E2 8E 0E C0 231          stx  ALTCHOFF        ; enable normal character set
C8E5      232 ;
C8E5 20 84 FE 233          jsr  SETNORM          ; set normal text
C8E8 20 2F FB 234          jsr  INIT            ; init screen coordinates
C8EB 20 89 FE 235          jsr  SETVID          ; enable screen
C8EE 20 93 FE 236          jsr  SETKBD          ; enable keyboard
C8F1      237 ;
C8F1 A0 05    238          ldy  #WAITMLEN        ; get loop count
C8F3      239 ;
C8F3 A9 C5    240 ^1      lda  #WAIT100M          ; wait 100 msec
C8F5 20 BD CC 241          jsr  CFWAIT
C8F8      242 ;
C8F8 88      243          dey

```

```

C8F9 D0 F8      244      bne <1
C8FB            245      ;
C8FB 20 1F CD   246      jsr CFRESET          ; software reset the CF
C8FE B0 50      247      bcs >5              ; ERR=1 if branch
C900            248      ;
C900 A0 00      249      ldy #PAGE08          ; get PAGE08 address
C902 A9 08      250      lda /PAGE08
C904            251      ;
C904 84 26      252      sty BUFRADRZ          ; save address
C906 85 27      253      sta BUFRADRZ+1
C908            254      ;
C908 20 75 CD   255      jsr READID           ; get Identify Device
C90B B0 46      256      bcs >6              ; error if branch
C90D            257      ;
C90D A0 13      258      ldy #IDSRLLEN-1      ; get loop count
C90F            259      ;
C90F B9 14 08   260      ^2      lda PAGE08+IDSRLLOFF,Y ; check CF serial number
C912 D9 04 C8   261      cmp CFSRLBUF,Y
C915 D0 36      262      bne >4              ; error if branch
C917            263      ;
C917 88          264      dey
C918 10 F5      265      bpl <2
C91A            266      ;
C91A A9 08      267      lda /PAGE08          ; get PAGE08 address
C91C 85 27      268      sta BUFRADRZ+1      ; save address
C91E            269      ;
C91E A9 00      270      lda #ZERO            ; get ZERO
C920 AE 78 04   271      ldx CFSLOT          ; get CF slot index
C923            272      ;
C923 85 2D      273      sta SECTORZ          ; init boot sector
C925 85 2E      274      sta TRACKZ          ; init boot track
C927 85 3D      275      sta ROMSECTR        ; init boot sector
C929            276      ;
C929 9D 78 04   277      sta ERRSTAT,X        ; clear error status
C92C 9D F8 04   278      sta CFERROR,X        ; clear last CF error
C92F 9D 78 07   279      sta DOSVRSN,X        ; clear DOS version
C932            280      ;
C932 A5 2F      281      lda VOLUMEZ          ; get image/volume number
C934 9D F8 05   282      sta VOLUME,X        ; save it
C937            283      ;
C937 A5 2C      284      lda DRIVEZ          ; get drive number
C939 9D 78 05   285      sta DRIVE,X        ; save drive
C93C            286      ;
C93C CD 3A C8   287      cmp CFMAXDRV        ; range check drive
C93F 90 63      288      bcc BOOTVOL2        ; Volume partition if less
C941            289      ;
C941 D0 07      290      bne >3              ; error if branch
C943            291      ;
C943 A5 2F      292      lda VOLUMEZ          ; recall volume number
C945 CD 3B C8   293      cmp CFREMOVOL        ; range check remaining volume
C948 90 5A      294      bcc BOOTVOL2        ; Volume partition if less
C94A            295      ;
C94A A2 2A      296      ^3      ldx #ERRTEXT5-TEXTS ; Wrong drive value.
C94C            297      ;
C94C 2C 00 00   298      bit *-*
C94F            299      dfs !-2
C94D            300      ;
C94D A2 24      301      ^4      ldx #ERRTEXT3-TEXTS ; Wrong CF card.
C94F            302      ;
C94F 2C 00 00   303      bit *-*
C952            304      dfs !-2

```

```

C950          305 ;
C950 A2 1E     306 ^5      ldx #ERRTEXT1-TEXTS ; CF reset error.
C952          307 ;
C952 2C 00 00  308          bit *-*
C955          309          dfs !-2
C953          310 ;
C953 A2 21     311 ^6      ldx #ERRTEXT2-TEXTS ; CF read ID error.
C955          312 ;
C955 4C 6C C8  313          jmp EXITERR          ; exit CFFA
C958          314 ;
C958          315 ;
C958          316 ; Calculates the LBA and Page for a DOS Image from the DOS
C958          317 ; Partition where
C958          318 ;
C958          319 ;     drive = 0
C958          320 ;     volume = 1:8
C958          321 ;     OFFSET0 = 0x10
C958          322 ;     IMAGSIZE = 0x18
C958          323 ;
C958          324 ;     INDEX = ( track * 0x10 + sector & 0x0F ) / 2
C958          325 ;     PAGE = remainder
C958          326 ;
C958          327 ;     LBA = INDEX + ( ( image-1 ) * IMAGSIZE ) + OFFSET0
C958          328 ;
C958 A5 2C     329 CHKDRVZ  lda DRIVEZ          ; recall drive
C95A D0 2C     330          bne >1          ; branch if not image boot
C95C          331 ;
C95C 8D F8 05  332          sta LBA1          ; initialize to zero
C95F 8D 78 06  333          sta LBA2
C962 8D F8 06  334          sta LBA3
C965 8D 78 07  335          sta PAGE
C968          336 ;
C968 A4 2F     337          ldy VOLUMEZ          ; recall image number
C96A F0 24     338          beq >3          ; error if branch
C96C          339 ;
C96C C0 09     340          cpy #MAXIMAG+1          ; check maximum value
C96E B0 20     341          bcs >3          ; error if branch
C970          342 ;
C970 A5 2E     343          lda TRACKZ          ; recall track
C972 C9 04     344          cmp #MAXIMTRK          ; check maximum image value
C974 B0 13     345          bcs >2          ; branch if not image
C976          346 ;
C976 0A        347          asl          ; multiply by 16
C977 0A        348          asl
C978 0A        349          asl
C979 0A        350          asl
C97A          351 ;
C97A 05 2D     352          ora SECTORZ          ; add in sector
C97C          353 ;
C97C 4A        354          lsr          ; divide by 2 as 512-byte page
C97D 6E 78 07  355          ror PAGE          ; save carry as page selection
C980          356 ;
C980 79 FA CE  357          adc IMAGLBA-1,Y          ; add image LBA with OFFSET0
C983 8D 78 05  358          sta LBA0          ; save LBA value
C986          359 ;
C986 A5 2C     360          lda DRIVEZ          ; recall drive
C988          361 ;
C988 60        362 ^1      rts          ; return to caller
C989          363 ;
C989 A2 D3     364 ^2      ldx #DOSCOLDD          ; get DOS cold address
C98B A0 03     365          ldy /DOSCOLDD

```

```

C98D      366 ;
C98D 4C 75 C8 367      jmp EXITRWTS      ; exit CFFA
C990      368 ;
C990 A2 27      369 ^3      ldx #ERRTEXT4-TEXTS ; Wrong boot image value.
C992      370 ;
C992 2C 00 00 371      bit *-*
C995      372      dfs !-2
C993      373 ;
C993 A2 2D      374 ^4      ldx #ERRTEXT6-TEXTS ; LBA range error.
C995      375 ;
C995 2C 00 00 376      bit *-*
C998      377      dfs !-2
C996      378 ;
C996 A2 30      379 ^5      ldx #ERRTEXT7-TEXTS ; LBA read error.
C998      380 ;
C998 2C 00 00 381      bit *-*
C99B      382      dfs !-2
C999      383 ;
C999 A2 03      384 ^6      ldx #IOTEXT1B-TEXTS ; unable to connect CF
C99B      385 ;
C99B 4C 6C C8 386      jmp EXITERR      ; exit CFFA
C99E      387 ;
C99E      388 ;
C99E      389 ; For booting an image, drive = 0, volume = 1:8
C99E      390 ;   Uses CHKDRVZ to calculate next LBA and page
C99E      391 ;
C99E      392 ; For booting a volume, drive > 0, volume = 0:255
C99E      393 ;   Calculates LBA in the Volume Partition
C99E      394 ;   Uses DVTS2LBA to transform DVTS to LBA
C99E      395 ;
C99E      396 ; Reads T/S 0/0 and writes to PAGE08.
C99E      397 ; Monitors Boot Stage 1 while the DOS image is loaded.
C99E      398 ; Hooks the CF to the Disk Address Table at end of Stage 1
C99E      399 ;   and before Stage 2.
C99E      400 ;
C99E AD 12 C0 401 BOOTVOL  lda RDLGRAM      ; get state of LC RAM
C9A1 8D 15 01 402      sta LGRAM      ; save the state
C9A4      403 ;
C9A4 A5 3D      404 BOOTVOL2 lda ROMSECTR      ; recall sector value
C9A6 29 0F      405      and #SECMASKL      ; mask lower sector
C9A8 A8      406      tay      ; index to sector lookup
C9A9      407 ;
C9A9 B9 EB CE 408      lda RVSELEAV,Y      ; reverse sector lookup
C9AC 85 2D      409      sta SECTORZ      ; correct sector value
C9AE      410 ;
C9AE AE 78 04 411      ldx CFSLOT      ; get CF slot index
C9B1      412 ;
C9B1 BD F8 05 413      lda VOLUME,X      ; recall image/volume number
C9B4 85 2F      414      sta VOLUMEZ      ; save it
C9B6      415 ;
C9B6 BD 78 05 416      lda DRIVE,X      ; recall drive number
C9B9 85 2C      417      sta DRIVEZ      ; save it
C9BB      418 ;
C9BB 20 58 C9 419      jsr CHKDRVZ      ; check if DRIVEZ is zero
C9BE F0 0C      420      beq >1      ; branch if booting an image
C9C0      421 ;
C9C0 C6 2C      422      dec DRIVEZ      ; align drive to 0:CFMAXDRV-1
C9C2      423 ;
C9C2 A5 3D      424      lda ROMSECTR      ; recall sector value
C9C4 20 DE CB 425      jsr DVTS2LBA      ; convert to LBA
C9C7      426 ;

```

```

C9C7 20 30 CC 427      jsr CHKCFLBA      ; range check the LBA
C9CA B0 C7 428      bcs <4      ; error if branch
C9CC 429      ;
C9CC 20 BD CD 430      ^1      jsr RD1PAGE      ; read selected page
C9CF B0 C5 431      bcs <5      ; error if branch
C9D1 432      ;
C9D1 433      ;
C9D1 434      ; DOS 3.X and 4.X load RWTS on track 0 from sector N given
C9D1 435      ; by BOOTPGS to sector 0. When BOOTPGS contains a negative
C9D1 436      ; value, RWTS is now fully in memory. A check for which
C9D1 437      ; DOS is in memory can be made at this time.
C9D1 438      ;
C9D1 AD FF 08 439      lda BOOTPGS      ; get decremented boot page
C9D4 10 4E 440      bpl >3      ; not negative if branch
C9D6 441      ;
C9D6 442      ;
C9D6 443      ; DOS 4.X.L Boot Stage 1 loads RWTS from 0xB900 to 0xBFFF.
C9D6 444      ; DOS 4.X.H Boot Stage 1 loads RWTS from 0xD000 to 0xDFFF.
C9D6 445      ;
C9D6 446      ; Now check for DOS 4.X RWTS.
C9D6 447      ;
C9D6 20 93 CA 448      jsr CHKDOS45      ; check for DOS 4.5 code
C9D9 90 0A 449      bcc >1
C9DB 450      ;
C9DB 20 7F CA 451      jsr CHKDOS43      ; check for DOS 4.3 code
C9DE 90 05 452      bcc >1
C9E0 453      ;
C9E0 20 DA CA 454      jsr CHKDOS41      ; check for DOS 4.1 code
C9E3 B0 14 455      bcs >2
C9E5 456      ;
C9E5 AE 78 04 457      ^1      ldx CFSLOT      ; get CF slot index
C9E8 AC FC BF 458      ldy BCFGNDX      ; get index to BOOTCFG
C9EB 459      ;
C9EB BD 78 05 460      lda DRIVE,X      ; get saved drive
C9EE 99 00 BF 461      sta PAGEBF,Y      ; save to BOOTCFG DNUM
C9F1 462      ;
C9F1 BD F8 05 463      lda VOLUME,X      ; get saved volume
C9F4 99 01 BF 464      sta PAGEBF+1,Y      ; save to BOOTCFG VOLEXPT
C9F7 465      ;
C9F7 90 2B 466      bcc >3      ; always taken
C9F9 467      ;
C9F9 468      ;
C9F9 469      ; DOS 3.X Boot Stage 1 loads memory from 0xBFFF to 0xB600.
C9F9 470      ;
C9F9 471      ; Now check for DOS 3.X RWTS.
C9F9 472      ;
C9F9 20 3A CA 473      ^2      jsr CHKDOS3X      ; check for DOS 3.X code
C9FC B0 9B 474      bcs <6      ; not DOS 3.X if branch
C9FE 475      ;
C9FE AE 78 04 476      ldx CFSLOT      ; get CF slot index
CA01 477      ;
CA01 BD 78 05 478      lda DRIVE,X      ; get saved drive
CA04 8D 07 B7 479      sta SETDRIVE+1      ; save to boot stage 2
CA07 480      ;
CA07 BD F8 05 481      lda VOLUME,X      ; get saved volume
CA0A 8D EB B7 482      sta VOLEXPT      ; save to RWTS IOCB
CA0D 483      ;
CA0D AC 48 B7 484      ldy RESTART+4      ; get address to DOSSTRT
CA10 AD 49 B7 485      lda RESTART+5
CA13 486      ;
CA13 8C 10 01 487      sty RSTRTSAB      ; save address

```

```
CA16 8D 11 01    488          sta RSTRTSV+1
CA19            489          ;
CA19 A0 F3      490          ldy #MODOS3          ; get CF modify address
CA1B AD F8 07    491          lda CFPAGECX          ; get CF MSB address
CA1E            492          ;
CA1E 8C 48 B7    493          sty RESTART+4          ; save address
CA21 8D 49 B7    494          sta RESTART+5
CA24            495          ;
CA24            496          ;
CA24            497          ; Normally, BUFRADRZ+1 would be incremented here, but it
CA24            498          ; was already incremented above by RD1PAGE.
CA24            499          ;
CA24            500          ; The design of DOS 4.3 is flawed at the start of STAGE 2.
CA24            501          ; DOS 4.3 requires Language Card memory to be re-enabled
CA24            502          ; before STAGE 2 can begin properly. DOS 4.1 and DOS 4.5
CA24            503          ; both enable Language Card memory properly before RWTS is
CA24            504          ; called. Enabling Language Card memory here does not
CA24            505          ; affect the booting of either DOS 4.1 or DOS 4.5.
CA24            506          ;
CA24 E6 3D      507          ^3          inc ROMSECTR          ; next sector
CA26            508          ;
CA26 2C 15 01    509          bit LCRAM          ; get state of LC RAM
CA29 10 06      510          bpl >4          ; branch if LC RAM not enabled
CA2B            511          ;
CA2B 2C 8B C0    512          bit RAM1WE          ; enable LC RAM, bank 1
CA2E 2C 8B C0    513          bit RAM1WE
CA31            514          ;
CA31 A5 3D      515          ^4          lda ROMSECTR          ; copy Disk ][ ROM example
CA33            516          ;
CA33 A2 01      517          ldx #PAGE08+1          ; get exit address
CA35 A0 08      518          ldy /PAGE08+1
CA37            519          ;
CA37 4C 75 C8    520          jmp EXITRWTS          ; exit CFFA
CA3A            521          ;
CA3A            522          ;
CA3A            523          icl "ROM2.L"
```

LLOAD ROM2.L,A\$4000


```

CA3A          1          ttl "CFFA Firmware Source Code, ROM2.L"
CA3A          2          ;
CA3A          3          ;
CA3A          4          ; ROM2.L
CA3A          5          ;
CA3A          6          ;
CA3A          7          ; Check for DOS 3.X CALLRWTS routine.
CA3A          8          ;
CA3A A0 04     9  CHKDOS3X ldy #FNDOSLEN-1      ; get length of code check
CA3C          10         ;
CA3C B9 BD B7 11  ^1      lda CALLRWTS+8,Y      ; get CALLRWTS code
CA3F D9 7A CA 12          cmp FNDOS,Y          ; compare to known code
CA42 D0 39     13          bne NODOS            ; not DOS 3.X if branch
CA44          14         ;
CA44 88        15          dey
CA45 10 F5     16          bpl <1
CA47          17         ;
CA47          18         ;
CA47          19         ; Connect the CF to DOS 3.X.
CA47          20         ;
CA47 AE 78 04 21          ldx CFSLOT            ; get CF slot index
CA4A          22         ;
CA4A AC B8 B7 23          ldy CALLRWTS+3        ; get LSB RWTS address
CA4D AD B9 B7 24          lda CALLRWTS+4        ; get MSB RWTS address
CA50          25         ;
CA50 C0 3B     26          cpy #DISKRWTS        ; check entry LSB address
CA52 D0 05     27          bne >2              ; not connected if branch
CA54          28         ;
CA54 CD F8 07 29          cmp CFPAGECX          ; check entry MSB address
CA57 F0 21     30          beq FNDOS            ; already connected if branch
CA59          31         ;
CA59 9D F8 06 32  ^2      sta SAVEADRH,X        ; save MSB
CA5C          33         ;
CA5C 98        34          tya
CA5D 9D 78 06 35          sta SAVEADRL,X        ; save LSB
CA60          36         ;
CA60 A0 3B     37          ldy #DISKRWTS        ; get DOS 3.X entry address
CA62 AD F8 07 38          lda CFPAGECX          ; get CF MSB address
CA65          39         ;
CA65 8C B8 B7 40          sty CALLRWTS+3        ; store LSB address
CA68 8D B9 B7 41          sta CALLRWTS+4        ; store MSB address
CA6B          42         ;
CA6B A9 33     43          lda #VRSN3.3         ; get DOS 3.X version
CA6D 9D 78 07 44          sta DOSVRSN,X        ; save version
CA70          45         ;
CA70 A9 BB     46          lda /NBUF1BUF        ; get nibble buffer page
CA72 9D F8 07 47          sta DOSNBUF1,X        ; save nibble buffer page
CA75          48         ;
CA75          49         ;
CA75          50         ; Able to connect CF and DOS3.X.
CA75          51         ;
CA75 A2 12     52          ldx #IOTEXT3A-TEXTS  ; get print message
CA77          53         ;
CA77 4C 99 C8 54          jmp PRTMESGS
CA7A          55         ;
CA7A          56         ;
CA7A          57         ; Code extract from DOS 3.X CALLRWTS.
CA7A          58         ;
CA7A 18        59  FNDOS   clc
CA7B          60         ;

```

```

CA7B 60          61          rts
CA7C          62          ;
CA7C 28          63          plp
CA7D          64          ;
CA7D 38          65          NODOS    sec
CA7E          66          ;
CA7E 60          67          rts
CA7F          68          ;
0005          69          FNDOSLEN equ *-FNDOS
CA7F          70          ;
CA7F          71          ;
CA7F          72          ; Check for DOS 4.3 initialization values.
CA7F          73          ;
CA7F AD F0 BF    74          CHKDOS43 lda BLDVRSN          ; get DOS version
CA82 C9 43       75          cmp #VRSN4.3
CA84 D0 F7       76          bne NODOS
CA86          77          ;
CA86 AD F1 BF    78          lda BLDNMBR          ; get DOS build
CA89 C9 08       79          cmp #BLD4.3
CA8B D0 F0       80          bne NODOS
CA8D          81          ;
CA8D A2 18       82          ldx #IOTEXT3C-TEXTS ; get print message
CA8F          83          ;
CA8F A9 43       84          lda #VRSN4.3          ; get DOS 4.3 version
CA91 D0 16       85          bne CHKDOS4X          ; always taken
CA93          86          ;
CA93          87          ;
CA93          88          ; Check for DOS 4.5 initialization values. Allow previous
CA93          89          ; and current build number as valid.
CA93          90          ;
CA93 AD F0 BF    91          CHKDOS45 lda BLDVRSN          ; get DOS version
CA96 C9 45       92          cmp #VRSN4.5
CA98 D0 E3       93          bne NODOS
CA9A          94          ;
CA9A AD F1 BF    95          lda BLDNMBR          ; get DOS build
CA9D C9 07       96          cmp #BLD4.5+1
CA9F B0 DC       97          bcs NODOS
CAA1          98          ;
CAA1 C9 05       99          cmp #BLD4.5-1
CAA3 90 D8      100          bcc NODOS
CAA5          101          ;
CAA5 A2 1B      102          ldx #IOTEXT3D-TEXTS ; get print message
CAA7 A9 45      103          lda #VRSN4.5          ; get DOS 4.5 version
CAA9          104          ;
CAA9          105          ;
CAA9          106          ; Connect the CFFA to the DOS 4.3/5 Disk Address Table.
CAA9          107          ;
CAA9 8E 12 01    108          CHKDOS4X stx MESSAGE          ; save message number
CAAC 8D 13 01    109          sta VERSION          ; save version number
CAAF          110          ;
CAAF A9 00      111          lda #ZERO          ; request current address
CAB1          112          ;
CAB1 AE 78 04    113          ldx CFSLOT          ; get CF slot index
CAB4 20 CC C8    114          jsr SETDISK          ; read current entry
CAB7          115          ;
CAB7 C0 4B      116          cpy #CFRWTS          ; check entry LSB address
CAB9 D0 05      117          bne >1          ; not connected if branch
CABB          118          ;
CABB CD F8 07    119          cmp CFPAGECX          ; check for CF slot page
CABE F0 BA      120          beq FNDOS          ; already connected if branch
CAC0          121          ;

```

```

CAC0 A0 4B      122  ^1      ldy #CFRWTS          ; get CF slot entry address
CAC2 AD F8 07   123          lda CFPAGECX          ; get CF MSB address
CAC5            124  ;
CAC5 20 CC C8   125          jsr SETDISK          ; save the entry
CAC8            126  ;
CAC8 AD 13 01   127          lda VERSION          ; recall version number
CACB 9D 78 07   128          sta DOSVRSN,X        ; save version
CACE            129  ;
CACE AD FD BF   130          lda NBUF1ADR          ; get nibble buffer page
CAD1 9D F8 07   131          sta DOSNBUF1,X        ; save nibble buffer page
CAD4            132  ;
CAD4            133  ;
CAD4            134  ; Able to connect CF and DOS 4.3/5.
CAD4            135  ;
CAD4 AE 12 01   136          ldx MESSAGE          ; recall message number
CAD7            137  ;
CAD7 4C 99 C8   138          jmp PRTMESGS
CADA            139  ;
CADA            140  ;
CADA            141  ; Check for DOS 4.1 initialization values.
CADA            142  ;
CADA AD F8 BF   143  CHKDOS41 lda INITDOS          ; get INITDOS LSB address
CADD C9 D9      144          cmp #INITADR
CADF D0 9C      145          bne NODOS
CAE1            146  ;
CAE1 AD F9 BF   147          lda INITDOS+1        ; get INITDOS MSB address
CAE4 C9 BE      148          cmp /INITADR
CAE6 D0 95      149          bne NODOS
CAE8            150  ;
CAE8            151  ;
CAE8            152  ; Connect the CFFA to the DOS 4.1 Disk Address Table.
CAE8            153  ;
CAE8 20 8E CB   154          jsr INITDAT          ; point to Disk Address Table
CAEB            155  ;
CAEB B1 EE      156          lda (DATPTR),Y        ; get MSB from DAT
CAED CD F8 07   157          cmp CFPAGECX          ; check for CF slot page
CAF0 F0 88      158          beq FNDOS            ; already connected if branch
CAF2            159  ;
CAF2 9D F8 06   160          sta SAVEADRH,X        ; save MSB
CAF5            161  ;
CAF5 AD F8 07   162          lda CFPAGECX          ; get CF MSB address
CAF8 91 EE      163          sta (DATPTR),Y        ; save MSB to DAT
CAFA            164  ;
CAFA 88         165          dey                  ; point to LSB
CAFB            166  ;
CAFB B1 EE      167          lda (DATPTR),Y        ; get LSB from DAT
CAFD 9D 78 06   168          sta SAVEADRL,X        ; save LSB
CB00            169  ;
CB00 A9 4B      170          lda #CFRWTS          ; get CF slot entry address
CB02 91 EE      171          sta (DATPTR),Y        ; save LSB to DAT
CB04            172  ;
CB04 A9 41      173          lda #VRSN4.1          ; get DOS 4.1 version
CB06 9D 78 07   174          sta DOSVRSN,X        ; save version
CB09            175  ;
CB09 AD FD BF   176          lda NBUF1ADR          ; get nibble buffer page
CB0C 9D F8 07   177          sta DOSNBUF1,X        ; save nibble buffer page
CB0F            178  ;
CB0F            179  ;
CB0F            180  ; Able to connect CF and DOS 4.1.
CB0F            181  ;
CB0F A2 15      182          ldx #IOTEXT3B-TEXTS ; get print message

```

```

CB11      183 ;
CB11 4C 99 C8 184      jmp PRTMESGS
CB14      185 ;
CB14      186 ;
CB14      187 ; Connect or disconnect the CF to DOS based on the state
CB14      188 ; of the Y-reg. Verify the connection.
CB14      189 ;
CB14 98      190 DOTOGGLE tya      ; recall connection flag
CB15 D0 18    191      bne >1      ; unhook DOS if branch
CB17      192 ;
CB17 20 93 CA 193      jsr CHKDOS45    ; check for DOS 4.5 code
CB1A 90 43    194      bcc >4      ; found DOS 4.5 if branch
CB1C      195 ;
CB1C 20 7F CA 196      jsr CHKDOS43    ; check for DOS 4.3 code
CB1F 90 3E    197      bcc >4      ; found DOS 4.3 if branch
CB21      198 ;
CB21 20 DA CA 199      jsr CHKDOS41    ; check for DOS 4.1 code
CB24 90 39    200      bcc >4      ; found DOS 4.1 if branch
CB26      201 ;
CB26 20 3A CA 202      jsr CHKDOS3X    ; check for DOS 3.X code
CB29 90 34    203      bcc >4      ; found DOS 3.X if branch
CB2B      204 ;
CB2B      205 ;
CB2B      206 ; Unable to connect CF and DOS.
CB2B      207 ;
CB2B A2 03    208      ldx #IOTEXT1B-TEXTS ; get print message
CB2D B0 2D    209      bcs >3      ; always taken
CB2F      210 ;
CB2F      211 ;
CB2F      212 ; Attempt to disconnect CF and DOS.
CB2F      213 ;
CB2F AE 78 04 214 ^1      ldx CFSLOT      ; get CF slot index
CB32      215 ;
CB32 BC 78 07 216      ldy DOSVRSN,X    ; get DOS version
CB35 F0 28    217      beq >4      ; do nothing if branch
CB37      218 ;
CB37 A9 00    219      lda #ZERO
CB39 9D 78 07 220      sta DOSVRSN,X    ; clear DOS version
CB3C      221 ;
CB3C C0 33    222      cpy #VRSN3.3    ; check for DOS 3.X
CB3E F0 3E    223      beq USTDOS33    ; DOS 3.X if branch
CB40      224 ;
CB40 C0 41    225      cpy #VRSN4.1    ; check for DOS 4.1
CB42 F0 28    226      beq USTDOS41    ; DOS 4.1 if branch
CB44      227 ;
CB44 A9 0C    228      lda #IOTEXT2C-TEXTS ; get print message
CB46      229 ;
CB46 C0 43    230      cpy #VRSN4.3    ; check for DOS 4.3
CB48 F0 06    231      beq USTDOS4X    ; DOS 4.3 if branch
CB4A      232 ;
CB4A A9 0F    233      lda #IOTEXT2D-TEXTS ; get print message
CB4C      234 ;
CB4C C0 45    235      cpy #VRSN4.5    ; check for DOS 4.5
CB4E D0 0A    236      bne >2      ; DOS 4.5 if not branch
CB50      237 ;
CB50      238 ;
CB50      239 ; Disconnect the CF from DOS 4.3/5.
CB50      240 ;
CB50 8D 12 01 241 USTDOS4X sta MESSAGE    ; save message number
CB53      242 ;
CB53 20 CA C8 243      jsr CLRDISK      ; restore default entry

```

```

CB56      244 ;
CB56      245 ;
CB56      246 ; Able to disconnect CF and DOS 4.5.
CB56      247 ;
CB56 AE 12 01 248      ldx MESSAGE      ; recall message number
CB59      249 ;
CB59 2C 00 00 250      bit *-*
CB5C      251      dfs !-2
CB5A      252 ;
CB5A      253 ;
CB5A      254 ; Unable to disconnect CF and DOS.
CB5A      255 ;
CB5A A2 00 256      ^2      ldx #IOTEXT1A-TEXTS ; get print message
CB5C      257 ;
CB5C 20 99 C8 258      ^3      jsr PRTMESGS      ; print messages
CB5F      259 ;
CB5F A0 EA 260      ^4      ldy #HOOKDOS      ; get LSB
CB61 A9 03 261      lda /HOOKDOS      ; get MSB
CB63      262 ;
CB63 8C 1B 01 263      sty EXITADR+1      ; save address
CB66 8D 1C 01 264      sta EXITADR+2
CB69      265 ;
CB69 4C 17 01 266      jmp EXIT2
CB6C      267 ;
CB6C      268 ;
CB6C      269 ; Disconnect the CF from DOS 4.1.
CB6C      270 ;
CB6C 20 8E CB 271      USTDOS41 jsr INITDAT      ; point to Disk Address Table
CB6F      272 ;
CB6F BD F8 06 273      lda SAVEADRH,X      ; get saved MSB address
CB72 91 EE 274      sta (DATPTR),Y      ; save MSB to DAT
CB74      275 ;
CB74 88 276      dey      ; point to LSB
CB75      277 ;
CB75 BD 78 06 278      lda SAVEADRL,X      ; get saved LSB address
CB78 91 EE 279      sta (DATPTR),Y      ; save LSB to DAT
CB7A      280 ;
CB7A      281 ;
CB7A      282 ; Able to disconnect CF and DOS 4.1.
CB7A      283 ;
CB7A A2 09 284      ldx #IOTEXT2B-TEXTS ; get print message
CB7C D0 DE 285      bne <3      ; always taken
CB7E      286 ;
CB7E      287 ;
CB7E      288 ; Disconnect the CF from DOS 3.X.
CB7E      289 ;
CB7E BC 78 06 290      USTDOS33 ldy SAVEADRL,X      ; get saved LSB address
CB81 BD F8 06 291      lda SAVEADRH,X      ; get saved MSB address
CB84      292 ;
CB84 8C B8 B7 293      sty CALLRWTS+3      ; store LSB address
CB87 8D B9 B7 294      sta CALLRWTS+4      ; store MSB address
CB8A      295 ;
CB8A      296 ;
CB8A      297 ; Able to disconnect CF and DOS 3.X.
CB8A      298 ;
CB8A A2 06 299      ldx #IOTEXT2A-TEXTS ; get print message
CB8C D0 CE 300      bne <3      ; always taken
CB8E      301 ;
CB8E      302 ;
CB8E      303 ; Initialize a pointer to the slot index of DOS 4.1
CB8E      304 ; Disk Address Table.

```

```

CB8E          305 ;
CB8E AE 78 04 306 INITDAT ldx CFSLOT ; get CF slot index
CB91          307 ;
CB91 AC FB BF 308 ldy DISKTBL ; get DAT offset (LSB)
CB94 A9 BF 309 lda /DISKTBL ; get DAT memory page (MSB)
CB96          310 ;
CB96 84 EE 311 sty DATPTR ; save LSB
CB98 85 EF 312 sta DATPTR+1 ; save MSB
CB9A          313 ;
CB9A 8A 314 txa ; form table index
CB9B 0A 315 asl
CB9C          316 ;
CB9C A8 317 tay ; point to LSB
CB9D C8 318 iny ; point to MSB
CB9E          319 ;
CB9E 60 320 rts ; return to caller
CB9F          321 ;
CB9F          322 ;
CB9F AE 78 04 323 DOMODOS3 ldx CFSLOT ; get CF slot index
CBA2          324 ;
CBA2 BD F8 05 325 lda VOLUME,X ; recall volume
CBA5 8D 66 AA 326 sta VOLVAL ; save in VOLVAL
CBA8          327 ;
CBA8          328 ;
CBA8          329 ; not sure if this needs to be done
CBA8          330 ;
CBA8 8D EB B7 331 sta VOLEXPT ; save in VOLEXPT
CBAB          332 ;
CBAB          333 ;
CBAB          334 ; Patches to DOS 3.X to support the CFFA.
CBAB          335 ;
CBAB          336 ; lda #$10 ; #CMDCLOSE-CMDTBL
CBAB          337 ; sta FTMOD+1 ; new FRSTIME boot command
CBAB          338 ;
CBAB A9 65 339 lda #KYWRDFND ; LSB of KYWRDFND
CBAD 8D DA A0 340 sta SDFMOD+1 ; overwrite VOLVAL LSB
CBB0          341 ;
CBB0 AD 3A C8 342 lda CFMAXDRV ; get maximum CF drives
CBB3 8D 5B A9 343 sta KWRANGE+6 ; overwrite drive KWRANGE
CBB6          344 ;
CBB6 A9 FE 345 lda #VOLNUMBR+5 ; LSB of $B5FE; unused memory
CBB8 8D 9E AD 346 sta CATHMOD1+1 ; overwrite VOLNUMBR LSB
CBBB          347 ;
CBBB A0 17 348 ldy #NAME SIZE-1 ; 24 character filenames
CBBD 8C 17 AE 349 sty CATHMOD2+1 ; overwrite loop count
CBC0          350 ;
CBC0 C8 351 iny ; 24 character filenames
CBC1 8C 03 B2 352 sty LCDMOD+1 ; overwrite NAME SIZE
CBC4          353 ;
CBC4 A9 01 354 lda #1 ; get original drive value
CBC6 8D 07 B7 355 sta SETDRIVE+1 ; overwrite drive value
CBC9          356 ;
CBC9 AC 10 01 357 ldy RSTRTSAV ; recall RESTART+4
CBCC AD 11 01 358 lda RSTRTSAV+1 ; recall RESTART+5
CBCF          359 ;
CBCF 8C 48 B7 360 sty RESTART+4 ; save LSB
CBD2 8D 49 B7 361 sta RESTART+5 ; save MSB
CBD5          362 ;
CBD5 8C 1B 01 363 sty EXITADR+1 ; save address
CBD8 8D 1C 01 364 sta EXITADR+2
CBDB          365 ;

```

```
CBDB 4C 17 01    366      jmp EXIT2      ; exit CFFA
CBDE             367      ;
CBDE             368      ;
CBDE             369      icl "ROM3.L"

LLOAD ROM3.L,A$4000
```

```

CBDE      1          ttl "CFFA Firmware Source Code, ROM3.L"
CBDE      2      ;
CBDE      3      ;
CBDE      4      ; ROM3.L
CBDE      5      ;
CBDE      6      ;
CBDE      7      ; Calculate LBA for Volume Partition.
CBDE      8      ;
CBDE      9      ; Before calling DVTS2LBA calculate page and sector.
CBDE     10      ; Sector number must be in A-reg.
CBDE     11      ;
CBDE     12      ;     page = Sector & 0x10
CBDE     13      ;     sector = Sector & 0x0F
CBDE     14      ;
CBDE     15      ;     sector = 0:15
CBDE     16      ;     track = 0:47
CBDE     17      ;     volume = 0:255
CBDE     18      ;     drive = 1:81
CBDE     19      ;     OFFSET1 = 0x0100
CBDE     20      ;
CBDE     21      ; LBA = ( (drive-1) * 0x30000 ) + ( volume * 0x300 )
CBDE     22      ;     + ( track * 0x10 ) + sector + OFFSET1
CBDE     23      ;
CBDE 29 10     24 DVTS2LBA and #SECMASKH      ; mask upper sector
CBE0 8D 78 07 25      sta PAGE                ; save page flag
CBE3      26      ;
CBE3 A5 2E     27      lda TRACKZ              ; get track
CBE5      28      ;
CBE5 0A      29      asl                      ; multiply by 16
CBE6 0A      30      asl
CBE7 0A      31      asl
CBE8 0A      32      asl
CBE9      33      ;
CBE9 05 2D     34      ora SECTORZ             ; add in sector
CBE8 8D 78 05 35      sta LBA0                ; track*16 | sector
CBEE      36      ;
CBEE A5 2F     37      lda VOLUMEZ             ; get volume
CBF0 29 0F     38      and #NIBLMASK          ; mask lower nibble
CBF2      39      ;
CBF2 AA      40      tax                      ; volume low nibble
CBF3      41      ;
CBF3 A5 2F     42      lda VOLUMEZ             ; recall volume
CBF5      43      ;
CBF5 4A      44      lsr                      ; divide by 16
CBF6 4A      45      lsr
CBF7 4A      46      lsr
CBF8 4A      47      lsr
CBF9      48      ;
CBF9 A8      49      tay                      ; volume high nibble
CBFA      50      ;
CBFA A5 2E     51      lda TRACKZ              ; recall track
CBFC      52      ;
CBFC 4A      53      lsr                      ; divide by 16
CBFD 4A      54      lsr
CBFE 4A      55      lsr
CBFF 4A      56      lsr
CC00      57      ;
CC00      58      ;     clc                  ; clear C-flag
CC00 38      59      sec                      ; add in /OFFSET1
CC01      60      ;

```



```

CC01          61 ;      adc /OFFSET1      ; add in OFFSET1
CC01 7D 03 CF  62      adc VOLXTBL,X      ; add in LSB of
CC04 79 13 CF  63      adc VOLYTBLH,Y      ;   volume*0x300
CC07          64 ;
CC07 8D F8 05  65      sta LBA1            ; save volume*0x300+track/16
CC0A          66 ;
CC0A A9 00     67      lda #ZERO           ; get zero
CC0C 79 23 CF  68      adc VOLYTBLH,Y      ; add in volume*0x300 MSB
CC0F          69 ;
CC0F 48        70      pha                ; save volume*0x300 MSB
CC10          71 ;
CC10 A5 2C     72      lda DRIVEZ          ; get drive
CC12 29 0F     73      and #NIBLMASK       ; mask lower nibble
CC14          74 ;
CC14 AA        75      tax                ; drive low nibble
CC15          76 ;
CC15 A5 2C     77      lda DRIVEZ          ; recall drive
CC17          78 ;
CC17 4A        79      lsr                ; divide by 16
CC18 4A        80      lsr
CC19 4A        81      lsr
CC1A 4A        82      lsr
CC1B          83 ;
CC1B A8        84      tay                ; drive high nibble
CC1C          85 ;
CC1C 18        86      clc                ; clear C-flag
CC1D          87 ;
CC1D 68        88      pla                ; recall volume*0x300 MSB
CC1E          89 ;
CC1E 7D 03 CF  90      adc DRVXTBL,X      ; add in value of
CC21 79 13 CF  91      adc DRVYTBLH,Y      ;   drive*0x30000
CC24          92 ;
CC24 8D 78 06  93      sta LBA2            ; drive*0x30000 + volume MSB
CC27          94 ;
CC27 A9 00     95      lda #ZERO           ; get zero
CC29 79 23 CF  96      adc DRVYTBLH,Y      ; add in drive*0x30000 MSB
CC2C          97 ;
CC2C 8D F8 06  98      sta LBA3            ; save drive*0x30000 MSB
CC2F          99 ;
CC2F 60       100      rts
CC30         101 ;
CC30         102 ;
CC30         103 ; Range check the LBA.
CC30         104 ;
CC30 AD F8 06  105  CHKCFLBA  lda LBA3      ; get LBA3
CC33 CD 36 C8  106            cmp CFLBANUM ; check MSB
CC36 90 1C     107            bcc >1       ; exit if branch
CC38         108 ;
CC38 D0 1A     109            bne >1       ; exit if branch
CC3A         110 ;
CC3A AD 78 06  111            lda LBA2      ; get LBA2
CC3D CD 37 C8  112            cmp CFLBANUM+1 ; check second byte
CC40 90 12     113            bcc >1       ; exit if branch
CC42         114 ;
CC42 D0 10     115            bne >1       ; exit if branch
CC44         116 ;
CC44 AD F8 05  117            lda LBA1      ; get LBA1
CC47 CD 38 C8  118            cmp CFLBANUM+2 ; check third byte
CC4A 90 08     119            bcc >1       ; exit if branch
CC4C         120 ;
CC4C D0 06     121            bne >1       ; exit if branch

```

```

CC4E      122 ;
CC4E AD 78 05 123         lda LBA0             ; get LBA0
CC51 CD 39 C8 124         cmp CFLBANUM+3        ; check LSB
CC54      125 ;
CC54 60     126 ^1      rts                     ; return to caller
CC55      127 ;
CC55      128 ;
CC55      129 ; References include OEM Product Manual for the SanDisk
CC55      130 ; CompactFlash Memory Card and ATA PIO Mode from OSDev
CC55      131 ; Wiki.
CC55      132 ;
CC55      133 ; The maximum time the BUSY bit should take to clear is 20
CC55      134 ; msec for IDE. However, the CompactFlash BUSY bit can
CC55      135 ; take up to 200 msec to clear. If this time is exceeded
CC55      136 ; the CompactFlash should be software reset.
CC55      137 ;
CC55      138 ; Status register:
CC55      139 ;
CC55      140 ; |  D7  |  D6  |  D5  |  D4  |  D3  |  D2  |  D1  |  D0  |
CC55      141 ; | ---- | ---- | ---- | ---- | ---- | ---- | ---- | ---- |
CC55      142 ; | BUSY | RDY  | DWF  | DSC  | DRQ  | CORR | 0    | ERR  |
CC55      143 ;
CC55 A0 C8   144 CFBUSY  ldy #200                 ; get loop count
CC57      145 ;
CC57 BD 8F C0 146 ^1     lda ATASTAT,X           ; get CF status
CC5A 10 0F    147         bpl >2                 ; status clear if branch
CC5C      148 ;
CC5C A9 11    149         lda #WAIT001M          ; wait 1 msec
CC5E 20 BD CC 150         jsr CFWAIT
CC61      151 ;
CC61 88       152         dey
CC62 D0 F3    153         bne <1                 ; not zero if branch
CC64      154 ;
CC64 20 AC CC 155         jsr DOATARST           ; do software reset
CC67      156 ;
CC67 A9 01    157         lda #BUSYERR           ; get BUSY error
CC69      158 ;
CC69 38       159         sec
CC6A      160 ;
CC6A 60       161         rts                     ; return with error
CC6B      162 ;
CC6B 18       163 ^2     clc
CC6C      164 ;
CC6C 60       165         rts                     ; return with no error
CC6D      166 ;
CC6D      167 ;
CC6D      168 ; Select drive first, then test for READY. READY should
CC6D      169 ; be set almost immediately but can take up to 10 seconds.
CC6D      170 ;
CC6D      171 ; Drive/Head register:
CC6D      172 ;
CC6D      173 ; |  D7  |  D6  |  D5  |  D4  |  D3  |  D2  |  D1  |  D0  |
CC6D      174 ; | ---- | ---- | ---- | ---- | ---- | ---- | ---- | ---- |
CC6D      175 ; |  1  | LBA  |  1  | DRV  | A27  | A26  | A25  | A24  |
CC6D      176 ;
CC6D BD 82 C0 177 CFREADY  lda CLRCMSK,X         ; enable 6502 pre-fetch
CC70      178 ;
CC70 A9 E0    179         lda #%11100000          ; 1,LBA,1,DRV
CC72 9D 8E C0 180         sta ATAHEAD,X         ; set drive to zero
CC75      181 ;
CC75 A0 FA    182         ldy #250                 ; get loop count

```

```

CC77      183 ;
CC77 BD 8F C0 184 ^1      lda ATASTAT,X      ; get CF status
CC7A 29 D0      185      and #%11010000      ; mask BUSY, RDY, DSC
CC7C      186 ;
CC7C C9 50      187      cmp #%01010000      ; RDY, DSC
CC7E F0 0C      188      beq >2              ; BUSY and RDY set if branch
CC80      189 ;
CC80 A9 7C      190      lda #WAIT040M      ; wait 40 msec
CC82 20 BD CC   191      jsr CFWAIT
CC85      192 ;
CC85 88      193      dey
CC86 D0 EF      194      bne <1              ; not zero if branch
CC88      195 ;
CC88 A9 02      196      lda #RDYERR      ; get READY error
CC8A      197 ;
CC8A 38      198      sec
CC8B      199 ;
CC8B 60      200      rts              ; return with error
CC8C      201 ;
CC8C 18      202 ^2      clc
CC8D      203 ;
CC8D 60      204      rts              ; return with no error
CC8E      205 ;
CC8E      206 ;
CC8E      207 ; Normal CF command processing occurs with ERR=0 and
CC8E      208 ; DRQ=1. As soon as all data is transferred then DRQ=0.
CC8E      209 ; If ERR=1 then the Error register is copied to ERRSTAT.
CC8E      210 ;
CC8E      211 ; Error register:
CC8E      212 ;
CC8E      213 ; | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
CC8E      214 ; | ---- | ---- | ---- | ---- | ---- | ---- | ---- | ---- |
CC8E      215 ; | BRK | UNC | 0 | IDNF | 0 | ABRT | 0 | AMNF |
CC8E      216 ;
CC8E AC 78 04   217 CFSTATUS ldy CFSLOT      ; get CF slot index
CC91      218 ;
CC91 BD 8F C0   219      lda ATASTAT,X      ; get CF status
CC94 29 01      220      and #%00000001      ; mask out ERR state
CC96 D0 0A      221      bne >1              ; ERR is set if branch
CC98      222 ;
CC98 99 78 04   223      sta ERRSTAT,Y      ; save status
CC9B      224 ;
CC9B BD 8F C0   225      lda ATASTAT,X      ; recall CF status
CC9E 29 08      226      and #%00001000      ; mask out DRQ state
CCA0      227 ;
CCA0 18      228      clc
CCA1      229 ;
CCA1 60      230      rts              ; return with no error
CCA2      231 ;
CCA2 BD 89 C0   232 ^1      lda ATAERROR,X      ; get CF error
CCA5 99 78 04   233      sta ERRSTAT,Y      ; save status
CCA8      234 ;
CCA8 A9 03      235      lda #STATERR      ; get STATUS error
CCAA      236 ;
CCAA 38      237      sec
CCAB      238 ;
CCAB 60      239      rts              ; return with error
CCAC      240 ;
CCAC      241 ;
CCAC      242 ; CF software reset. Bit D2 must be asserted for at least
CCAC      243 ; 25 usec. Bit D1 is left asserted to disable interrupts.

```

```

CCAC      244 ; Fall into CFWAIT.
CCAC      245 ;
CCAC      246 ; Device Control register:
CCAC      247 ;
CCAC      248 ; |   D7   |   D6   |   D5   |   D4   |   D3   |   D2   |   D1   |   D0   |
CCAC      249 ; |   ----   |   ----   |   ----   |   ----   |   ----   |   ----   |   ----   |   ----   |
CCAC      250 ; |   X   |   X   |   X   |   X   |   1   | SW Rst | nIEN |   0   |
CCAC      251 ;
CCAC A9 0E 252 DOATARST lda #%00001110 ; 1,SRST,nIEN,0
CCAE 9D 86 C0 253 sta ATADEVCT,X ; set device
CCB1      254 ;
CCB1 A9 04 255 lda #WAIT100U ; wait for 100 usec
CCB3 20 BD CC 256 jsr CFWAIT
CCB6      257 ;
CCB6 A9 0A 258 lda #%00001010 ; nIEN=1, disable interrupts
CCB8 9D 86 C0 259 sta ATADEVCT,X ; set device
CCBB      260 ;
CCBB A9 7C 261 lda #WAIT040M ; wait for 40 msec
CCBD      262 ;
CCBD      263 ; jmp CFWAIT ; and return to caller
CCBD      264 ;
CCBD      265 ;
CCBD      266 ; Delay routine based on A-reg value.
CCBD      267 ;
CCBD      268 ; Delay(usec) = ( 5*A^2 + 27*A + 26 ) / 2
CCBD      269 ;
CCBD      270 ; A = ( Delay(usec) / 2.5 + 2.09 ) ^ 0.5 - 2.7
CCBD      271 ;
CCBD      272 ; Same wait routine as in monitor.
CCBD      273 ;
CCBD 38 274 CFWAIT sec ; set for subtraction
CCBE      275 ;
CCBE 48 276 ^1 pha ; save value
CCBF      277 ;
CCBF E9 01 278 ^2 sbc #1 ; inner loop subtraction
CCC1 D0 FC 279 bne <2 ; not zero if branch
CCC3      280 ;
CCC3 68 281 pla ; recall value
CCC4      282 ;
CCC4 E9 01 283 sbc #1 ; outer loop subtraction
CCC6 D0 F6 284 bne <1 ; not zero if branch
CCC8      285 ;
CCC8 60 286 rts ; return to caller
CCC9      287 ;
CCC9      288 ;
CCC9 20 55 CC 289 CFCHECK jsr CFBUSY ; wait for BUSY=0
CCCC B0 05 290 bcs >1 ; error if branch
CCCE      291 ;
CCCE 20 8E CC 292 jsr CFSTATUS ; get ERR and DRQ state
CCD1 90 02 293 bcc >2 ; ERR=0 if branch
CCD3      294 ;
CCD3 09 0C 295 ^1 ora #CHKERR ; build error path
CCD5      296 ;
CCD5 60 297 ^2 rts ; return to caller
CCD6      298 ;
CCD6      299 ;
CCD6      300 ; Load the CF command register. Processing must complete
CCD6      301 ; within 20 msec. Must wait at least 400 nsec before
CCD6      302 ; reading status for the first time. From CFSTATUS, want
CCD6      303 ; C-flag=0 (ERR=0) and A-reg>0 (DRQ=1).
CCD6      304 ;

```

```
CCD6 9D 8F C0    305  LOADCMD  sta ATACMD,X
CCD9              306  ;
CCD9 20 55 CC    307              jsr CFBUSY          ; wait for BUSY=0
CCDC B0 11       308              bcs >3            ; error if branch
CCDE              309  ;
CCDE A0 C8       310              ldy #200           ; get loop count
CCE0              311  ;
CCE0 A9 04       312  ^1      lda #WAIT100U         ; wait 100 usec
CCE2 20 BD CC    313              jsr CFWAIT
CCE5              314  ;
CCE5 20 8E CC    315              jsr CFSTATUS        ; get ERR and DRQ state
CCE8 B0 02       316              bcs >2            ; ERR=1 if branch
CCEA              317  ;
CCEA D0 05       318              bne >4            ; DRQ=1 if branch
CCEC              319  ;
CCEC 88          320  ^2      dey
CCED D0 F1       321              bne <1            ; not zero if branch
CCEF              322  ;
CCEF 09 08       323  ^3      ora #CMDERR          ; build error path
CCF1              324  ;
CCF1 60          325  ^4      rts                  ; return to caller
CCF2              326  ;
CCF2              327  ;
CCF2              328      icl "ROM4.L"
```

LLOAD ROM4.L,A\$4000

```

CCF2      1          ttl "CCFA Firmware Source Code, ROM4.L"
CCF2      2      ;
CCF2      3      ;
CCF2      4      ; ROM4.L
CCF2      5      ;
CCF2      6      ;
CCF2      7      ; Load the CCFA registers with an LBA. Return error
CCF2      8      ; value if CF is BUSY.
CCF2      9      ;
CCF2 AE F8 04 10 LOADLBA  ldx CFSLOT16          ; get CF SLOT*16
CCF5      11      ;
CCF5 20 55 CC 12          jsr CFBUSY          ; BUSY bit must be clear
CCF8 B0 22   13          bcs >1              ; error if branch
CCFA      14      ;
CCFA A9 01   15          lda #1              ; set for one block I/O
CCFC 9D 8A C0 16          sta ATASECCT,X
CCFF      17      ;
CCFF AD 78 05 18          lda LBA0            ; load LBA0
CD02 9D 8B C0 19          sta ATASECTR,X
CD05      20      ;
CD05 AD F8 05 21          lda LBA1            ; load LBA1
CD08 9D 8C C0 22          sta ATACYLNL,X
CD0B      23      ;
CD0B AD 78 06 24          lda LBA2            ; load LBA2
CD0E 9D 8D C0 25          sta ATACYLNH,X
CD11      26      ;
CD11 AD F8 06 27          lda LBA3            ; load LBA3
CD14 29 0F   28          and #LBAMASK        ; mask bits 27:24
CD16      29      ;
CD16 09 E0   30          ora #%11100000      ; LBA=1, DRV=0
CD18 9D 8E C0 31          sta ATAHEAD,X
CD1B      32      ;
CD1B 60      33          rts                  ; return with no error
CD1C      34      ;
CD1C 09 04   35 ^1      ora #LBAERR          ; build error path
CD1E      36      ;
CD1E 60      37          rts                  ; return with error
CD1F      38      ;
CD1F      39      ;
CD1F      40      ; Reset CF. Wait for BUSY=0, but continue even if error.
CD1F      41      ; Wait for RDY=1, do reset, then test for ERR=0.
CD1F      42      ;
CD1F AE F8 04 43 CFRESET  ldx CFSLOT16          ; get CF SLOT*16
CD22      44      ;
CD22 20 55 CC 45          jsr CFBUSY          ; wait for BUSY=0
CD25      46      ;
CD25 20 6D CC 47          jsr CFREADY         ; wait for RDY=1
CD28 B0 08   48          bcs >1              ; error if branch
CD2A      49      ;
CD2A 20 AC CC 50          jsr DOATARST        ; do reset
CD2D      51      ;
CD2D 20 C9 CC 52          jsr CFCHECK         ; get ERR and DRQ state
CD30 90 02   53          bcc >2              ; ERR=0 if branch
CD32      54      ;
CD32 09 80   55 ^1      ora #RSTERR          ; build error path
CD34      56      ;
CD34 60      57 ^2      rts                  ; return to caller
CD35      58      ;
CD35      59      ;
CD35      60      ; Process CF command: recognize command, load LBA, and

```

```

CD35      61 ; transfer data.
CD35      62 ;
CD35      63 ; Command value is invalid.
CD35      64 ;
CD35 A9 F0 65 ^1      lda #UNKERR      ; command not recognized
CD37      66 ;
CD37 38    67      sec
CD38      68 ;
CD38 60    69      rts      ; return to caller
CD39      70 ;
CD39      71 ;
CD39 AC 78 04 72 DOFCMD  ldy CFSLOT      ; get CF slot index
CD3C      73 ;
CD3C A9 00 74      lda #ZERO
CD3E 85 3E 75      sta GENPTR      ; initialize pointer
CD40      76 ;
CD40 B9 F8 07 77      lda DOSNBUF1,Y    ; get nibble buffer 1 page
CD43 85 3F 78      sta GENPTR+1      ; save address
CD45      79 ;
CD45 A9 FF 80      lda #NEGONE      ; assume error value
CD47 99 78 04 81      sta ERRSTAT,Y    ; initialize to error
CD4A      82 ;
CD4A A5 3C 83      lda COMMANDZ      ; get command
CD4C      84 ;
CD4C      85 ;      cmp #SEEKCMD      ; check for load LBA
CD4C F0 A4 86      beq LOADLBA
CD4E      87 ;
CD4E C9 20 88      cmp #RSETCMD      ; check for CF Reset
CD50 F0 CD 89      beq CFRESET
CD52      90 ;
CD52 C9 01 91      cmp #READCMD      ; check for read 1 page
CD54 F0 67 92      beq RD1PAGE
CD56      93 ;
CD56 C9 02 94      cmp #WRITCMD      ; check for write 1 page
CD58 F0 7F 95      beq WR1PAGE
CD5A      96 ;
CD5A C9 11 97      cmp #READCMD2     ; check for read 2 pages
CD5C F0 3E 98      beq RD2PAGE
CD5E      99 ;
CD5E C9 12 100     cmp #WRITCMD2     ; check for write 2 pages
CD60 F0 49 101     beq WR2PAGE
CD62      102 ;
CD62 C9 21 103     cmp #RDLBACMD     ; check for read 2 pages
CD64 F0 36 104     beq RD2PAGE
CD66      105 ;
CD66 C9 22 106     cmp #WRLBACMD     ; check for write 2 pages
CD68 F0 41 107     beq WR2PAGE
CD6A      108 ;
CD6A C9 30 109     cmp #RDIDCMD      ; check for read ID
CD6C F0 07 110     beq READID
CD6E      111 ;
CD6E C9 04 112     cmp #FMTCMD      ; check for format volume
CD70 D0 C3 113     bne <1
CD72      114 ;
CD72 4C 25 CE 115     jmp FMTVOL
CD75      116 ;
CD75      117 ;
CD75      118 ; Read Identify Device to selected buffer. First set the
CD75      119 ; LBA to zero, then load the LBA and CF command, and read
CD75      120 ; the data.
CD75      121 ;

```

```

CD75 A9 00      122 READID   lda #ZERO                ; set LBA=0
CD77 8D F8 06   123          sta LBA3
CD7A 8D 78 06   124          sta LBA2
CD7D 8D F8 05   125          sta LBA1
CD80 8D 78 05   126          sta LBA0
CD83           127 ;
CD83 20 F2 CC   128          jsr LOADLBA                ; load LBA
CD86 B0 10      129          bcs >1                    ; error if branch
CD88           130 ;
CD88 A9 EC      131          lda #ATA_ID                ; set CF to read ID
CD8A           132 ;
CD8A 20 D6 CC   133          jsr LOADCMD                ; load command
CD8D B0 09      134          bcs >1                    ; error if branch
CD8F           135 ;
CD8F BD 82 C0   136          lda CLRCMSK,X              ; enable 6502 pre-fetch
CD92           137 ;
CD92 20 93 CE   138          jsr CF2BUFR                ; read lower page
CD95 20 93 CE   139          jsr CF2BUFR                ; read upper page
CD98           140 ;
CD98 A0 C0      141 ^1      ldy #IDERR                  ; READID error
CD9A D0 79      142          bne GETSTAT                ; always taken
CD9C           143 ;
CD9C           144 ;
CD9C           145 ; Read both lower and upper page.
CD9C           146 ;
CD9C 20 73 CE   147 RD2PAGE jsr SET2READ                ; set CF to read
CD9F B0 06      148          bcs >1                    ; error if branch
CDA1           149 ;
CDA1 20 93 CE   150          jsr CF2BUFR                ; read lower page
CDA4 20 93 CE   151          jsr CF2BUFR                ; read upper page
CDA7           152 ;
CDA7 A0 D0      153 ^1      ldy #RD2ERR                  ; RD2PAGE error
CDA9 D0 6A      154          bne GETSTAT                ; always taken
CDAB           155 ;
CDAB           156 ;
CDAB           157 ; Write to both lower and upper page.
CDAB           158 ;
CDAB 20 83 CE   159 WR2PAGE jsr SET2WRIT                ; set CF to write
CDAE B0 09      160          bcs >1                    ; error if branch
CDB0           161 ;
CDB0 20 A6 CE   162          jsr BUFR2CF                  ; write buffer
CDB3 20 A6 CE   163          jsr BUFR2CF                  ; write buffer
CDB6           164 ;
CDB6 BD 82 C0   165          lda CLRCMSK,X              ; enable 6502 pre-fetch
CDB9           166 ;
CDB9 A0 E0      167 ^1      ldy #WR2ERR                  ; WR2PAGE error
CDBB D0 58      168          bne GETSTAT                ; always taken
CDBD           169 ;
CDBD           170 ;
CDBD           171 ; Read selected page and toss the other page.
CDBD           172 ;
CDBD 20 73 CE   173 RD1PAGE jsr SET2READ                ; set CF to read
CDC0 B0 13      174          bcs >2                    ; error if branch
CDC2           175 ;
CDC2 AD 78 07   176          lda PAGE                    ; select page
CDC5 D0 08      177          bne >1                    ; upper page if branch
CDC7           178 ;
CDC7           179 ;
CDC7           180 ; Read lower page and toss upper page.
CDC7           181 ;
CDC7 20 93 CE   182          jsr CF2BUFR                ; read lower page

```



```

CDCA          183 ;
CDCA 20 DF CE 184      jsr TOSS          ; toss upper page
CDCD F0 06    185      beq >2          ; always taken
CDCF          186 ;
CDCF          187 ;
CDCF          188 ; Toss lower page and read upper page.
CDCF          189 ;
CDCF 20 DF CE 190      ^1      jsr TOSS          ; toss lower page
CDD2          191 ;
CDD2 20 93 CE 192      jsr CF2BUFR       ; read upper page
CDD5          193 ;
CDD5 A0 90    194      ^2      ldy #RD1ERR      ; RD1PAGE error
CDD7 D0 3C    195      bne GETSTAT      ; always taken
CDD9          196 ;
CDD9          197 ;
CDD9          198 ; Save the opposite page and write the selected and the
CDD9          199 ; saved page. Fall into GETSTAT.
CDD9          200 ;
CDD9 20 73 CE 201      WR1PAGE jsr SET2READ      ; set CF to read
CDDC B0 35    202      bcs >5          ; error if branch
CDDE          203 ;
CDDE AD 78 07 204      lda PAGE          ; select page
CDE1 D0 08    205      bne >1          ; upper page if branch
CDE3          206 ;
CDE3          207 ;
CDE3          208 ; Toss lower page and save upper page.
CDE3          209 ;
CDE3 20 DF CE 210      jsr TOSS          ; toss lower page
CDE6          211 ;
CDE6 20 BB CE 212      jsr CF2TEMP       ; save upper page
CDE9 F0 06    213      beq >2          ; always taken
CDEB          214 ;
CDEB          215 ;
CDEB          216 ; Save lower page and toss upper page.
CDEB          217 ;
CDEB 20 BB CE 218      ^1      jsr CF2TEMP       ; save lower page
CDEE          219 ;
CDEE 20 DF CE 220      jsr TOSS          ; toss upper page
CDF1          221 ;
CDF1          222 ;
CDF1          223 ; Write the data.
CDF1          224 ;
CDF1 A0 A0    225      ^2      ldy #WR1ERR      ; WR1PAGE error
CDF3          226 ;
CDF3 20 15 CE 227      jsr GETSTAT      ; check status
CDF6 B0 2C    228      bcs CFRTN        ; error if branch
CDF8          229 ;
CDF8 20 83 CE 230      jsr SET2WRIT      ; set CF to write
CDFB B0 16    231      bcs >5          ; error if branch
CDFD          232 ;
CDFD AD 78 07 233      lda PAGE          ; select page
CE00 D0 08    234      bne >3          ; upper page if branch
CE02          235 ;
CE02          236 ;
CE02          237 ; Write buffer and saved data.
CE02          238 ;
CE02 20 A6 CE 239      jsr BUFR2CF       ; write buffer
CE05          240 ;
CE05 20 CC CE 241      jsr TEMP2CF       ; write saved data
CE08 F0 06    242      beq >4          ; always taken
CE0A          243 ;

```

```

CE0A      244 ;
CE0A      245 ; Write saved data and buffer.
CE0A      246 ;
CE0A 20 CC CE 247 ^3      jsr TEMP2CF      ; write saved data
CE0D      248 ;
CE0D 20 A6 CE 249      jsr BUFR2CF      ; write buffer
CE10      250 ;
CE10 BD 82 C0 251 ^4      lda CLRCMSK,X    ; enable 6502 pre-fetch
CE13      252 ;
CE13 A0 A0    253 ^5      ldy #WR1ERR      ; WR1PAGE error
CE15      254 ;
CE15      255 ;
CE15      256 ; Check status. If C-flag is set build the error path.
CE15      257 ; From CFSTATUS, want C-flag=0 (ERR=0) and A-reg=0 (DRQ=0).
CE15      258 ;
CE15 8C 14 01 259 GETSTAT sty TEMPERR      ; save command error
CE18      260 ;
CE18 B0 07    261      bcs >1      ; error if branch
CE1A      262 ;
CE1A 20 C9 CC 263      jsr CFCHECK      ; get ERR and DRQ state
CE1D B0 02    264      bcs >1      ; error if branch
CE1F      265 ;
CE1F F0 03    266      beq CFRTN      ; DRQ=0 if branch
CE21      267 ;
CE21 0D 14 01 268 ^1      ora TEMPERR      ; build error path
CE24      269 ;
CE24 60      270 CFRTN      rts      ; return to caller
CE25      271 ;
CE25      272 ;
CE25      273 ; Format the volume specified by drive and volume number.
CE25      274 ; Clear A-reg before calling DVTS2LBA.
CE25      275 ;
CE25 A9 03    276 FMTVOL      lda /VOLSIZE      ; get MSB of volume size
CE27 85 3F    277      sta GENPTR+1      ; set for block counter
CE29      278 ;
CE29 A9 00    279      lda #VOLSIZE      ; get LSB of volume size
CE2B 85 3E    280      sta GENPTR      ; clear for block counter
CE2D      281 ;
CE2D 85 2D    282      sta SECTORZ      ; clear sector number
CE2F 85 2E    283      sta TRACKZ      ; clear track number
CE31      284 ;
CE31 20 DE CB 285      jsr DVTS2LBA      ; get start of volume LBA
CE34 20 30 CC 286      jsr CHKCFLBA      ; range check the LBA
CE37      287 ;
CE37 A9 B0    288      lda #FMTERR      ; assume FMTVOL error
CE39      289 ;
CE39 B0 37    290      bcs >4      ; error if branch
CE3B      291 ;
CE3B 20 83 CE 292 ^1      jsr SET2WRIT      ; set CF to write
CE3E 09 B0    293      ora #FMTERR      ; assume FMTVOL error
CE40      294 ;
CE40 B0 30    295      bcs >4      ; error if branch
CE42      296 ;
CE42      297 ;
CE42      298 ; Initialize entire block (512 bytes) to zero.
CE42      299 ;
CE42 A0 00    300      ldy #ZERO
CE44 98      301      tya      ; set data to zero
CE45      302 ;
CE45 9D 80 C0 303 ^2      sta ATADATAH,X    ; write data
CE48 9D 88 C0 304      sta ATADATAL,X    ; write data

```

```

CE4B      305 ;
CE4B C8    306      iny
CE4C D0 F7  307      bne <2
CE4E      308 ;
CE4E BD 82 C0 309      lda CLRCMSK,X      ; enable 6502 pre-fetch
CE51      310 ;
CE51 A0 B0   311      ldy #FMTERR      ; FMTVOL error
CE53      312 ;
CE53 20 15 CE 313      jsr GETSTAT      ; check status
CE56 B0 1A    314      bcs >4      ; error if branch
CE58      315 ;
CE58      316 ;
CE58      317 ; Increment LBA and decrement counter until counter becomes
CE58      318 ; zero.
CE58      319 ;
CE58 EE 78 05 320      inc LBA0      ; next LBA0
CE5B D0 0D    321      bne >3
CE5D      322 ;
CE5D EE F8 05 323      inc LBA1      ; next LBA1
CE60 D0 08    324      bne >3
CE62      325 ;
CE62 EE 78 06 326      inc LBA2      ; next LBA2
CE65 D0 03    327      bne >3
CE67      328 ;
CE67 EE F8 06 329      inc LBA3      ; next LBA3
CE6A      330 ;
CE6A C6 3E    331      ^3      dec GENPTR      ; decrement LSB counter
CE6C D0 CD    332      bne <1      ; more to do if branch
CE6E      333 ;
CE6E C6 3F    334      dec GENPTR+1      ; decrement MSB counter
CE70 D0 C9    335      bne <1      ; more to do if branch
CE72      336 ;
CE72 60      337      ^4      rts      ; return to caller
CE73      338 ;
CE73      339 ;
CE73      340 ; Configure the CF to read the selected LBA.
CE73      341 ;
CE73 20 F2 CC 342 SET2READ jsr LOADLBA      ; load LBA
CE76 B0 0A    343      bcs >1      ; error if branch
CE78      344 ;
CE78 A9 20    345      lda #ATA_READ      ; set CF to read
CE7A      346 ;
CE7A 20 D6 CC 347      jsr LOADCMD      ; load command
CE7D B0 03    348      bcs >1      ; error if branch
CE7F      349 ;
CE7F BD 82 C0 350      lda CLRCMSK,X      ; enable 6502 pre-fetch
CE82      351 ;
CE82 60      352      ^1      rts      ; return to caller
CE83      353 ;
CE83      354 ;
CE83      355 ; Configure the CF to write the selected LBA.
CE83      356 ;
CE83 20 F2 CC 357 SET2WRIT jsr LOADLBA      ; load LBA
CE86 B0 0A    358      bcs >1      ; error if branch
CE88      359 ;
CE88 A9 30    360      lda #ATA_WRIT      ; set CF to write
CE8A      361 ;
CE8A 20 D6 CC 362      jsr LOADCMD      ; load command
CE8D B0 03    363      bcs >1      ; error if branch
CE8F      364 ;
CE8F BD 81 C0 365      lda SETCMSK,X      ; disable 6502 pre-fetch

```

```

CE92          366 ;
CE92 60        367 ^1      rts          ; return to caller
CE93          368 ;
CE93          369 ;
CE93          370 ; Read CF page to buffer.
CE93          371 ;
CE93 A0 00     372 CF2BUFR ldy #ZERO      ; get loop count
CE95          373 ;
CE95 BD 88 C0  374 ^1      lda ATADATAL,X    ; read data
CE98 91 26     375          sta (BUFRADRZ),Y  ; save data
CE9A          376 ;
CE9A C8        377          iny
CE9B          378 ;
CE9B BD 80 C0  379          lda ATADATAH,X    ; read data
CE9E 91 26     380          sta (BUFRADRZ),Y  ; save data
CEA0          381 ;
CEA0 C8        382          iny
CEA1 D0 F2     383          bne <1
CEA3          384 ;
CEA3 E6 27     385          inc BUFRADRZ+1      ; next page
CEA5          386 ;
CEA5 60        387          rts          ; return to caller
CEA6          388 ;
CEA6          389 ;
CEA6          390 ; Write buffer to CF page.
CEA6          391 ;
CEA6 A0 00     392 BUFR2CF ldy #ZERO      ; get loop count
CEA8          393 ;
CEA8 B1 26     394 ^1      lda (BUFRADRZ),Y    ; read data
CEAA 48        395          pha          ; save
CEAB          396 ;
CEAB C8        397          iny
CEAC          398 ;
CEAC B1 26     399          lda (BUFRADRZ),Y    ; read data
CEAE 9D 80 C0  400          sta ATADATAH,X    ; write data
CEB1          401 ;
CEB1 68        402          pla          ; recall data
CEB2 9D 88 C0  403          sta ATADATAL,X    ; write data
CEB5          404 ;
CEB5 C8        405          iny
CEB6 D0 F0     406          bne <1
CEB8          407 ;
CEB8 E6 27     408          inc BUFRADRZ+1      ; next page
CEBA          409 ;
CEBA 60        410          rts          ; return to caller
CEBB          411 ;
CEBB          412 ;
CEBB          413 ; Read CF page to nibble buffer.
CEBB          414 ;
CEBB A0 00     415 CF2TEMP ldy #ZERO      ; get loop count
CEBD          416 ;
CEBD BD 88 C0  417 ^1      lda ATADATAL,X    ; read data
CEC0 91 3E     418          sta (GENPTR),Y    ; save data
CEC2          419 ;
CEC2 C8        420          iny
CEC3          421 ;
CEC3 BD 80 C0  422          lda ATADATAH,X    ; read data
CEC6 91 3E     423          sta (GENPTR),Y    ; save data
CEC8          424 ;
CEC8 C8        425          iny
CEC9 D0 F2     426          bne <1

```

```
CECB          427 ;
CECB 60        428             rts             ; return to caller
CECC          429 ;
CECC          430 ;
CECC          431 ; Write nibble buffer to CF page.
CECC          432 ;
CECC A0 00     433 TEMP2CF ldy #ZERO             ; get loop count
CECE          434 ;
CECE B1 3E     435 ^1      lda (GENPTR),Y         ; read data
CED0 48        436             pha             ; save
CED1          437 ;
CED1 C8        438             iny
CED2          439 ;
CED2 B1 3E     440             lda (GENPTR),Y         ; read data
CED4 9D 80 C0  441             sta ATADATAH,X         ; write data
CED7          442 ;
CED7 68        443             pla             ; recall data
CED8 9D 88 C0  444             sta ATADATAL,X         ; write data
CEDB          445 ;
CEDB C8        446             iny
CEDC D0 F0     447             bne <1
CEDE          448 ;
CEDE 60        449             rts             ; return to caller
CEDF          450 ;
CEDF          451 ;
CEDF          452 ; Toss the data of the page that comes before or after the
CEDF          453 ; selected page of data that was or will be read or
CEDF          454 ; written.
CEDF          455 ;
CEDF A0 80     456 TOSS      ldy #PAGESIZE/2         ; get loop count
CEE1          457 ;
CEE1 BD 88 C0  458 ^1      lda ATADATAL,X         ; read data, toss
CEE4 BD 80 C0  459             lda ATADATAH,X         ; read data, toss
CEE7          460 ;
CEE7 C8        461             iny
CEE8 D0 F7     462             bne <1
CEEA          463 ;
CEEA 60        464             rts             ; return to caller
CEEB          465 ;
CEEB          466 ;
CEEB          467             icl "ROM5.L"
```

LLOAD ROM5.L,A\$4000

```

CEEB      1          ttl "CFFA Firmware Source Code, ROM5.L"
CEEB      2      ;
CEEB      3      ;
CEEB      4      ; ROM5.L
CEEB      5      ;
CEEB      6      ;
CEEB      7      ; Reverse Interleave Table for booting a drive volume.
CEEB      8      ;
CEEB 00 07 0E      9  RVSELEAV hex 00070E060D050C04
CEEE 06 0D 05
CEF1 0C 04
CEF3 0B 03 0A      10          hex 0B030A020901080F
CEF6 02 09 01
CEF9 08 0F
CEFB      11      ;
CEFB      12      ;
CEFB 10      13  IMAGLBA  byt OFFSET0+IMAGSIZE*0
CEFC 28      14          byt OFFSET0+IMAGSIZE*1
CEFD 40      15          byt OFFSET0+IMAGSIZE*2
CEFE 58      16          byt OFFSET0+IMAGSIZE*3
CEFF 70      17          byt OFFSET0+IMAGSIZE*4
CF00 88      18          byt OFFSET0+IMAGSIZE*5
CF01 A0      19          byt OFFSET0+IMAGSIZE*6
CF02 B8      20          byt OFFSET0+IMAGSIZE*7
CF03      21      ;
CF03      22      ;
CF03      23      ; Table lookup and simple addition provides the fastest
CF03      24      ; method to calculate a unique LBA from the parameters
CF03      25      ; sector, track, volume, and drive.
CF03      26      ;
CF03      27      ; Volume needs to be multiplied by 0x300 and drive needs
CF03      28      ; to be multiplied by 0x30000. These tables provide
CF03      29      ; this multiplication conveniently for addition.
CF03      30      ;
CF03      31      ; Each parameter is separated into its two nibbles and
CF03      32      ; they are moved into the X and Y registers. Because a
CF03      33      ; nibble is just 4 bits, each table is just 16 bytes.
CF03      34      ;
CF03      35      ; Volume and drive share the first two tables since drive
CF03      36      ; does not and will not exceed 256/3.
CF03      37      ;
CF03      38  DRVXTBL:
CF03      39  VOLXTBL:
CF03 00      40          byt VOLXINC*00
CF04 03      41          byt VOLXINC*01
CF05 06      42          byt VOLXINC*02
CF06 09      43          byt VOLXINC*03
CF07 0C      44          byt VOLXINC*04
CF08 0F      45          byt VOLXINC*05
CF09 12      46          byt VOLXINC*06
CF0A 15      47          byt VOLXINC*07
CF0B 18      48          byt VOLXINC*08
CF0C 1B      49          byt VOLXINC*09
CF0D 1E      50          byt VOLXINC*10
CF0E 21      51          byt VOLXINC*11
CF0F 24      52          byt VOLXINC*12
CF10 27      53          byt VOLXINC*13
CF11 2A      54          byt VOLXINC*14
CF12 2D      55          byt VOLXINC*15
CF13      56      ;

```

```

CF13      57 ;
CF13      58 DRVYTBLL:
CF13      59 VOLYTBLL:
CF13 00    60      byt VOLYINC*00
CF14 30    61      byt VOLYINC*01
CF15 60    62      byt VOLYINC*02
CF16 90    63      byt VOLYINC*03
CF17 C0    64      byt VOLYINC*04
CF18 F0    65      byt VOLYINC*05
CF19 20    66      byt VOLYINC*06
CF1A 50    67      byt VOLYINC*07
CF1B 80    68      byt VOLYINC*08
CF1C B0    69      byt VOLYINC*09
CF1D E0    70      byt VOLYINC*10
CF1E 10    71      byt VOLYINC*11
CF1F 40    72      byt VOLYINC*12
CF20 70    73      byt VOLYINC*13
CF21 A0    74      byt VOLYINC*14
CF22 D0    75      byt VOLYINC*15
CF23      76 ;
CF23      77 DRVYTBLLH:
CF23      78 VOLYTBLLH:
CF23 00    79      hby VOLYINC*00
CF24 00    80      hby VOLYINC*01
CF25 00    81      hby VOLYINC*02
CF26 00    82      hby VOLYINC*03
CF27 00    83      hby VOLYINC*04
CF28 00    84      hby VOLYINC*05
CF29 01    85      hby VOLYINC*06
CF2A 01    86      hby VOLYINC*07
CF2B 01    87      hby VOLYINC*08
CF2C 01    88      hby VOLYINC*09
CF2D 01    89      hby VOLYINC*10
CF2E 02    90      hby VOLYINC*11
CF2F 02    91      hby VOLYINC*12
CF30 02    92      hby VOLYINC*13
CF31 02    93      hby VOLYINC*14
CF32 02    94      hby VOLYINC*15
CF33      95 ;
CF33      96 ;
CF33      97 MSGS:
CF33      98 ;
CF33 2E    99 MSG1A   asc  `.`
CF34      100 ;
CF34 0D 8D 101 MSG1B   hex  0D8D
CF36      102 ;
CF36      103 ;
CF36      104 ; Information message words.
CF36      105 ;
CF36 55 6E 61 106 MSG2A   dci  `Unable to`
CF39 62 6C 65
CF3C 20 74 EF
CF3F      107 ;
CF3F 41 62 6C 108 MSG2B   dci  `Able to`
CF42 65 20 74
CF45 EF
CF46      109 ;
CF46 64 69 73 110 MSG3A   asc  `dis`
CF49      111 ;
CF49 63 6F 6E 112 MSG3B   dci  `connect CF and DOS`
CF4C 6E 65 63

```

```

CF4F 74 20 43
CF52 46 20 61
CF55 6E 64 20
CF58 44 4F D3
CF5B          113 ;
CF5B          114 ;
CF5B          115 ; DOS versions.
CF5B          116 ;
CF5B 33 2E B3 117 MSG4A    dci `3.3`
CF5E          118 ;
CF5E 34 2E B1 119 MSG4B    dci `4.1`
CF61          120 ;
CF61 34 2E B3 121 MSG4C    dci `4.3`
CF64          122 ;
CF64 34 2E B5 123 MSG4D    dci `4.5`
CF67          124 ;
CF67          125 ;
CF67          126 ; Error message words.
CF67          127 ;
CF67 43 C6    128 MSG5A    dci `CF`
CF69          129 ;
CF69 72 65 73 130 MSG5B    dci `reset`
CF6C 65 F4
CF6E          131 ;
CF6E 49 44 20 132 MSG5C    asc `ID `
CF71          133 ;
CF71 65 72 72 134 MSG5D    dci `error`
CF74 6F F2
CF76          135 ;
CF76 57 72 6F 136 MSG5E    dci `Wrong`
CF79 6E E7
CF7B          137 ;
CF7B 63 61 72 138 MSG5F    dci `card`
CF7E E4
CF7F          139 ;
CF7F 62 6F 6F 140 MSG5G    dci `boot image`
CF82 74 20 69
CF85 6D 61 67
CF88 E5
CF89          141 ;
CF89 64 72 69 142 MSG5H    dci `drive`
CF8C 76 E5
CF8E          143 ;
CF8E 4C 42 C1 144 MSG5I    dci `LBA`
CF91          145 ;
CF91 72 61 6E 146 MSG5J    dci `range`
CF94 67 E5
CF96          147 ;
CF96 72 65 61 148 MSG5K    dci `read`
CF99 E4
CF9A          149 ;
CF9A 76 61 6C 150 MSG5L    dci `value`
CF9D 75 E5
CF9F          151 ;
CF9F          152 ;
CF9F          153 TEXTS:
CF9F          154 ;
CF9F          155 ; Information message texts.
CF9F          156 ;
CF9F          157 ; Unable to disconnect CF and DOS.
CF9F 03 13 00 158 IOTEXT1A byt MSG2A-MESGS,MESG3A-MESGS,ZERO

```



```

CFA2      159 ;
CFA2      160 ; Unable to connect CF and DOS.
CFA2 03 16 00 161 IOTEXT1B byt MSG2A-MESGS,MSG3B-MESGS,ZERO
CFA5      162 ;
CFA5      163 ;
CFA5      164 ; Disconnect message texts.
CFA5      165 ;
CFA5      166 ; Able to disconnect CF and DOS 3.3.
CFA5 0C 13 28 167 IOTEXT2A byt MSG2B-MESGS,MSG3A-MESGS,MSG4A-MESGS
CFA8      168 ;
CFA8      169 ; Able to disconnect CF and DOS 4.1.
CFA8 0C 13 2B 170 IOTEXT2B byt MSG2B-MESGS,MSG3A-MESGS,MSG4B-MESGS
CFAB      171 ;
CFAB      172 ; Able to disconnect CF and DOS 4.3.
CFAB 0C 13 2E 173 IOTEXT2C byt MSG2B-MESGS,MSG3A-MESGS,MSG4C-MESGS
CFAE      174 ;
CFAE      175 ; Able to disconnect CF and DOS 4.5.
CFAE 0C 13 31 176 IOTEXT2D byt MSG2B-MESGS,MSG3A-MESGS,MSG4D-MESGS
CFB1      177 ;
CFB1      178 ;
CFB1      179 ; Connect message texts.
CFB1      180 ;
CFB1      181 ; Able to connect CF and DOS 3.3.
CFB1 0C 16 28 182 IOTEXT3A byt MSG2B-MESGS,MSG3B-MESGS,MSG4A-MESGS
CFB4      183 ;
CFB4      184 ; Able to connect CF and DOS 4.1.
CFB4 0C 16 2B 185 IOTEXT3B byt MSG2B-MESGS,MSG3B-MESGS,MSG4B-MESGS
CFB7      186 ;
CFB7      187 ; Able to connect CF and DOS 4.3.
CFB7 0C 16 2E 188 IOTEXT3C byt MSG2B-MESGS,MSG3B-MESGS,MSG4C-MESGS
CFBA      189 ;
CFBA      190 ; Able to connect CF and DOS 4.5.
CFBA 0C 16 31 191 IOTEXT3D byt MSG2B-MESGS,MSG3B-MESGS,MSG4D-MESGS
CFBD      192 ;
CFBD      193 ;
CFBD      194 ; Error message texts.
CFBD      195 ;
CFBD      196 ; CF reset error.
CFBD 34 36 3E 197 ERRTEXT1 byt MSG5A-MESGS,MSG5B-MESGS,MSG5D-MESGS
CFC0      198 ;
CFC0      199 ; CF read ID error.
CFC0 34 63 3B 200 ERRTEXT2 byt MSG5A-MESGS,MSG5K-MESGS,MSG5C-MESGS
CFC3      201 ;
CFC3      202 ; Wrong CF card.
CFC3 43 34 48 203 ERRTEXT3 byt MSG5E-MESGS,MSG5A-MESGS,MSG5F-MESGS
CFC6      204 ;
CFC6      205 ; Wrong boot image value.
CFC6 43 4C 67 206 ERRTEXT4 byt MSG5E-MESGS,MSG5G-MESGS,MSG5L-MESGS
CFC9      207 ;
CFC9      208 ; Wrong drive value.
CFC9 43 56 67 209 ERRTEXT5 byt MSG5E-MESGS,MSG5H-MESGS,MSG5L-MESGS
CFCC      210 ;
CFCC      211 ; LBA range error.
CFCC 5B 5E 3E 212 ERRTEXT6 byt MSG5I-MESGS,MSG5J-MESGS,MSG5D-MESGS
CFCF      213 ;
CFCF      214 ; LBA read error.
CFCF 5B 63 3E 215 ERRTEXT7 byt MSG5I-MESGS,MSG5K-MESGS,MSG5D-MESGS
CFD2      216 ;
CFD2      217 ;
CFD2      218 ; Note: accessing 0xCFEF turns the CFFA card off.
CFD2      219 ;

```

```
CFD2      220 ;
CFD2      221      dfs PAGED0-*,NEGONE
D000      222 ;
D000      223 ;
D000      224 ; Copyright (c) 2023 April 04
D000      225 ;      by
D000      226 ;      Walland Philip Vrbancic Jr
D000      227 ;
D000      228 ;      All Rights Reserved
D000      229 ;
D000      230 ;
```

```
BSAVE CFFA_ROM_BUILD56,A$0900,B,L$0800
```

```
D000      231      usr CFFA_ROM_BUILD56
D000      232 ;
D000      233 ;
D000      234      stt "CFFA Symbol Table"
D000      235 ;
D000      236 ;
D000      237      end 111
```

```
*** End of Assembly
```

Symbol List starts at 0x7800, ends at 0x850C, used 0x0D0C, remaining 0x3184

Symbols unsorted:

BUFRADRZ	0026	SLOT16Z	002B	DRIVEZ	002C	SECTORZ	002D	TRACKZ	002E
VOLUMEZ	002F	COMMANDZ	003C	ROMSECTR	003D	GENPTR	003E	IOBADR	004A
DATPTR	00EE	CFFAVRSN	0005	CFFABLD	0006	VRSN3.3	0033	VRSN4.1	0041
BLD4.1	0046	VRSN4.3	0043	BLD4.3	0008	VRSN4.5	0045	BLD4.5	0006
ZERO	0000	NAMESIZE	0018	NEGONE	00FF	ASCIMASK	007F	ASCIFLAG	0080
LARROW	0088	RETURN	008D	SPACE	00A0	WAIT100U	0004	WAIT001M	0011
WAIT040M	007C	WAIT100M	00C5	MAXIMTRK	0004	MINIMAG	0001	MAXIMAG	0008
VOLXINC	0003	VOLYINC	0030	MAXCFSEC	0020	MAXCFTRK	0030	SLOTMASK	0007
LBAMASK	000F	NIBLMASK	000F	SECMASKL	000F	SECMASKH	0010	TRKMASK	003F
WAITMLN	0005	IDSRLLEN	0014	IDSRLFFF	0014	VOLSIZE	0300	ROMENTRY	005C
VRSNOFF	00FE	SEEKCMD	0000	READCMD	0001	WRITCMD	0002	FMTCMD	0004
READCMD2	0011	WRITCMD2	0012	RSETCMD	0020	RDLBACMD	0021	WRLBACMD	0022
RDIDCMD	0030	RWSYNERR	0030	SLOTNDX	0001	DRVNDX	0002	VOLNDX	0003
TRKNDX	0004	SECNDX	0005	BUFRNDX	0008	PHASNDX	000A	XFERNDX	000B
CMDNDX	000C	ERRNDX	000D	LVOLNDX	000E	LSLTNDX	000F	LDRVNDX	0010
ATA_READ	0020	ATA_WRIT	0030	ATA_ID	00EC	NOERROR	0000	BUSYERR	0001
RDYERR	0002	STATERR	0003	LBAERR	0004	CMDERR	0008	CHKERR	000C
RSTERR	0080	RD1ERR	0090	WR1ERR	00A0	FMTERR	00B0	IDERR	00C0
RD2ERR	00D0	WR2ERR	00E0	UNKERR	00F0	SIGVAL1	00C4	SIGVAL2	00B8
SIGVAL3	0090	SIGVAL4	00ED	PAGESIZE	0100	STACK	0100	RSTRTSAV	0110
MESSAGE	0112	VERSION	0113	TEMPERR	0114	LCRAM	0115	STKCODE	0116
DOSWARM	03D0	DOSCOLD	03D3	CALLFM	03D6	RWTS	03D9	GETFMCB	03DC
RDCLKVSN	03E1	GETIOCB	03E3	PRTERADR	03E8	HOOKDOS	03EA	XFERADR	03ED
AUTOBRK	03EF	AUTORSET	03F2	PWRSTATE	03F4	USRAHAND	03F5	USRYHAND	03F8
NMASKIRQ	03FB	MASKIRQ	03FE	XMODE	04FB	BOOTPGS	08FF	CFSLOT	0478
CFSLOT16	04F8	LBA0	0578	LBA1	05F8	LBA2	0678	LBA3	06F8
PAGE	0778	CFPAGECX	07F8	ERRSTAT	0478	CFERROR	04F8	DRIVE	0578
VOLUME	05F8	SAVEADRL	0678	SAVEADRH	06F8	DOSVRSN	0778	DOSNBUF1	07F8
FTMOD	9E41	SDFMOD	A0D9	KWRANGE	A955	KYWRDFND	AA65	VOLVAL	AA66
CATHMOD1	AD9D	CATHMOD2	AE16	LCDMOD	B202	VOLNUMBR	B5F9	SETDRIVE	B706
RESTART	B744	CALLRWTS	B7B5	VOLEXPT	B7EB	NBUF1BUF	BB00	INITADR	BED9
INITDOS	BFF8	DISKTBL	BFFB	BLDVRSN	BFF0	BLDNMBR	BFF1	MNGDISK	BFF2
BCFGNDX	BFFC	NBUF1ADR	BFFD	PAGE08	0800	PAGE09	0900	PAGEBF	BF00
PAGEC0	C000	PAGEC8	C800	PAGED0	D000	VID80OFF	C00C	ALTCHOFF	C00E
RDLGRAM	C012	ATADATAH	C080	SETCSMSK	C081	CLRCMSK	C082	ATADEVCT	C086
ATASTAT2	C086	ATADATAL	C088	ATAERROR	C089	ATASECCT	C08A	ATASECTR	C08B
ATACYLNL	C08C	ATACYLNH	C08D	ATAHEAD	C08E	ATACMD	C08F	ATASTAT	C08F
IMAGSIZE	0018	OFFSET0	0010	OFFSET1	0100	ROM2WP	C082	RAM2WE	C083
RAM1WE	C08B	CLRROM	CFFF	INIT	FB2F	HOME	FC58	CROUT	FD8E
COUT	FDED	SETNORM	FE84	SETVID	FE89	SETKBD	FE93	MONITOR	FF69
CFBOOT	C000	ROMHOOK	C010	ROMUHOOK	C018	USRBOOT	C020	TOGGLE	C024
VOLBOOT2	C02A	VOLBOOT	C030	DISKRWTS	C03B	CFRWTS	C04B	CFRWTS2	C052
ROMBOOT	C05C	CFRWTS3	C064	MODOS3	C0F3	ROMSIG	C800	CFSRLBUF	C804
CFNAME	C818	CFDATE	C830	CFLBANUM	C836	CFMAXDRV	C83A	CFREMVOL	C83B
DFLTBOOT	C83C	GETSLOT2	C83D	GETSLOT	C841	EXIT	C860	EXITBGN	0116
EXIT1	0116	EXIT2	0117	EXITADR	011A	EXIT3	011D	EXITLEN	000C
EXITERR	C86C	EXITRWTS	C875	ENTRDOS	C883	PRTMESGS	C899	PRTMSG	C8BB
PRTMSG1	C8BF	CLRDISK	C8CA	SETDISK	C8CC	DOBOOT	C8D0	CHKDRVZ	C958
BOOTVOL	C99E	BOOTVOL2	C9A4	CHKDOS3X	CA3A	FNDOS	CA7A	NODOS	CA7D
FNDOSLEN	0005	CHKDOS43	CA7F	CHKDOS45	CA93	CHKDOS4X	CAA9	CHKDOS41	CADA
DOTOGGLE	CB14	USTDOS4X	CB50	USTDOS41	CB6C	USTDOS33	CB7E	INITDAT	CB8E
DOMODOS3	CB9F	DVTS2LBA	CBDE	CHKCFLBA	CC30	CFBUSY	CC55	CFREADY	CC6D
CFSTATUS	CC8E	DOATARST	CCAC	CFWAIT	CCBD	CFCHECK	CCC9	LOADCMD	CCD6
LOADLBA	CCF2	CFRESET	CD1F	DOCFCMD	CD39	READID	CD75	RD2PAGE	CD9C
WR2PAGE	CDAB	RD1PAGE	CDBD	WR1PAGE	CDD9	GETSTAT	CE15	CFRTN	CE24
FMTVOL	CE25	SET2READ	CE73	SET2WRIT	CE83	CF2BUFR	CE93	BUFR2CF	CEA6

CF2TEMP	CEBB	TEMP2CF	CECC	TOSS	CEDF	RVSELEAV	CEEB	IMAGLBA	CEFB
DRVXTBL	CF03	VOLXTBL	CF03	DRVYTBLL	CF13	VOLYTBLL	CF13	DRVYTBLLH	CF23
VOLYTBLLH	CF23	MESGS	CF33	MESG1A	CF33	MESG1B	CF34	MESG2A	CF36
MESG2B	CF3F	MESG3A	CF46	MESG3B	CF49	MESG4A	CF5B	MESG4B	CF5E
MESG4C	CF61	MESG4D	CF64	MESG5A	CF67	MESG5B	CF69	MESG5C	CF6E
MESG5D	CF71	MESG5E	CF76	MESG5F	CF7B	MESG5G	CF7F	MESG5H	CF89
MESG5I	CF8E	MESG5J	CF91	MESG5K	CF96	MESG5L	CF9A	TEXTS	CF9F
IOTEXT1A	CF9F	IOTEXT1B	CFA2	IOTEXT2A	CFA5	IOTEXT2B	CFA8	IOTEXT2C	CFAB
IOTEXT2D	CFAE	IOTEXT3A	CFB1	IOTEXT3B	CFB4	IOTEXT3C	CFB7	IOTEXT3D	CFBA
ERRTEXT1	CFBD	ERRTEXT2	CFC0	ERRTEXT3	CFC3	ERRTEXT4	CFC6	ERRTEXT5	CFC9
ERRTEXT6	CFCC	ERRTEXT7	CFCF						

Symbols alphabetically sorted:

ALTCHOFF	C00E	ASCIFLAG	0080	ASCIMASK	007F	ATACMD	C08F	ATACYLNH	C08D
ATACYLNL	C08C	ATADATAH	C080	ATADATAL	C088	ATADEVCT	C086	ATAERROR	C089
ATAHEAD	C08E	ATASECCT	C08A	ATASECTR	C08B	ATASTAT	C08F	ATASTAT2	C086
ATA_ID	00EC	ATA_READ	0020	ATA_WRIT	0030	AUTOBRK	03EF	AUTORSET	03F2
BCFGNDX	BFFC	BLD4.1	0046	BLD4.3	0008	BLD4.5	0006	BLDNMBR	BFF1
BLDVRSN	BFF0	BOOTPGS	08FF	BOOTVOL	C99E	BOOTVOL2	C9A4	BUFR2CF	CEA6
BUFRADRZ	0026	BUFRNDX	0008	BUSYERR	0001	CALLFM	03D6	CALLRWTS	B7B5
CATHMOD1	AD9D	CATHMOD2	AE16	CF2BUFR	CE93	CF2TEMP	CEBB	CFBOOT	C000
CFBUSY	CC55	CFCHECK	CCC9	CFDATE	C830	CFERROR	04F8	CFFABLD	0006
CFFAVRSN	0005	CFLBANUM	C836	CFMAXDRV	C83A	CFNAME	C818	CFPAGECX	07F8
CFREADY	CC6D	CFREMOVOL	C83B	CFRESET	CD1F	CFRTN	CE24	CFRWTS	C04B
CFRWTS2	C052	CFRWTS3	C064	CFSLOT	0478	CFSLOT16	04F8	CFSRLBUF	C804
CFSTATUS	CC8E	CFWAIT	CCBD	CHKCFLBA	CC30	CHKDOS3X	CA3A	CHKDOS41	CADA
CHKDOS43	CA7F	CHKDOS45	CA93	CHKDOS4X	CAA9	CHKDRVZ	C958	CHKERR	000C
CLRCMSK	C082	CLRDISK	C8CA	CLRROM	CFFF	CMDERR	0008	CMDNDX	000C
COMMANDZ	003C	COUT	FD8E	CROUT	FD8E	DATPTR	00EE	DFLTBOOT	C83C
DISKRWTS	C03B	DISKTBL	BFFB	DOATARST	CCAC	DOBOOT	C8D0	DOFCMD	CD39
DOMODOS3	CB9F	DOSCOLD	03D3	DOSNBUF1	07F8	DOSVRSN	0778	DOSWARM	03D0
DOTOGGLE	CB14	DRIVE	0578	DRIVEZ	002C	DRVNDX	0002	DRVXTBL	CF03
DRVYTBLLH	CF23	DRVYTBLL	CF13	DVTS2LBA	CBDE	ENTRDOS	C883	ERRNDX	000D
ERRSTAT	0478	ERRTEXT1	CFBD	ERRTEXT2	CFC0	ERRTEXT3	CFC3	ERRTEXT4	CFC6
ERRTEXT5	CFC9	ERRTEXT6	CFCC	ERRTEXT7	CFCF	EXIT	C860	EXIT1	0116
EXIT2	0117	EXIT3	011D	EXITADR	011A	EXITBGN	0116	EXITERR	C86C
EXITLEN	000C	EXITRWTS	C875	FMTCMD	0004	FMterr	00B0	FMTVOL	CE25
FNDOS	CA7A	FNDOSLEN	0005	FTMOD	9E41	GENPTR	003E	GETFMCB	03DC
GETIOCB	03E3	GETSLOT	C841	GETSLOT2	C83D	GETSTAT	CE15	HOME	FC58
HOOKDOS	03EA	IDERR	00C0	IDSRLLEN	0014	IDSRLOFF	0014	IMAGLBA	CEFB
IMAGSIZE	0018	INIT	FB2F	INITADR	BED9	INITDAT	CB8E	INITDOS	BFF8
IOBADR	004A	IOTEXT1A	CF9F	IOTEXT1B	CFA2	IOTEXT2A	CFA5	IOTEXT2B	CFA8
IOTEXT2C	CFAB	IOTEXT2D	CFAE	IOTEXT3A	CFB1	IOTEXT3B	CFB4	IOTEXT3C	CFB7
IOTEXT3D	CFBA	KWRANGE	A955	KYWRDFND	AA65	LARROW	0088	LBA0	0578
LBA1	05F8	LBA2	0678	LBA3	06F8	LBAERR	0004	LBAMASK	000F
LCDMOD	B202	LCRAM	0115	LDRVNDX	0010	LOADCMD	CCD6	LOADLBA	CCF2
LSLTNDX	000F	LVOLNDX	000E	MASKIRQ	03FE	MAXCFSEC	0020	MAXCFTRK	0030
MAXIMAG	0008	MAXIMTRK	0004	MESG1A	CF33	MESG1B	CF34	MESG2A	CF36
MESG2B	CF3F	MESG3A	CF46	MESG3B	CF49	MESG4A	CF5B	MESG4B	CF5E
MESG4C	CF61	MESG4D	CF64	MESG5A	CF67	MESG5B	CF69	MESG5C	CF6E
MESG5D	CF71	MESG5E	CF76	MESG5F	CF7B	MESG5G	CF7F	MESG5H	CF89
MESG5I	CF8E	MESG5J	CF91	MESG5K	CF96	MESG5L	CF9A	MESGS	CF33
MESSAGE	0112	MINIMAG	0001	MNGDISK	BFF2	MODOS3	C0F3	MONITOR	FF69
NAMEsize	0018	NBUF1ADR	BFFD	NBUF1BUF	BB00	NEGONE	00FF	NIBLMASK	000F
NMASKIRQ	03FB	NODOS	CA7D	NOERROR	0000	OFFSET0	0010	OFFSET1	0100
PAGE	0778	PAGE08	0800	PAGE09	0900	PAGEBF	BF00	PAGEC0	C000
PAGEC8	C800	PAGED0	D000	PAGESIZE	0100	PHASNDX	000A	PRTERADR	03E8
PRTMESG	C8BB	PRTMESG1	C8BF	PRTMESGS	C899	PWRSTATE	03F4	RAM1WE	C08B
RAM2WE	C083	RD1ERR	0090	RD1PAGE	CDBD	RD2ERR	00D0	RD2PAGE	CD9C

RDCLKVSN	03E1	RDIDCMD	0030	RDLBACMD	0021	RDLGRAM	C012	RDYERR	0002
READCMD	0001	READCMD2	0011	READID	CD75	RESTART	B744	RETURN	008D
ROM2WP	C082	ROMBOOT	C05C	ROMENTRY	005C	ROMHOOK	C010	ROMSECTR	003D
ROMSIG	C800	ROMUHOOK	C018	RSETCMD	0020	RSTERR	0080	RSTRTSV	0110
RVSELEAV	CEEB	RWSYNERR	0030	RWTS	03D9	SAVEADRH	06F8	SAVEADRL	0678
SDFMOD	A0D9	SECMASKH	0010	SECMASKL	000F	SECNDX	0005	SECTORZ	002D
SEEKCMD	0000	SET2READ	CE73	SET2WRIT	CE83	SETCSMSK	C081	SETDISK	C8CC
SETDRIVE	B706	SETKBD	FE93	SETNORM	FE84	SETVID	FE89	SIGVAL1	00C4
SIGVAL2	00B8	SIGVAL3	0090	SIGVAL4	00ED	SLOT16Z	002B	SLOTMASK	0007
SLOTNDX	0001	SPACE	00A0	STACK	0100	STATERR	0003	STKCODE	0116
TEMP2CF	CECC	TEMPERR	0114	TEXTS	CF9F	TOGGLE	C024	TOSS	CEDF
TRACKZ	002E	TRKMASK	003F	TRKNDX	0004	UNKERR	00F0	USRAHAND	03F5
USRBOOT	C020	USRYHAND	03F8	USTDOS33	CB7E	USTDOS41	CB6C	USTDOS4X	CB50
VERSION	0113	VID80OFF	C00C	VOLBOOT	C030	VOLBOOT2	C02A	VOLEXPT	B7EB
VOLNDX	0003	VOLNUMBR	B5F9	VOLSIZE	0300	VOLUME	05F8	VOLUMEZ	002F
VOLVAL	AA66	VOLXINC	0003	VOLXTBL	CF03	VOLYINC	0030	VOLYTBLLH	CF23
VOLYTBLL	CF13	VRSN3.3	0033	VRSN4.1	0041	VRSN4.3	0043	VRSN4.5	0045
VRSNOFF	00FE	WAIT001M	0011	WAIT040M	007C	WAIT100M	00C5	WAIT100U	0004
WAITMLN	0005	WR1ERR	00A0	WR1PAGE	CDD9	WR2ERR	00E0	WR2PAGE	CDAB
WRITCMD	0002	WRITCMD2	0012	WRLBACMD	0022	XFERADR	03ED	XFERNDX	000B
XMODE	04FB	ZERO	0000						

Symbols numerically sorted:

ZERO	0000	SEEKCMD	0000	NOERROR	0000	SLOTNDX	0001	READCMD	0001
MINIMAG	0001	BUSYERR	0001	WRITCMD	0002	RDYERR	0002	DRVNDX	0002
VOLXINC	0003	VOLNDX	0003	STATERR	0003	WAIT100U	0004	TRKNDX	0004
MAXIMTRK	0004	LBAERR	0004	FMTCMD	0004	WAITMLN	0005	SECNDX	0005
FNDOSLEN	0005	CFFAVRSN	0005	CFFABLD	0006	BLD4.5	0006	SLOTMASK	0007
MAXIMAG	0008	CMDERR	0008	BUFRNDX	0008	BLD4.3	0008	PHASNDX	000A
XFERNDX	000B	EXITLEN	000C	CMDNDX	000C	CHKERR	000C	ERRNDX	000D
LVOLNDX	000E	SECMASKL	000F	NIBLMASK	000F	LSLTNDX	000F	LBAMASK	000F
SECMASKH	0010	OFFSET0	0010	LDRVNDX	0010	WAIT001M	0011	READCMD2	0011
WRITCMD2	0012	IDSRL0FF	0014	IDSRLLEN	0014	NAME SIZE	0018	IMAGSIZE	0018
RSETCMD	0020	MAXCFSEC	0020	ATA_READ	0020	RDLBACMD	0021	WRLBACMD	0022
BUFRADRZ	0026	SLOT16Z	002B	DRIVEZ	002C	SECTORZ	002D	TRACKZ	002E
VOLUMEZ	002F	VOLYINC	0030	RWSYNERR	0030	RDIDCMD	0030	MAXCFTRK	0030
ATA_WRIT	0030	VRSN3.3	0033	COMMANDZ	003C	ROMSECTR	003D	GENPTR	003E
TRKMASK	003F	VRSN4.1	0041	VRSN4.3	0043	VRSN4.5	0045	BLD4.1	0046
IOBADR	004A	ROMENTRY	005C	WAIT040M	007C	ASCIMASK	007F	RSTERR	0080
ASCIFLAG	0080	LARROW	0088	RETURN	008D	SIGVAL3	0090	RD1ERR	0090
WR1ERR	00A0	SPACE	00A0	FMTERR	00B0	SIGVAL2	00B8	IDERR	00C0
SIGVAL1	00C4	WAIT100M	00C5	RD2ERR	00D0	WR2ERR	00E0	ATA_ID	00EC
SIGVAL4	00ED	DATPTR	00EE	UNKERR	00F0	VRSNOFF	00FE	NEGONE	00FF
STACK	0100	PAGESIZE	0100	OFFSET1	0100	RSTRTSV	0110	MESSAGE	0112
VERSION	0113	TEMPERR	0114	LCRAM	0115	STKCODE	0116	EXITBGN	0116
EXIT1	0116	EXIT2	0117	EXITADR	011A	EXIT3	011D	VOLSIZE	0300
DOSWARM	03D0	DOSCOLD	03D3	CALLFM	03D6	RWTS	03D9	GETFMCB	03DC
RDCLKVSN	03E1	GETIOCB	03E3	PRTERADR	03E8	HOOKDOS	03EA	XFERADR	03ED
AUTOBRK	03EF	AUTORSET	03F2	PWRSTATE	03F4	USRAHAND	03F5	USRYHAND	03F8
NMASKIRQ	03FB	MASKIRQ	03FE	ERRSTAT	0478	CFSLOT	0478	CFSLOT16	04F8
CFERROR	04F8	XMODE	04FB	LBA0	0578	DRIVE	0578	VOLUME	05F8
LBA1	05F8	SAVEADRL	0678	LBA2	0678	SAVEADRH	06F8	LBA3	06F8
PAGE	0778	DOSVRSN	0778	DOSNBUF1	07F8	CFPAGECX	07F8	PAGE08	0800
BOOTPGS	08FF	PAGE09	0900	FTMOD	9E41	SDFMOD	A0D9	KWRANGE	A955
KYWRDFND	AA65	VOLVAL	AA66	CATHMOD1	AD9D	CATHMOD2	AE16	LCDMOD	B202
VOLNUMBR	B5F9	SETDRIVE	B706	RESTART	B744	CALLRWTS	B7B5	VOLEXPT	B7EB
NBUF1BUF	BB00	INITADR	BED9	PAGEBF	BF00	BLDVRSN	BFF0	BLDNMBR	BFF1
MNGDISK	BFF2	INITDOS	BFF8	DISKTBL	BFFB	BCFGNDX	BFFC	NBUF1ADR	BFFD
PAGEC0	C000	CFBOOT	C000	VID80OFF	C00C	ALTCHOFF	C00E	ROMHOOK	C010

RDLDRAM	C012	ROMUHOOK	C018	USRBOOT	C020	TOGGLE	C024	VOLBOOT2	C02A
VOLBOOT	C030	DISKRWTS	C03B	CFRWTS	C04B	CFRWTS2	C052	ROMBOOT	C05C
CFRWTS3	C064	ATADATAH	C080	SETCSMSK	C081	ROM2WP	C082	CLRCSMSK	C082
RAM2WE	C083	ATASTAT2	C086	ATADEVCT	C086	ATADATAL	C088	ATAERROR	C089
ATASECCT	C08A	RAM1WE	C08B	ATASECTR	C08B	ATACYLNL	C08C	ATACYLNH	C08D
ATAHEAD	C08E	ATASTAT	C08F	ATACMD	C08F	MODOS3	C0F3	ROMSIG	C800
PAGEC8	C800	CFSRLBUF	C804	CFNAME	C818	CFDATE	C830	CFLBANUM	C836
CFMAXDRV	C83A	CFREMVOL	C83B	DFLTBOOT	C83C	GETSLOT2	C83D	GETSLOT	C841
EXIT	C860	EXITERR	C86C	EXITRWTS	C875	ENTRDOS	C883	PRTMESGS	C899
PRTMSG	C8BB	PRTMSG1	C8BF	CLRDISK	C8CA	SETDISK	C8CC	DOBOOT	C8D0
CHKDRVZ	C958	BOOTVOL	C99E	BOOTVOL2	C9A4	CHKDOS3X	CA3A	FNDOS	CA7A
NODOS	CA7D	CHKDOS43	CA7F	CHKDOS45	CA93	CHKDOS4X	CAA9	CHKDOS41	CADA
DOTOGGLE	CB14	USTDOS4X	CB50	USTDOS41	CB6C	USTDOS33	CB7E	INITDAT	CB8E
DOMODOS3	CB9F	DVTS2LBA	CBDE	CHKCFLBA	CC30	CFBUSY	CC55	CFREADY	CC6D
CFSTATUS	CC8E	DOATARST	CCAC	CFWAIT	CCBD	CFCHECK	CCC9	LOADCMD	CCD6
LOADLBA	CCF2	CFRESET	CD1F	DOCFCMD	CD39	READID	CD75	RD2PAGE	CD9C
WR2PAGE	CDAB	RD1PAGE	CDBD	WR1PAGE	CDD9	GETSTAT	CE15	CFRTN	CE24
FMTVOL	CE25	SET2READ	CE73	SET2WRIT	CE83	CF2BUFR	CE93	BUFR2CF	CEA6
CF2TEMP	CEBB	TEMP2CF	CECC	TOSS	CEDF	RVSELEAV	CEEB	IMAGLBA	CEFB
VOLXTBL	CF03	DRVXTBL	CF03	VOLYTBLL	CF13	DRVYTBLL	CF13	VOLYTBLLH	CF23
DRVYTBLLH	CF23	MESGS	CF33	MESG1A	CF33	MESG1B	CF34	MESG2A	CF36
MESG2B	CF3F	MESG3A	CF46	MESG3B	CF49	MESG4A	CF5B	MESG4B	CF5E
MESG4C	CF61	MESG4D	CF64	MESG5A	CF67	MESG5B	CF69	MESG5C	CF6E
MESG5D	CF71	MESG5E	CF76	MESG5F	CF7B	MESG5G	CF7F	MESG5H	CF89
MESG5I	CF8E	MESG5J	CF91	MESG5K	CF96	MESG5L	CF9A	TEXTS	CF9F
IOTEXT1A	CF9F	IOTEXT1B	CFA2	IOTEXT2A	CFA5	IOTEXT2B	CFA8	IOTEXT2C	CFAB
IOTEXT2D	CFAE	IOTEXT3A	CFB1	IOTEXT3B	CFB4	IOTEXT3C	CFB7	IOTEXT3D	CFBA
ERRTEXT1	CFBD	ERRTEXT2	CFC0	ERRTEXT3	CFC3	ERRTEXT4	CFC6	ERRTEXT5	CFC9
ERRTEXT6	CFCC	ERRTEXT7	CFCF	CLRROM	CFFF	PAGED0	D000	INIT	FB2F
HOME	FC58	CROUT	FD8E	COUT	FDED	SETNORM	FE84	SETVID	FE89
SETKBD	FE93	MONITOR	FF69						