# MORE ABOUT

# CBM 4040 DOS

*fully documented & cross referenced by*

Gary van Beeck

ThuisComputer

Published by
**Uitgeverij ThuisComputer**
Zwedenburg 119, 2591 BD The Hague, Holland

Copyright © 1985 by Uitgeverij ThuisComputer
**All rights reserved**

Copying, duplicating, selling or otherwise distributing this product is hereby strictly forbidden, unless prior written consent has been obtained from the publisher.

The words Commodore and CBM are registered trademarks of Commodore Business Machines.

Every effort has been made to provide accurate information in this book. However, neither the author nor the publisher will accept any responsibility for any loss or damage, tangible or intangible, resulting from use or improper or unintended use of the information provided.

Printed in Holland

# INTRODUCTION

This manual is the result of a desperate effort to find out more about the mysteries of the Commodore 4040 Disk Drive and its innards. Unlike its brothers, the CBM 8050/8250 and the 1541 systems, about which — without much help from Commodore -- reasonable documentation was published soon after their introduction, most of its operating system remained a closely guarded secret. It was only through the efforts of enthusiasts like Jim Butterfield in Canada and Raeto West in England that titbits of interesting and useful information have managed to come to light at all.

The author has made an effort to unravel the 4040 DOS in order to provide an annotated source listing. It is obvious that the reader is assumed to be reasonably familiar with machine language and with the contents of the 4040 User Manual. The book is primarily aimed at the intermediate and advanced programmer but the author hopes it will also provide invaluable help to the novice user.

No claim is made as to the accuracy of the information presented in this manual. It has been compiled without any help from Commodore and none of its contents have been reviewed by them. The labels used in the source listings are generally those used by Commodore and non-Commodore authors.

We welcome any supplementary information and suggestions for this manual, which we shall gladly use in a subsequent edition.

## CONTENTS

### Bus Controller Section

```
line   addr  object     source code

00006  0000              ************************************
00007  0000              *                                  *
00008  0000              *                                  *
00009  0000              *      Dual Floppy Controller      *
00010  0000              *                                  *
00011  0000              *      CBM 2030 - 3030 - 3040      *
00012  0000              *                                  *
00013  0000              *      Disk Operating System       *
00014  0000              *                                  *
00015  0000              *      Compiled by                 *
00016  0000              *          Gary van Beeck          *
00017  0000              *          The Hague, Holland      *
00018  0000              *                                  *
00019  0000              *                                  *
00020  0000              *          January 1985            *
00021  0000              *                                  *
00022  0000              *                                  *
00023  0000              *      Copyright - © 1895 - All    *
00024  0000              *          rights reserved         *
00025  0000              *                                  *
00026  0000              *      UITGEVERIJ THUISCOMPUTER    *
00027  0000              *                                  *
00028  0000              *      Zwedenburg 119              *
00029  0000              *      2591 BD The Hague, Holland  *
00030  0000              *      Phone (+31)(0)70 - 473.777  *
00031  0000              *                                  *
00032  0000              ************************************

00034  0000  Adapted from — among others — the Version 2.1 original source file
```

```
line   addr  object     source code

00036  0000  ===> Equates <===
00037  0000
00038  0000            rom    = $d000
00039  0000
00040  0000            lrf    = $80          last record flag
00041  0000            getflg = $40          buffer dirty flag
00042  0000            ovrflo = $20          overflow flag
00043  0000            outran = $50          out of range flag
00044  0000            nssl   = 6            number of side sector links
00045  0000            ssioff = 4+nssl+nssl  side sector offset
00046  0000            nssp   = 120          number of side sector pointers in
                                             buffer
00047  0000            mxchns = 8            maximum available channels
00048  0000            id     = $1000        controller ID byte
00049  0000            maxsa  = 18           highest secondary address
00050  0000            vererr = 7
00051  0000            cr     = 13           carriage return
00052  0000            maxtrk = 36           maximum number of tracks plus one
00053  0000            bfcnt  = 12           number of buffers
00054  0000            bamjob = bfcnt
00055  0000            bam0   = $4100        BAM drive 0
00056  0000            bam1   = $4200        BAM drive 1
00057  0000            cbptr  = bamjob+bamjob+4      command buffer pointer
00058  0000            errchn = mxchns-1     error channel
00059  0000            errsa  = 16           secondary address for error channel
00060  0000            cmdchn = mxchns-2     command channel
00061  0000            lxint  = $3f          internal channel for LINDX
00062  0000            cmdsa  = 15           command secondary address
00063  0000            apmode = 2            append mode
00064  0000            mdmode = 3            modify mode
00065  0000            rdmode = 0            read mode
00066  0000            wtmode = 1            write mode
00067  0000            reltyp = 4            relative file type
00068  0000            dirtyp = 7            direct access
00069  0000            seqtyp = 1            sequential file type
00070  0000            prgtyp = 2            program file type
00071  0000            usrtyp = 3            user file type
00072  0000            typmsk = $e           type mask
00073  0000            irsa   = 17           internal read secondary address
00074  0000            iwsa   = 18           internal write secondary address
00075  0000            id2040 = $f           2040 controller ID
00076  0000            id2030 = $64          2030 controller ID
00077  0000            dosver = 2            DOS version
00078  0000            fm2040 = $41          2040 format version
00079  0000            fm2030 = $42          2030 format version
00080  0000
00081  0000
00082  0000  ===> Controller Job Types <===
00083  0000
00084  0000            read   = $80          read a sector
00085  0000            write  = $90          write a sector
00086  0000            wverfy = $a0          verify after write
00087  0000            seek   = $b0          seek any sector
```

```
line   addr  object     source code

00088  0000             bump   = $c0          move head to track 1
00089  0000             jumpc  = $d0          jump to machine code in buffer
00090  0000             exec   = $e0          execute code in buffer when speed &
                                              head ready
00091  0000
00092  0000
00093  0000  ===> Job Error Codes <===
00094  0000
00095  0000             goodj  = 1            job completed successfully
00096  0000             nohdr  = 2            header block not found
00097  0000             nosync = 3            no sync character
00098  0000             nodblk = 4            data block not found
00099  0000             badbch = 5            data block checksum error
00100  0000             wverer = 7            verify error after write
00101  0000             wrprot = 8            write protect error
00102  0000             badhch = 9            header block checksum error
00103  0000             badblk = 10           data block too long
00104  0000             badid  = 11           ID mismatch error
00105  0000             badbyt = 16           byte decoding error
00106  0000             mxfils = 5            maximum number of filenames in
                                              string
00107  0000             cmdind = 30           command buffer index (*2)
00108  0000             dirlen = 24           directory length used
00109  0000             nbsiz  = 27           nambuf text size
00110  0000             ctbsiz = 7            controller table size
00111  0000             cmdlen = 58           command string size
00112  0000
00113  0000
00114  0000  ===> I/O Definitions <===
00115  0000
00116  0000             unlsn  = $3f          IEEE unlisten command
00117  0000             notrdy = $0           not ready
00118  0000             talker = $80          IEEE talker flag
00119  0000             lisner = 1            IEEE listener flag
00120  0000             eoiout = $80          talk with EOI
00121  0000             eoisnd = $08          not (EOI) to send
00122  0000             rdytlk = $88          talk no EOI
00123  0000             rdylst = $1           ready to listen
00124  0000             rndrdy = rdytlk+rdylst random CHNRDY
00125  0000             rndeoi = eoiout+rdylst random with EOI
00126  0000
00127  0000
00128  0000  ===> I/O Registers <===
00129  0000
00130  0000                                   MOS 6532-1
00131  0000                   * = $200
00132  0200             ieeedi * = *+1        IEEE data in register
00133  0201             paddl  * = *+1        IEEE data direction register
00134  0202             ieeedo * = *+1        IEEE data out register
00135  0203             pbddl  * = *+1        IEEE data out direction register
00136  0204
00137  0204                                   MOS 6532-2
00138  0204                   * = $280        IEEE control port register 1
```

```
line   addr  object     source code

00139  0280             pad2   * = *+1
00140  0281             atna   = 1              attention acknowledge line
00141  0281             daco   = 2              (inverted)
00142  0281             rfdo   = 4
00143  0281             eoio   = 8
00144  0281             davo   = 16
00145  0281             eoii   = 32
00146  0281             davi   = 64
00147  0281             atni   = 128            (inverted)
00148  0281             padd2  * = *+1          direction register 1
00149  0282             pbd2   * = *+1          IEEE/LED register 2. Bits 0-2 =
                                                 device number select
00150  0283             led1   = $8             active LED 1
00151  0283             led0   = $10            active LED 0
00152  0283             errled = 32             hardware initialization error LED
00153  0283             ndaci  = 64
00154  0283             nrfdi  = 128
00155  0283             pbdd2  * = *+1          IEEE/LED direction register, control
                                                 register 2
00156  0284             atnnd  * = *+1          ATN causes IRQ
00157  0285             atnpd  * = *+1
00158  0286             atnne  * = *+1
00159  0287             atnpe  * = *+1          bus interrupt flag (lda=clr)
00160  0288
00161  0288
00162  0288  ===> Common Area Defines <===
00163  0288
00164  0288                    *=$1003
00165  1003             jobs   *=*+15           job queue
00166  1012             trks   *=*+15           job track table
00167  1021                    *=$1021         headers at $1021
00168  1021             hdrs   *=*+120          job header
00169  1099                    *=$1099         sectors/track table
00170  1099             numsec *=*+ctbsiz-1
00171  109f             vernum *=*+1            version number
00172  10a0             actjob *=*+1            controller's active job
00173  10a1                    *=$10f0
00174  10f0             vnmi   *=*+2            indirect for NMI
00175  10f2             nmiflg *=*+1            flag for NMI in progress
00176  10f3             autofg *=*+1            enable (0)/disable (1) auto
                                                 initialisation (read BAM)
00177  10f4             bufs   = $1100          start of data buffers
00178  10f4
00179  10f4
00180  10f4  ===> Zero Page Variables <===
00181  10f4
00182  10f4                    *=$0
00183  0000             usrjmp *=*+2            user jump table pointer (=$dd on
                                                 release)
00184  0002             bmpnt  *=*+2            bit map pointer
00185  0004             temp   *=*+6            temporary work space
00186  000a             ip     *=*+2            indirect pointer variable
00187  000c             lsnadr *=*+1            listen address
```

```
line   addr  object     source code

00188  000d            tlkadr *=*+1          talker address
00189  000e            lsnact *=*+1          active listener flag
00190  000f            tlkact *=*+1          active talker flag
00191  0010            adrsed *=*+1          addressed flag
00192  0011            prgtrk *=*+1          last program accessed
00193  0012            drvnum *=*+1          current drive number
00194  0013            track  *=*+1          current track
00195  0014            sector *=*+1          current sector
00196  0015            lindx  *=*+1          logical index
00197  0016            sa     *=*+1          secondary address
00198  0017            orgsa  *=*+1          original SA
00199  0018            data   *=*+1          temporary data byte
00200  0019
00201  0019            t0     = temp         temporary storage addresses
00202  0019            t1     = temp+1
00203  0019            t2     = temp+2
00204  0019            t3     = temp+3
00205  0019            t4     = temp+4
00206  0019            r0     *=*+1          temporary results
00207  001a            r1     *=*+1
00208  001b            r2     *=*+1
00209  001c            r3     *=*+1
00210  001d            r4     *=*+1
00211  001e            result *=*+4          storage for multiplication/division
00212  0022            accum  *=*+5          register for multiplication/division
00213  0027            dirbuf *=*+2          current buffer pointer lo
00214  0029
00215  0029
00216  0029  ===> Zero Page Arrays <===
00217  0029
00218  0029            buftab *=*+cbptr+4    buffer pointer table lo/hi, buffer 0
                                             - 15
00219  0049            cb     = buftab+cbptr address lo, ind. pointer into
                                             command buffer
00220  0049            buf0   *=*+mxchns     table of channel numbers for
                                             buffers. $FF = inactive
00221  0051            buf1   *=*+mxchns     table of channel numbers for
                                             buffers. $FF = inactive
00222  0059            nbkl
00223  0059            recl   *=*+mxchns     table of lo bytes for record number
                                             for each buffer
00224  0061            nbkh
00225  0061            rech   *=*+mxchns     table of hi bytes for record number
                                             for each buffer
00226  0069            nr     *=*+mxchns     table of next record number for
                                             buffers
00227  0071            rs     *=*+mxchns     table of record sizes for each
                                             buffer
00228  0079            ss     *=*+mxchns     table of side sectors for each
                                             buffer
00229  0081            flptr  *=*+1          file stream 1 pointer
00230  0082
00231  0082
```

```
line   addr  object     source code

00232  0082  ===> RAM Variables moved to Zero Page <===
00233  0082
00234  0082            recptr *=*+1            position in record
00235  0083            ssnum  *=*+1            number of side sector
00236  0084            ssind  *=*+1            index to side sector
00237  0085            relptr *=*+1            relative file pointer to track
00238  0086            filent *=*+mxfils       directory entry
00239  008b            fildat *=*+mxfils       drive number, pattern
00240  0090            filtyp *=*+mxchns       channel file type, bit 0 = drive
00241  0098            chnrdy *=*+mxchns       channel status
00242  00a0            eoiflg *=*+1            temp eoi, 0 = set
00243  00a1            jobnum *=*+1            current job number
00244  00a2            lintab *=*+maxsa+1      SA: LINDX table
00245  00b5            chndat *=*+mxchns       channel data byte
00246  00bd            lstchr *=*+mxchns       channel last character pointer
00247  00c5            type   *=*+1            active file type
00248  00c6
00249  00c6
00250  00c6  ===> RAM Variables in $4300 <===
00251  00c6
00252  00c6                   *=$4300
00253  4300            cmdbuf *=*+cmdlen       command buffer
00254  433a            strsiz *=*+1            command string size
00255  433b            tempsa *=*+1            temporary SA
00256  433c            cmd    *=*+1            temporary job command
00257  433d            lstsec *=*+1
00258  433e            bufuse *=*+2            buffer allocation
00259  4340            dskid  *=*+4            current disk IDs
00260  4344            secinc *=*+1            sector increment for seq
00261  4345            entfnd *=*+1            directory entry found (0) flag
00262  4346            dirlst *=*+1            directory listen flag
00263  4347            cmdwat *=*+1            command waiting flag
00264  4348            linuse *=*+1            LINDX use word
00265  4349            lbused *=*+1            last buffer used
00266  434a            erblks *=*+1            blocks before abort
00267  434b            rec    *=*+1            record size, used by directory
                                               routines
00268  434c            trkss  *=*+1            side sector track for directory
                                               routines
00269  434d            secss  *=*+1            side sector sector for directory
                                               routines
00270  434e
00271  434e
00272  434e  ===> RAM Array Area <===
00273  434e
00274  434e            lstjob *=*+bfcnt+2      last job
00275  435c            revcnt *=*+1            errors recovery count
00276  435d            errcnt *=*+bfcnt+2      error count on job
00277  436b            dirent *=*+mxchns       directory entry
00278  4373            erword *=*+1            error word for recovery
00279  4374            prgsec *=*+1            last program sector
00280  4375            wlindx *=*+1            write lindx
00281  4376            rlindx *=*+1            read lindx
```

```
line   addr  object     source code

00282  4377             nbtemp *=*+2          temporary number of blocks
00283  4379             cmdsiz *=*+1          command string size
00284  437a             cmdnum *=*+1          command number
00285  437b             char   *=*+1          character under parser
00286  437c             limit  *=*+1          pointer limit in comparing
00287  437d             f1cnt  *=*+1          file stream 1 count
00288  437e             f2cnt  *=*+1          file stream 2 count
00289  437f             f2ptr  *=*+1          file stream 2 pointer
00290  4380
00291  4380
00292  4380  ===> Parser Tables <===
00293  4380
00294  4380             filtbl *=*+mxfils+1   filename pointer
00295  4386             filtrk *=*+mxfils     first link/track
00296  438b             filsec *=*+mxfils     first link/sector
00297  4390
00298  4390
00299  4390  ===> Channel Tables <===
00300  4390
00301  4390             patflg *=*+1          pattern present flag
00302  4391             image  *=*+1          file stream image
00303  4392             drvcnt *=*+1          number of drive searches
00304  4393             drvflg *=*+1          drive search flag
00305  4394             lstdrv *=*+1          last drive without error
00306  4395             found  *=*+1          found flag in directory searches
00307  4396             dirsec *=*+1          directory sector
00308  4397             delsec *=*+1          sector of first available entry
00309  4398             delind *=*+1          index of first available entry
00310  4399             lstbuf *=*+1          = 0 if last block
00311  439a             index  *=*+1          current index in buffer
00312  439b             filcnt *=*+1          counter, file entries
00313  439c             typflg *=*+1          match by type flag
00314  439d             mode   *=*+1          active file mode (read/write)
00315  439e             jobrtn *=*+1          job return flag
00316  439f
00317  439f
00318  439f  ===> RAM in Bitmap Buffers <===
00319  439f
00320  439f                    *=$4100+180
00321  41b4             nambuf *=*+36         directory buffer
00322  41d8                    *=$4400-36
00323  43dc             errbuf =*             error message buffer
00323  43dc
00324  43dc                    .lib romtbl
```

```
line   addr  object      source code

00326  43dc                    *=rom

00328  d000  ************************************************
00329  d000  *                                              *
00330  d000  *                                              *
00331  d000  *     FORMAT code for controller resides here  *
00332  d000  *                                              *
00333  d000  *                                              *
00334  d000  ************************************************

00336  d000            code    *=*+$2a0
00337  d2a0
00338  d2a0  00        dchksm .byte 0          checksum d-rom
00339  d2a1
00340  d2a1
00341  d2a1  ===> Command Search Table <===
00342  d2a1
00343  d2a1  49 56 44  cmdtbl .byte 'ivdmbupcrsn'    init - drive verify - dir.
                                                    duplicate
00344  d2a4  4d 42 55
00345  d2a7  50 43 52
00346  d2aa  53 4e
00347  d2ac                                         memory - block - user - position
00348  d2ac                                         disk copy - rename - scratch - new
00349  d2ac
00350  d2ac            ncmds  = *-cmdtbl
00351  d2ac
00352  d2ac  ca f3 50  cjumpl .byte <intdrv, <verdir, <duplct jump table lo
00353  d2af  af b6 0f         .byte <mem, <block, <user, <record
00354  d2b2  ea
00355  d2b3  54 7c c1         .byte <dskcpy, <rename, <scrtch, <new
00356  d2b6  17
00357  d2b7
00358  d2b7  ec e6 e3  cjumph .byte >intdrv, >verdir, >duplct jump table hi
00359  d2ba  e7 e8 e8         .byte >mem, >block, >user, >record
00360  d2bd  fc
00361  d2be  e4 e6 e2         .byte >dskcpy, >rename, >scrtch, >new
00362  d2c1  e2
00363  d2c2
00364  d2c2                   * = cjumph+ncmds
00365  d2c2
00366  d2c2            val    = 1              validate (verify) cmd#
00367  d2c2
00368  d2c2
00369  d2c2  ===> Structure images for commands <===
00370  d2c2
00371  d2c2            pcmd   = 8
00372  d2c2  51               .byte %01010001  disk copy
00373  d2c3            struct = *-pcmd         commands not parsed
00374  d2c3  dd               .byte %11011101  rename
00375  d2c4  1c               .byte %00011100  scratch
00376  d2c5  9e               .byte %10011110  new
00377  d2c6            ldcmd  = *-struct       load command image
```

```
line   addr  object     source code

00378  d2c6  1c                .byte %00011100  load
00379  d2c7                            --- ---
00380  d2c7                            pgdrpgdr
00381  d2c7                            FS1 FS2
00382  d2c7
00383  d2c7  ------------------------------------------------------
00384  d2c7  bit representations:      p  not pattern
00385  d2c7                            g  not greater than one file
00386  d2c7                            d  no default drive(s)
00387  d2c7                            r  requested filename
00388  d2c7  ------------------------------------------------------
00389  d2c7
00390  d2c7  11 18 1e   trktbl .byte 17,24,30,37       track/group table
00391  d2ca  25
00392  d2cb  52 57 41   modlst .byte 'rwam'      mode table: read, write, append,
                                                 modify
00393  d2ce  4d
00394  d2cf             nmodes = *-modlst
00395  d2cf  44 53 50   tplst  .byte 'dspul'     file type table
00396  d2d2  55 4c
00397  d2d4  44 53 50   typlst .byte 'dspur'     DEL, SEQ, PRG, USR, REL
00398  d2d7  55 52
00399  d2d9             ntypes =*-typlst
00400  d2d9  45 45 52   tp1lst .byte 'eerse'
00401  d2dc  53 45
00402  d2de  4c 51 47   tp2lst .byte 'lqgrl'
00403  d2e1  52 4c
00404  d2e3
00405  d2e3  00         er00   .byte 0           error flag variables for BIT
00406  d2e4  3f         er0    .byte $3f
00407  d2e5  7f         er1    .byte $7f
00408  d2e6  bf         er2    .byte $bf
00409  d2e7  ff         er3    .byte $ff
00410  d2e8
00411  d2e8  41 42      ipbm   .byte $41, $42
00412  d2ea
00413  d2ea  11 12 13   sectrk .byte 17, 18, 19, 21, 9, 2, fm2040
00414  d2ed  15 09 02
00415  d2f0  41
00416  d2f1  0e 0f 10           .byte 14, 15, 16, 18, 28, 30, fm2030
00417  d2f4  12 1c 1e
00418  d2f7  42
00419  d2f8
00420  d2f8
00421  d2f8                    *=sectrk+ctbsiz+ctbsiz
00422  d2f8             rom1   = *
00422  d2f8
00423  d2f8                    .lib diskint
```

```
line   addr  object      source code

00425  d2f8  ===> Error display routine <===
00426  d2f8        blinks the (error number)+1 in all three LEDs
00427  d2f8
00428  d2f8  78          tabjmp sei
00429  d2f9  a9 00               lda #0
00430  d2fb  8d 03 04            sta $403
00431  d2fe  4c 04 fc            jmp $fc04        to wait loop
00432  d301
00433  d301
00434  d301  ===> Flash LED to signal error <===
00435  d301
00436  d301  a2 00       pezro   ldx #0           no error status entry
00437  d303  2c                  .byte $2c
00438  d304  a6 04       perr    ldx temp         temporary work area holds error
                                                  number
00439  d306  9a                  txs              use the stack as a storage register
00440  d307  ba          pe20    tsx
00441  d308  a9 38       pe30    lda #errled+led0+led1
00442  d30a
00443  d30a  ------------------------------------------------
00444  d30a  The LED mask $38 is made up as follows:
00445  d30a      ERRLED = 32 ($20)  - hardware init error led
00446  d30a    + LED0   =     $10   - active LED 0
00447  d30a    + LED1   =     $08   - active LED 1
00448  d30a  ------------------------------------------------
00449  d30a
00450  d30a  8d 82 02            sta pbd2         turn on all three LEDs
00451  d30d  98                  tya              clear the inner counter
00452  d30e  18          pd10    clc
00453  d30f  69 01       pd20    adc #1           count the inner counter until zero
00454  d311  d0 fc               bne pd20
00455  d313  88                  dey              when the hi byte of the timer
                                                  reaches zero,
00456  d314  d0 f8               bne pd10
00457  d316  8c 82 02            sty pbd2         turn off all LEDs
00458  d319              pe40                     wait
00459  d319  98                  tya              clear the
00460  d31a  18          pd11    clc              inner counter
00461  d31b  69 01       pd21    adc #1           wait for it to reach zero
00462  d31d  d0 fc               bne pd21
00463  d31f  88                  dey
00464  d320  d0 f8               bne pd11
00465  d322  ca                  dex              have we blinked?
00466  d323  10 e3               bpl pe30         no, blink again
00467  d325  e0 fc               cpx #$fc         have we waited long enough between
                                                  flashes?
00468  d327  d0 f0               bne pe40         if not, wait some more,
00469  d329  f0 dc               beq pe20         else repeat the entire sequence
00470  d32b
00471  d32b
```

```
line   addr  object     source code

00472  d32b  ===> Disk initialization routine <===
00473  d32b
00474  d32b  78         dskint sei              prevent interrupts
00475  d32c  d8                cld
00476  d32d  a2 ff             ldx #$ff        enable output:
00477  d32f  8e 02 02          stx ieeedo      IEEE data out
00478  d332  8e 03 02          stx pbdd1       IEEE data out direction
00479  d335  e8                inx             .X=0
00480  d336  8e 82 02          stx pbd2
00481  d339  a9 1c             lda #davo+eoio+rfdo
00482  d33b  8d 80 02          sta pad2        IEEE control port
00483  d33e  a9 1f             lda #$1f
00484  d340  8d 81 02          sta padd2
00485  d343  a9 38             lda #errled+led0+led1
00486  d345  8d 83 02          sta pbdd2
00487  d348
00488  d348  ====> Power-Up Diagnostic <====
00489  d348
00490  d348  8a         pu10   txa             fill zero page with ascending
                                               pattern
00491  d349  95 00             sta $0,x
00492  d34b  e8                inx
00493  d34c  d0 fa             bne pu10
00494  d34e  8a         pu20   txa             get .X into .Y
00495  d34f  a8                tay
00496  d350  c8                iny             start .Y counter one ahead of memory
00497  d351  f6 00      pu30   inc $0,x        bump memory around to $00
00498  d353  c8                iny             do the same with .Y
00499  d354  d0 fb             bne pu30        we're not there yet!
00500  d356  b4 00             ldy $0,x        check if memory at $ff
00501  d358  c8                iny
00502  d359  d0 a6             bne pezro       no -- something is wrong, so show
                                               error number
00503  d35b  f6 00             inc $0,x        $ff now, so bump memory to $00
00504  d35d  d0 a2             bne pezro       not zero -- show error number
00505  d35f  e8                inx             check next memory location
00506  d360  d0 ec             bne pu20
00507  d362
00508  d362  Test the three file side ROMS. On entry, .X is start of page. Exit if
                                               o.k.
00509  d362  e6 04      rm10   inc temp        next error number (1 - 3 for ROMs D
                                               - F)
00510  d364  86 0b             stx ip+1        save the page number as hi byte of
                                               pointer
00511  d366  a9 00             lda #0          zeroize
00512  d368  85 0a             sta ip          the lo byte
00513  d36a  a8                tay
00514  d36b  a2 10             ldx #$10        16 pages in 4-K ROM
00515  d36d  18                clc
00516  d36e  c6 0b      rt10   dec ip+1        let's do it backwards:
00517  d370  71 0a      rt20   adc (ip),y      add the ROM value to the contents of
                                               .A,
00518  d372  c8                iny             increment the pointer and until it's
                                               zero
```

```
line   addr  object      source code

00519  d373  d0 fb             bne rt20          branch back to RT20 to do another
                                                 byte
00520  d375  ca                dex               then do likewise with the other
                                                 pages
00521  d376  d0 f6             bne rt10
00522  d378  69 00             adc #0            add in the last carry
00523  d37a  aa                tax               transfer the checksum
00524  d37b  c5 0b             cmp ip+1          and compare it with the hi byte of
                                                 the count
00525  d37d  d0 85       perr2 bne perr          if they don't match, report error
00526  d37f  e0 d0             cpx #jumpc        all three done?
00527  d381  d0 df             bne rm10
00528  d383
00529  d383              rcon1                   Error Display Routine
00530  d383  b9 f8 d2          lda tabjmp,y      transfer "jump to wait" loop
00531  d386  99 00 11          sta bufs,y        to buffer at $1100
00532  d389  c8                iny
00533  d38a  d0 f7             bne rcon1
00534  d38c  a9 d0             lda #jumpc        send job type (jump) to
00535  d38e  8d 03 10          sta jobs          job queue
00536  d391  e6 04             inc temp
00537  d393  c8          cdelay iny
00538  d394  d0 fd             bne cdelay
00539  d396  ad 03 10          lda jobs          job queue definitions
00540  d399  f0 05             beq cr20
00541  d39b  ca                dex
00542  d39c  30 f5             bmi cdelay
00543  d39e  d0 dd             bne perr2
00544  d3a0
00545  d3a0
00546  d3a0  ===> Test disk RAM except page $1000 <===
00547  d3a0
00548  d3a0  a9 10       cr20  lda #16           save start of first block (page
                                                 number)
00549  d3a2  85 0b       cr30  sta ip+1          as hi pointer
00550  d3a4  e6 04             inc temp          bump the error number ($03 is
                                                 RAM problem)
00551  d3a6
00552  d3a6                                     On entry, .X contains number of
                                                 pages in block
00553  d3a6                                        if pointer to first page
00554  d3a6                                     Exit if o.k.
00555  d3a6
00556  d3a6  a2 04       ramtst ldx #4           save page count
00557  d3a8  98          ra10   tya              fill with address sensitive pattern
00558  d3a9  18                 clc
00559  d3aa  65 0b              adc ip+1          add counter hi to accumulator and
                                                 store
00560  d3ac  91 0a              sta (ip),y
00561  d3ae  c8                 iny
00562  d3af  d0 f7              bne ra10
00563  d3b1  e6 0b              inc ip+1          if .Y is zero, first increment the
                                                 hi pointer
```

```
line   addr  object     source code

00564  d3b3  ca                dex             then decrement the page count
00565  d3b4  d0 f2             bne ra10        and repeat until zero
00566  d3b6  a2 04             ldx #4          restore the page count
00567  d3b8  c6 0b       ra30  dec ip+1        check the pattern backwards
00568  d3ba  88          ra40  dey
00569  d3bb  98                tya             generate the pattern again
00570  d3bc  18                clc
00571  d3bd  65 0b             adc ip+1
00572  d3bf  d1 0a             cmp (ip),y      if not OK,
00573  d3c1  d0 ba             bne perr2       report error
00574  d3c3  49 ff             eor #$ff        now test the reverse pattern
00575  d3c5  91 0a             sta (ip),y
00576  d3c7  51 0a             eor (ip),y      result should be $00 -- is it?
00577  d3c9  91 0a             sta (ip),y
00578  d3cb  d0 b0             bne perr2       no -- report error
00579  d3cd  98                tya             if .Y not 0, we have
00580  d3ce  d0 ea             bne ra40        more to do on this page
00581  d3d0  ca                dex             any pages left?
00582  d3d1  d0 e5             bne ra30
00583  d3d3  a5 0b             lda ip+1        get first page of block
00584  d3d5  18                clc
00585  d3d6  69 10             adc #$10        no -- next block
00586  d3d8  c9 50             cmp #$50        are we done?
00587  d3da  d0 c6             bne cr30        no
00588  d3dc
00589  d3dc
00590  d3dc  ===>  Controller test  <===
00591  d3dc
00592  d3dc  a2 ff       diagok ldx #$ff        reset
00593  d3de  9a                 txs             the stack
00594  d3df  ad 82 02           lda pbd2        clear LEDs
00595  d3e2  29 c7              and #255-errled-led0-led1
00596  d3e4  8d 82 02           sta pbd2        turn it off
00597  d3e7  ad 82 02           lda pbd2        compute primary address
00598  d3ea  29 07              and #%00000111  mask device number
00599  d3ec  09 48              ora #%01001000  set bit 3 and talk flag
00600  d3ee  85 0d              sta tlkadr      talker address
00601  d3f0  49 60              eor #%01100000  clear talk, set listen
00602  d3f2  85 0c              sta lsnadr      listener address
00603  d3f4
00604  d3f4  a2 00       inttab ldx #0         initialize buffer pointer table
00605  d3f6  a0 00              ldy #0
00606  d3f8  a9 00       intt1  lda #0
00607  d3fa  95 29              sta buftab,x    buffer 0 pointer lo
00608  d3fc  e8                 inx
00609  d3fd  b9 ff f0           lda bufind,y    hi byte table of pointers to data
                                                buffer
00610  d400  95 29              sta buftab,x    buffer 0 pointer lo
00611  d402  e8                 inx
00612  d403  c8                 iny
00613  d404  c0 0e              cpy #bfcnt+2    14 buffers
00614  d406  d0 f0              bne intt1       if more buffers to do.
00615  d408  a9 00              lda #<cmdbuf    set pointers to command buffer
                                                at$4300
```

```
line   addr  object     source code

00616  d40a  95 29            sta buftab,x      buffer 0 pointer lo
00617  d40c  e8              inx
00618  d40d  a9 43            lda #>cmdbuf
00619  d40f  95 29            sta buftab,x
00620  d411  e8              inx
00621  d412  a9 dc            lda #<errbuf      set pointers to error buffer at
                                                 $43dc
00622  d414  95 29            sta buftab,x
00623  d416  e8              inx
00624  d417  a9 43            lda #>errbuf
00625  d419  95 29            sta buftab,x
00626  d41b  a9 ff            lda #$ff
00627  d41d  a2 12            ldx #maxsa
00628  d41f  95 a2     dskin1 sta lintab,x      make all SAs inactive
00629  d421  ca              dex
00630  d422  10 fb            bpl dskin1
00631  d424  a2 07            ldx #mxchns-1     set maximum channels minus 1 as
                                                 unused
00632  d426  95 49     dskin2 sta buf0,x        channel buffer table 1
00633  d428  95 51            sta buf1,x        channel buffer table 2
00634  d42a  95 79            sta ss,x          side sectors table
00635  d42c  ca              dex
00636  d42d  10 f7            bpl dskin2
00637  d42f  a9 0e            lda #bfcnt+2      set buffer pointers
00638  d431  85 4f            sta buf0+cmdchn
00639  d433  a9 0f            lda #bfcnt+3
00640  d435  85 50            sta buf0+errchn
00641  d437  a9 07            lda #errchn
00642  d439  85 b2            sta lintab+errsa
00643  d43b  a9 86            lda #cmdchn+$80   channel 6
00644  d43d  85 b1            sta lintab+cmdsa
00645  d43f  a9 3f            lda #lxint        LINDX 0 to 5 free
00646  d441  8d 48 43         sta linuse        LINDX use word
00647  d444  a9 01            lda #rdylst
00648  d446  85 9e            sta chnrdy+cmdchn
00649  d448  a9 88            lda #rdytlk
00650  d44a  85 9f            sta chnrdy+errchn
00651  d44c  a9 00            lda #0            set up
00652  d44e  8d 3e 43         sta bufuse        buffer allocation register
00653  d451  a9 f0            lda #$f0
00654  d453  8d 3f 43         sta bufuse+1
00655  d456  20 16 e8         jsr usrint        u0 points to ($00)
00656  d459  a9 dc            lda #<diagok
00657  d45b  8d f0 10         sta vnmi          indirect for NMI to
00658  d45e  a9 d3            lda #>diagok      point to diagnostic
00659  d460  8d f1 10         sta vnmi+1        routine
00660  d463  a9 0a            lda #10           normal sector offset
00661  d465  8d 44 43         sta secinc        increment between sectors
00662  d468  8d 5c 43         sta revcnt        error recovery counter
00663  d46b
00664  d46b
00665  d46b
```

```
line   addr  object     source code

00666  d46b  ===> Set up sectors per track depending on resident controller <===
00667  d46b
00668  d46b  ad 00 10   setsec lda id           look at controller ID byte
00669  d46e  a2 00             ldx #0           is it
00670  d470  c9 0f             cmp #id2040      a 2040?
00671  d472  f0 0b             beq sets30       jump to load table
00672  d474  a2 07      sets10 ldx #$07         is it a
00673  d476  c9 64             cmp #id2030      3040 or 4040?
00674  d478  f0 05             beq sets30       jump to load table
00675  d47a  e6 04      sets20 inc temp
00676  d47c  4c 04 d3          jmp perr         not a good controller
00677  d47f
00678  d47f  a0 00      sets30 ldy #0           set up table
00679  d481  bd ea d2   sets40 lda sectrk,x     sectors/track for formatting
00680  d484  99 99 10          sta numsec,y     number of sectors per track
00681  d487  e8                inx
00682  d488  c8                iny
00683  d489  c0 07             cpy #ctbsiz      have we done 7?
00684  d48b  d0 f4             bne sets40       not yet
00685  d48d
00686  d48d
00687  d48d  ===> Set up Power On error message <===
00688  d48d
00689  d48d  a9 73      seterr lda #cbmv2       ID mismatch
00690  d48f  20 d7 d9          jsr errts0       error handling
00691  d492
00692  d492
00693  d492  ===> Power On bump <===
00694  d492
00695  d492             ponbmp
00696  d492  a9 01             lda #1           track 1
00697  d494  8d 23 10          sta $1023        HDRS
00698  d497  8d 2b 10          sta $102b        initialize track in header table
00699  d49a  a9 c0             lda #bump        bump
00700  d49c  8d 03 10          sta $1003        drive 0
00701  d49f  a9 c1             lda #bump+1      bump
00702  d4a1  8d 04 10          sta $1004        drive 1
00703  d4a4  8d 87 02          sta atnpe        allow ATN to interrupt
00704  d4a7
00705  d4a7
00706  d4a7  ===> Running idle, waiting for something to do <===
00707  d4a7
00708  d4a7  ad 47 43   idle   lda cmdwat       if no command waiting,
00709  d4aa  f0 0c             beq idle2        Test for drive running or open
00710  d4ac  78                sei
00711  d4ad  a9 00             lda #0
00712  d4af  8d 47 43          sta cmdwat       command waiting flag
00713  d4b2  8d f2 10          sta nmiflg       clear debounce
00714  d4b5  20 5b db          jsr parsxq       Parse & execute string in command
                                                buffer
00715  d4b8
00716  d4b8
00717  d4b8
```

```
line   addr  object     source code

00718  d4b8  ===> Test for drive running or open <===
00719  d4b8
00720  d4b8  58         idle2  cli              allow interrupts
00721  d4b9  a9 0e             lda #14          highest possible SA for files
00722  d4bb  85 07             sta t3
00723  d4bd  a9 00             lda #0           if file open, turn on active LED
00724  d4bf  85 04             sta temp
00725  d4c1  85 05             sta t1
00726  d4c3  a6 07       file01 ldx t3
00727  d4c5  b5 a2             lda lintab,x     current status SA
00728  d4c7  c9 ff             cmp #$ff         look for an active file. FF means
                                                none
00729  d4c9  f0 10             beq file02
00730  d4cb  29 3f             and #%00111111   active file found, so AND and store
                                                result as
00731  d4cd  85 15             sta lindx        the current channel number
00732  d4cf  20 95 fa          jsr getact       Get active buffer number
00733  d4d2  aa                tax
00734  d4d3  bd 4e 43          lda lstjob,x     find out which drive it is on
00735  d4d6  29 01             and #1
00736  d4d8  aa                tax              and store in .X
00737  d4d9  f6 04             inc temp,x       then add to the count of active
                                                files on drive X
00738  d4db  c6 07       file02 dec t3           set flag indicating drive has file
                                                open
00739  d4dd  10 e4             bpl file01       if more secondary addresses left to
                                                check.
00740  d4df  a0 0b       tstfil ldy #bfcnt-1     look through job queue for
00741  d4e1  b9 03 10    fil5   lda jobs,y       jobs pending
00742  d4e4  10 05             bpl fil6         if bit 7 not set, no job in
                                                progress.
00743  d4e6  29 01             and #1           mask of the non-drive bits in the
                                                job code
00744  d4e8  aa                tax
00745  d4e9  f6 04             inc temp,x       set flag indicating drive is active
00746  d4eb  88          fil6  dey
00747  d4ec  10 f3             bpl fil5         go check more buffers if any left,
                                                else
00748  d4ee  ad 82 02          lda pbd2         fetch data byte from port
                                                controlling LED
00749  d4f1  29 e7             and #255-led1-led0
00750  d4f3  48                pha
00751  d4f4  a5 04             lda temp         test active file count on drive 0
00752  d4f6  f0 04             beq fil03
00753  d4f8  68                pla              turn on LED if drive flag
00754  d4f9  09 10             ora #%10000      if not 0
00755  d4fb  48                pha
00756  d4fc  a5 05       fil03 lda t1           test active file count on drive 1
00757  d4fe  f0 04             beq fil04
00758  d500  68                pla
00759  d501  09 08             ora #%1000
00760  d503  48                pha
00761  d504  68          fil04 pla
```

```
line   addr  object     source code

00762  d505  8d 82 02          sta pbd2
00763  d508  4c b8 d4          jmp idle2        back to top of loop
00763  d50b
00764  d50b                    .lib ieee
```

```
line   addr  object      source code

00766  d50b  ===> Main system IRQ routine - IRQ vector points here <===
00767  d50b
00768  d50b  a2 ff       atnirq ldx #$ff          clear the stack
00769  d50d  9a                 txs
00770  d50e  ad 87 02           lda atnpe         clear IRQ flag
00771  d511  a9 18              lda #davo+eoio
00772  d513  0d 80 02           ora pad2          free control lines
00773  d516  8d 80 02           sta pad2          send out DAV & EOI
00774  d519  a9 ff              lda #$ff
00775  d51b  8d 02 02           sta ieeedo        free data lines
00776  d51e  a9 07       atn10  lda #daco+rfdo+atna
00777  d520  0d 80 02           ora pad2
00778  d523  8d 80 02           sta pad2
00779  d526  2c 80 02    atn20  bit pad2          wait for DAV received
00780  d529  50 04              bvc atn30         DAV lo
00781  d52b  30 f9              bmi atn20         ATN lo, ATNI hi
00782  d52d  10 7b              bpl atn50         ATN hi
00783  d52f  a9 fb       atn30  lda #$ff-rfdo     NRFD lo
00784  d531  2d 80 02           and pad2
00785  d534  8d 80 02           sta pad2
00786  d537  29 20              and #eoii         save EOI
00787  d539  85 a0              sta eoiflg        current EOI status
00788  d53b  ad 00 02           lda ieeedi        IEEE data in, now invert
00789  d53e  49 ff              eor #$ff          this byte and store as
00790  d540  85 18              sta data          command
00791  d542  a9 fd              lda #$ff-daco     send NDAC
00792  d544  2d 80 02           and pad2
00793  d547  8d 80 02           sta pad2          and send it
00794  d54a  a0 00       dcde   ldy #0
00795  d54c  a5 18              lda data          temporary data byte
00796  d54e  29 60              and #%01100000    mask bits 5 and 6
00797  d550  c9 40              cmp #$40          talk?         (5)
00798  d552  f0 29              beq dcde60
00799  d554  c9 20              cmp #$20          listen?       (6)
00800  d556  f0 06              beq dcde20
00801  d558  c9 60              cmp #$60          secondary?    (5 and 6)
00802  d55a  f0 2f              beq dcde70        (note: SA = $60 + N)
00803  d55c  d0 44              bne atn40         other, so ignore it
00804  d55e  a5 18       dcde20 lda data          temporary data byte
00805  d560  c5 0c              cmp lsnadr        listener address
00806  d562  f0 0b              beq dcde40        is mine
00807  d564  c9 3f              cmp #unlsn
00808  d566  d0 02              bne dcde30
00809  d568  84 0e              sty lsnact        active listener flag
00810  d56a  84 10       dcde30 sty adrsed        not primary addressed
00811  d56c  4c a2 d5           jmp atn40         wait for end of DAV, then return to
                                                  main
00812  d56f
00813  d56f  85 0e       dcde40 sta lsnact        active listener flag
00814  d571  84 0f              sty tlkact        active talker flag
00815  d573  a9 20       dcde50 lda #32
00816  d575  85 16              sta sa            current secondary address = default
00817  d577  85 17              sta orgsa         original secondary address
```

```
line   addr  object     source code

00818  d579  85 10            sta adrsed       primary addressed
00819  d57b  d0 25             bne atn40
00820  d57d  84 0f      dcde60 sty tlkact       active talker flag
00821  d57f  a5 18             lda data         temporary data byte
00822  d581  c5 0d             cmp tlkadr       talker address?
00823  d583  d0 e5             bne dcde30
00824  d585  85 0f             sta tlkact       active talker flag
00825  d587  84 0e             sty lsnact       active listener flag
00826  d589  f0 e8             beq dcde50       always
00827  d58b  a5 10      dcde70 lda adrsed       not addressed
00828  d58d  f0 13             beq atn40
00829  d58f  a5 18             lda data         temporary data byte
00830  d591  85 17             sta orgsa        original secondary address
00831  d593  48                pha
00832  d594  29 0f             and #$0f         use the lo nibble as
00833  d596  85 16             sta sa           current secondary address
00834  d598  68                pla
00835  d599  29 f0             and #$f0         mask hi nibble
00836  d59b  c9 e0             cmp #$e0         is it a CLOSE command?
00837  d59d  d0 03             bne atn40        not a CLOSE command
00838  d59f  20 8d f5          jsr close        Close the file related to the
                                                 specified sec. address
00839  d5a2                dcde80               wait for end of DAV
00840  d5a2  2c 80 02   atn40  bit pad2
00841  d5a5  50 fb             bvc atn40
00842  d5a7  4c 1e d5          jmp atn10        go back for more
00843  d5aa
00844  d5aa  a5 0e      atn50  lda lsnact       active listener flag
00845  d5ac  f0 0f             beq atn60        no, we're talking, so send
00846  d5ae  a9 fa             lda #$ff-rfdo-atna     NATN, NRFD
00847  d5b0  2d 80 02          and pad2
00848  d5b3  8d 80 02          sta pad2
00849  d5b6  58                cli
00850  d5b7  20 d0 d5          jsr listen       open the read channel, send data
00851  d5ba  4c a7 d4          jmp idle         then go to idle
00852  d5bd
00853  d5bd  a9 fc      atn60  lda #$ff-atna-daco
00854  d5bf  2d 80 02          and pad2
00855  d5c2  8d 80 02          sta pad2
00856  d5c5  a5 0f             lda tlkact
00857  d5c7  f0 04             beq atn70        if listen mode, go back to idle!
00858  d5c9  58                cli
00859  d5ca  20 60 d6          jsr talk         be an active talker
00860  d5cd  4c a7 d4   atn70  jmp idle
00861  d5d0
00862  d5d0  a9 04      listen lda #rfdo        RFD: hi
00863  d5d2  0d 80 02          ora pad2
00864  d5d5  8d 80 02          sta pad2
00865  d5d8  2c 80 02   lsn10  bit pad2         DAV: lo
00866  d5db  70 fb             bvs lsn10        if not, wait
00867  d5dd  20 89 ed          jsr fndwch       Find an unused write channel
00868  d5e0                                     ==>> earlier releases may have "ldx
                                                 SA" here!
```

```
line   addr  object      source code

00869  d5e0  b0 05             bcs lsn15       branch if no channel found
00870  d5e2  b5 98             lda chnrdy,x    write, read, eoi flags, channel
                                                status
00871  d5e4  6a                ror a           OK, open for listen
00872  d5e5  b0 49             bcs lsn30       if carry set, write channel inactive
00873  d5e7  a5 17       lsn15 lda orgsa       original secondary address
00874  d5e9                                    ==>> earlier releases may have "txa"
                                                here!
00875  d5e9  29 f0             and #$f0        is it an OPEN command?
00876  d5eb  c9 f0             cmp #$f0        see if bits 4 and higher are set. If
                                                so,
00877  d5ed  f0 41             beq lsn30       we can expect a filename
00878  d5ef  a5 16       lsn20 lda sa          current secondary address
00879  d5f1  c9 01             cmp #$01        is it a SAVE?
00880  d5f3  f0 0e             beq lsn25       Accept all data
00881  d5f5  2c 80 02    lsn21 bit pad2        DAV received?
00882  d5f8  50 fb             bvc lsn21
00883  d5fa  a9 fd             lda #$ff-daco   send NDAC
00884  d5fc  2d 80 02          and pad2
00885  d5ff  8d 80 02          sta pad2
00886  d602  60                rts
00887  d603
00888  d603
00889  d603  ===> Accept all data <===
00890  d603
00891  d603  a9 fb       lsn25 lda #$ff-rfdo   send NRFD
00892  d605  2d 80 02          and pad2
00893  d608  8d 80 02          sta pad2        RFD lo
00894  d60b  a9 fd             lda #$ff-daco   send NDAC
00895  d60d  2d 80 02          and pad2        DAC hi
00896  d610  8d 80 02          sta pad2
00897  d613  2c 80 02    lsn26 bit pad2        DAV hi received?
00898  d616  50 fb             bvc lsn26       if yes, wait, else
00899  d618  a9 02             lda #$02        send DAC
00900  d61a  0d 80 02          ora pad2        DAV lo
00901  d61d  8d 80 02          sta pad2
00902  d620  a9 04             lda #rfdo       RFD hi
00903  d622  0d 80 02          ora pad2        send RFD
00904  d625  8d 80 02          sta pad2
00905  d628  2c 80 02    lsn28 bit pad2        wait for DAV lo
00906  d62b  50 fb             bvc lsn28
00907  d62d  4c 03 d6          jmp lsn25       do until ATN pulled
00908  d630
00909  d630  ===> data received <===
00910  d630
00911  d630  a9 fb       lsn30 lda #$ff-rfdo   send NRFD
00912  d632  2d 80 02          and pad2
00913  d635  8d 80 02          sta pad2
00914  d638  29 20             and #eoii       $20 to mask EOI, then save:
00915  d63a  85 a0             sta eoiflg      current EOI status
00916  d63c  ad 00 02          lda ieeedi      IEEE data
00917  d63f  49 ff             eor #$ff        inverted becomes
00918  d641  85 18             sta data        temporary data byte
```

```
line   addr  object     source code

00919  d643  78               sei
00920  d644  a9 fd            lda #$ff-daco     send DAC
00921  d646  2d 80 02         and pad2
00922  d649  8d 80 02         sta pad2
00923  d64c  2c 80 02  lsn40  bit pad2          DAV received?
00924  d64f  50 fb            bvc lsn40         then wait
00925  d651  a9 02            lda #$02          send DAC
00926  d653  0d 80 02         ora pad2
00927  d656  8d 80 02         sta pad2
00928  d659  20 fd eb         jsr put           put data byte in its proper place
                                                (DATA, EOI, SA)
00929  d65c  58               cli
00930  d65d  4c d0 d5         jmp listen        keep on listening
00931  d660
00932  d660
00933  d660  ===> Talk routines <===
00934  d660
00935  d660  20 6e ed  talk   jsr fndrch        Find the assigned read channel
00936  d663  b0 06            bcs notlk         and see if it is active. If not: set
00937  d665  a6 15      talk1 ldx lindx         logical index, channel number
00938  d667  b5 98            lda chnrdy,x      write, read, eoi flags, channel
                                                status
00939  d669  30 01            bmi tlk10         if READ flag not set, channel not
                                                ready and
00940  d66b  60        notlk  rts               we're not talking
00941  d66c
00942  d66c  2c 82 02  tlk10  bit pbd2          Wait to receive
00943  d66f  10 fb            bpl tlk10         RFD
00944  d671  b5 b5            lda chndat,x      channel data byte
00945  d673  49 ff            eor #$ff          inverted to
00946  d675  8d 02 02         sta ieeedo        IEEE data out
00947  d678  b5 98            lda chnrdy,x      write, read, eoi flags, channel
                                                status
00948  d67a  09 e7            ora #$ff-eoio-davo     when 7 set, DAV & EOI --
                                                status ok
00949  d67c  2d 80 02         and pad2          bit 3 = 0: EOI set
00950  d67f  8d 80 02         sta pad2
00951  d682  2c 82 02  tlk20  bit pbd2          NRFD?
00952  d685  10 0d            bpl tlk30
00953  d687  50 f9            bvc tlk20         NDAC?
00954  d689  a9 18            lda #davo+eoio    NDAV, NEOI
00955  d68b  0d 80 02         ora pad2
00956  d68e  8d 80 02         sta pad2
00957  d691  4c a7 d4         jmp idle
00958  d694
00959  d694  20 a3 ef  tlk30  jsr get           Get next byte from any type of file
00960  d697  2c 82 02  tlk35  bit pbd2          wait for NDAC
00961  d69a  50 fb            bvc tlk35
00962  d69c  a9 ff            lda #$ff          reset
00963  d69e  8d 02 02         sta ieeedo        IEEE data output register
00964  d6a1  a9 18            lda #davo+eoio    send NDAV, NEOI
00965  d6a3  0d 80 02         ora pad2
00966  d6a6  8d 80 02         sta pad2
```

```
line   addr  object      source code

00967  d6a9  2c 82 02   tlk40  bit pbd2          wait for DAV
00968  d6ac  70 fb              bvs tlk40
00969  d6ae  50 b5              bvc talk1
00970  d6b0
00971  d6b0
00972  d6b0  ===> Find the next available track and sector <===
00973  d6b0
00974  d6b0  20 3e f9   nxtts  jsr gethdr        set track/sector from most recent
                                                 header
00975  d6b3  a9 03              lda #$03          counter
00976  d6b5  85 04              sta temp          temporary work area
00977  d6b7  a6 12       nxtds  ldx drvnum        current drive#
00978  d6b9  bd e8 d2           lda ipbm,x        BAM address hi
00979  d6bc  85 03              sta bmpnt+1
00980  d6be  a9 00              lda #0
00981  d6c0  85 02              sta bmpnt         bit map pointer
00982  d6c2  a5 13       nxt1   lda track         current track number
00983  d6c4  0a                 asl a
00984  d6c5  0a                 asl a
00985  d6c6  a8                 tay
00986  d6c7  b1 02              lda (bmpnt),y     bit map pointer. If <>0 there are no
                                                 free sectors
00987  d6c9  d0 33              bne fndnxt        Find the next optimum sector
00988  d6cb  a5 13              lda track         current track number
00989  d6cd  c9 12              cmp #$12          is it the directory track?
00990  d6cf  f0 16              beq nxterr        then abort
00991  d6d1  90 19              bcc nxt2          (if smaller than 18)
00992  d6d3  e6 13              inc track         current track number
00993  d6d5  a5 13              lda track         current track number
00994  d6d7  c9 24              cmp #maxtrk       36, highest track number
00995  d6d9  d0 e7              bne nxt1          if unequal, check it out
00996  d6db  a9 11              lda #$11
00997  d6dd  85 13              sta track         current track number
00998  d6df  a9 00              lda #0
00999  d6e1  85 14              sta sector        current sector number
01000  d6e3  c6 04              dec temp          check if counter
01001  d6e5  d0 db              bne nxt1          is zero
01002  d6e7  a9 72       nxterr lda #dskful      disk is full!
01003  d6e9  4c c9 db           jmp cmderr        Command level error handling
01004  d6ec
01005  d6ec  c6 13       nxt2   dec track         current track number
01006  d6ee  d0 d2              bne nxt1          check it out if not 0
01007  d6f0  a9 13              lda #19
01008  d6f2  85 13              sta track         current track number
01009  d6f4  a9 00              lda #0
01010  d6f6  85 14              sta sector        current sector number
01011  d6f8  c6 04              dec temp          check if counter
01012  d6fa  d0 c6              bne nxt1          is zero
01013  d6fc  f0 e9              beq nxterr        if it is, report disk full
01014  d6fe
01015  d6fe
01016  d6fe
```

```
line   addr  object      source code

01017  d6fe  ===> Find the next optimum sector <===
01018  d6fe       next sector = current + N (increment)
01019  d6fe       n = normally 10. directory track n = 3
01020  d6fe
01021  d6fe  a5 14       fndnxt lda sector       current sector number
01022  d700  18                clc
01023  d701  6d 44 43          adc secinc       sector increment
01024  d704  85 14              sta sector       current sector number
01025  d706  a5 13              lda track        current track number
01026  d708  20 db d7           jsr maxsec       Tell how many sectors allowed for
                                                  this track
01027  d70b  8d 3d 43           sta lstsec       work area, best sector to do
01028  d70e  8d 3c 43           sta cmd          temporary job command
01029  d711  c5 14              cmp sector       current sector number
01030  d713  b0 12              bcs fndn0        (if new sector value is less than
                                                  the maximum)
01031  d715  38                 sec              new sector number is too big, so
                                                  load the
01032  d716  a5 14              lda sector       current sector number
01033  d718  ed 3d 43           sbc lstsec       subtract the maximum sector number
                                                  and store into
01034  d71b  85 14              sta sector       current sector number
01035  d71d  f0 08              beq fndn0        (if revised sector number is 0)
01036  d71f  c6 14              dec sector       current sector number
01037  d721  d0 04              bne fndn0
01038  d723  a9 00       fndn3  lda #0
01039  d725  85 14              sta sector       current sector number
01040  d727  20 95 d7    fndn0  jsr avail        Check BAM for available sector
01041  d72a  20 b1 d7    fndn1  jsr av2
01042  d72d  b0 15              bcs fndn2
01043  d72f  ce 3c 43           dec cmd          temporary job command
01044  d732  10 05              bpl fndn5
01045  d734  a9 71              lda #direrr      directory error
01046  d736  4c 5c d9           jmp cmder2
01047  d739
01048  d739  a5 14       fndn5  lda sector       current sector number
01049  d73b  e6 14              inc sector       current sector number
01050  d73d  cd 3d 43           cmp lstsec       best sector to do
01051  d740  d0 e8              bne fndn1
01052  d742  f0 df              beq fndn3
01053  d744  4c 9f eb    fndn2  jmp usedts       mark track & sector as used
01054  d747
01055  d747  ===> Find best initial track and sector <===
01056  d747
01057  d747  a9 11       intts  lda #17
01058  d749  85 13              sta track        current track number
01059  d74b  20 89 d7           jsr setbmp
01060  d74e  a5 13       loop   lda track        current track number
01061  d750  0a                 asl a
01062  d751  0a                 asl a
01063  d752  a8                 tay
01064  d753  b1 02              lda (bmpnt),y    bit map pointer
01065  d755  d0 15              bne fndsec
```

```
line   addr  object     source code

01066  d757  a9 24            lda #maxtrk
01067  d759  38               sec
01068  d75a  e5 13            sbc track        current track number
01069  d75c  0a               asl a
01070  d75d  0a               asl a
01071  d75e  a8               tay
01072  d75f  b1 02            lda (bmpnt),y    bit map pointer
01073  d761  d0 09            bne fndsec
01074  d763  c6 13            dec track        current track number
01075  d765  d0 e7            bne loop
01076  d767  a9 72            lda #dskful
01077  d769  4c c9 db         jmp cmderr       Command level error handling
01078  d76c
01079  d76c  98         fndsec tya              pull original value
01080  d76d  4a               lsr a
01081  d76e  4a               lsr a
01082  d76f  85 13            sta track        current track number
01083  d771  a9 00            lda #0
01084  d773  85 14            sta sector       current sector number
01085  d775  20 95 d7         jsr avail        Check BAM for available sector
01086  d778  20 b1 d7   fnd1   jsr av2
01087  d77b  b0 09            bcs fnd3
01088  d77d  e6 14            inc sector       current sector number
01089  d77f  d0 f7            bne fnd1
01090  d781  a9 71            lda #direrr      error in BAM
01091  d783  4c 5c d9         jmp cmder2
01092  d786
01093  d786  4c 9f eb   fnd3   jmp usedts       mark track & sector as used
01094  d789
01095  d789
01096  d789  ===> Set (indirect) BAM pointer by DRVNUM <===
01097  d789
01098  d789  a6 12      setbmp ldx drvnum       current drive number
01099  d78b  bd e8 d2         lda ipbm,x       BAM address hi
01100  d78e  85 03            sta bmpnt+1
01101  d790  a9 00            lda #0
01102  d792  85 02            sta bmpnt        bit map pointer
01103  d794  60               rts
01104  d795
01105  d795  ===> Check BAM for available sector <===
01106  d795
01107  d795  a5 13      avail  lda track        current track number
01108  d797  0a               asl a
01109  d798  0a               asl a
01110  d799  a8               tay
01111  d79a  b1 02            lda (bmpnt),y    bit map pointer
01112  d79c  85 07            sta t3
01113  d79e  a2 02            ldx #$02
01114  d7a0  c8         avl    iny
01115  d7a1  b1 02            lda (bmpnt),y    bit map pointer
01116  d7a3  95 04            sta temp,x       temporary work area
01117  d7a5  ca               dex
01118  d7a6  10 f8            bpl avl
```

```
line   addr  object      source code

01119  d7a8  20 bd d7          jsr avck          check validity of bit map
01120  d7ab  a4 14             ldy sector        current sector number
01121  d7ad  f0 0d             beq av4
01122  d7af  d0 02             bne av3
01123  d7b1  a0 01       av2   ldy #$01
01124  d7b3  66 04       av3   ror temp          temporary work area
01125  d7b5  66 05             ror t1
01126  d7b7  66 06             ror t2
01127  d7b9  88                dey
01128  d7ba  d0 f7             bne av3
01129  d7bc  60          av4   rts
01130  d7bd
01131  d7bd
01132  d7bd  ===> check bit map validity <===
01133  d7bd
01134  d7bd  a2 00       avck  ldx #0
01135  d7bf  a0 03             ldy #$03          number of bytes
01136  d7c1  d0 06             bne avck5         always
01137  d7c3  e8          avck3 inx
01138  d7c4  4a          avck4 lsr a
01139  d7c5  b0 fc             bcs avck3
01140  d7c7  d0 fb             bne avck4
01141  d7c9  b9 03 00    avck5 lda bmpnt+1,y
01142  d7cc  88                dey
01143  d7cd  10 f5             bpl avck4
01144  d7cf  e4 07             cpx t3            compare bytes free on track as per
                                                 BAM
01145  d7d1  f0 07             beq avck6
01146  d7d3  a9 71             lda #direrr       count doesn't match!
01147  d7d5  a0 00             ldy #0
01148  d7d7  4c 5c d9          jmp cmder2
01149  d7da  60          avck6 rts
01150  d7db
01151  d7db  ===> Tell how many sectors allowed for this track <===
01152  d7db
01153  d7db  a2 04       maxsec ldx #$04         number of zones counter
01154  d7dd  dd e6 d7    max1  cmp trknum-1,x    zone border value
01155  d7e0  ca                dex
01156  d7e1  b0 fa             bcs max1          track number in .A less than
                                                 boundary value
01157  d7e3  bd 99 10          lda numsec,x      number of sectors per track this
                                                 zone allows
01158  d7e6  60                rts
01159  d7e7
01160  d7e7
01161  d7e7  ===> Tables used by MAXSEC <===
01162  d7e7
01163  d7e7  24 1f 19   trknum .byte 36,31,25,18      zone border values
01164  d7ea  12
01164  d7eb
01165  d7eb                    .lib erproc
```

```
line   addr  object     source code

01167  d7eb  ===> Error Processing <===
01168  d7eb
01169  d7eb  ----------------------------------------
01170  d7eb
01171  d7eb  Controller errors:
01172  d7eb
01173  d7eb   0  (1)    no errors
01174  d7eb  20  (2)    can't find block header
01175  d7eb  21  (3)    no sync character
01176  d7eb  22  (4)    data block not present
01177  d7eb  23  (5)    checksum error in data
01178  d7eb  24 (16)    byte decoding error
01179  d7eb  25  (7)    write-verify error
01180  d7eb  26  (8)    write with write protect on
01181  d7eb  27  (9)    checksum error in header
01182  d7eb  28 (10)    data extends into next block
01183  d7eb  29 (11)    disk ID mismatch
01184  d7eb
01185  d7eb  Command errors:
01186  d7eb
01187  d7eb  30  general syntax
01188  d7eb  31  invalid command
01189  d7eb  32  long line
01190  d7eb  33  invalid filename
01191  d7eb  34  no file given
01192  d7eb
01193  d7eb  50  record not present
01194  d7eb  51  overflow in record
01195  d7eb  52  file too large
01196  d7eb
01197  d7eb  60  file open for write
01198  d7eb  61  file not open
01199  d7eb  62  file not found
01200  d7eb  63  file exists
01201  d7eb  64  file type mismatch
01202  d7eb  65  no block
01203  d7eb  66  illegal track or sector
01204  d7eb  67  illegal system track or sector
01205  d7eb
01206  d7eb  70  no channels available
01207  d7eb  71  directory error
01208  d7eb  72  disk full
01209  d7eb  73  CBM DOS V2
01210  d7eb
01211  d7eb   1  files scratched response
01212  d7eb
01213  d7eb  ----------------------------------------
01214  d7eb
01215  d7eb  ===> Error codes <===
01216  d7eb
01217  d7eb            badsyn = $30
01218  d7eb            badcmd = $31
01219  d7eb            longln = $32
```

```
line   addr  object     source code

01220  d7eb             badfn  = $33
01221  d7eb             nofile = $34
01222  d7eb             norec  = $50
01223  d7eb             recovf = $51
01224  d7eb             bigfil = $52
01225  d7eb             filopn = $60
01226  d7eb             filnop = $61
01227  d7eb             flntfd = $62
01228  d7eb             flexst = $63
01229  d7eb             mistyp = $64
01230  d7eb             noblk  = $65
01231  d7eb             badts  = $66
01232  d7eb             nochnl = $70
01233  d7eb             direrr = $71
01234  d7eb             dskful = $72
01235  d7eb             cbmv2  = $73
01236  d7eb
01237  d7eb
01238  d7eb  ===> Error Message Table <===
01239  d7eb
01240  d7eb  leading error numbers, text with 1st & last characters
01241  d7eb  ORed with $80,
01242  d7eb  tokens for key words are less than $10 (ANDed with $80)
01243  d7eb
01244  d7eb  00 a0 4f    errtab .byte $00, $a0, 'oK'
01245  d7ee  cb
01246  d7ef  20 21 22           .byte $20, $21, $22, $23, $24, $27, 'Read', $89
01247  d7f2  23 24 27
01248  d7f5  d2 45 41
01249  d7f8  44 89
01250  d7fa  52 83 20           .byte $52, $83, ' too largE'
01251  d7fd  54 4f 4f
01252  d800  20 4c 41
01253  d803  52 47 c5
01254  d806  50 8b 06           .byte $50, $8b, $06,' presenT'
01255  d809  20 50 52
01256  d80c  45 53 45
01257  d80f  4e d4
01258  d811  51 cf 56           .byte $51, 'Overflow in', $8b
01259  d814  45 52 46
01260  d817  4c 4f 57
01261  d81a  20 49 4e
01262  d81d  8b
01263  d81e  25 28 8a           .byte $25, $28, $8a, $89 write error
01264  d821  89
01265  d822  26 8a 20           .byte $26, $8a, ' protect oN'
01266  d825  50 52 4f
01267  d828  54 45 43
01268  d82b  54 20 4f
01269  d82e  ce
01270  d82f  29 88 20           .byte $29, $88, ' id', $85
01271  d832  49 44 85
01272  d835  30 31 32           .byte $30, $31, $32, $33, $34 syntax error
```

```
line   addr  object     source code

01273  d838  33 34
01274  d83a  d3 59 4e          .byte 'Syntax', $89
01275  d83d  54 41 58
01276  d840  89
01277  d841  60 8a 03          .byte $60, $8a, $03, $84 write file open
01278  d844  84
01279  d845  63 83 20          .byte $63, $83, ' existS'
01280  d848  45 58 49
01281  d84b  53 54 d3
01282  d84e  64 83 20          .byte $64, $83, ' type', $85
01283  d851  54 59 50
01284  d854  45 85
01285  d856  65 ce 4f          .byte $65, 'No blocK'
01286  d859  20 42 4c
01287  d85c  4f 43 cb
01288  d85f  66 67 c9          .byte $66, $67, 'Illegal track or sectoR'
01289  d862  4c 4c 45
01290  d865  47 41 4c
01291  d868  20 54 52
01292  d86b  41 43 4b
01293  d86e  20 4f 52
01294  d871  20 53 45
01295  d874  43 54 4f
01296  d877  d2
01297  d878  61 83 06          .byte $61, $83, $06, $84 file not open
01298  d87b  84
01299  d87c  62 83 06          .byte $62, $83, $06, $87 file not found
01300  d87f  87
01301  d880  01 83 53          .byte $01, $83, 's scratcheD'
01302  d883  20 53 43
01303  d886  52 41 54
01304  d889  43 48 45
01305  d88c  c4
01306  d88d  70 ce 4f          .byte $70, 'No channeL'
01307  d890  20 43 48
01308  d893  41 4e 4e
01309  d896  45 cc
01310  d898  71 c4 49          .byte $71, 'Dir', $89
01311  d89b  52 89
01312  d89d  72 88 20          .byte $72, $88, ' fulL'
01313  d8a0  46 55 4c
01314  d8a3  cc
01315  d8a4  73 c3 42          .byte $73, 'Cbm dos v', $b2 CBM DOS V2
01316  d8a7  4d 20 44
01317  d8aa  4f 53 20
01318  d8ad  56 b2
01319  d8af
01320  d8af  ===> Error token keywords used more than once <===
01321  d8af
01322  d8af              errtok = *-errtab
01323  d8af
01324  d8af  09 c5 52          .byte $09, 'ErroR'
01325  d8b2  52 4f d2
```

```
line   addr  object     source code

01326  d8b5  0a d7 52          .byte $0a, 'WritE'
01327  d8b8  49 54 c5
01328  d8bb  03 c6 49          .byte $03, 'FilE'
01329  d8be  4c c5
01330  d8c0  04 cf 50          .byte $04, 'OpeN'
01331  d8c3  45 ce
01332  d8c5  05 cd 49          .byte $05, 'MismatcH'
01333  d8c8  53 4d 41
01334  d8cb  54 43 c8
01335  d8ce  06 ce 4f          .byte $06, 'NoT'
01336  d8d1  d4
01337  d8d2  07 c6 4f          .byte $07, 'FounD'
01338  d8d5  55 4e c4
01339  d8d8  08 c4 49          .byte $08, 'DisK'
01340  d8db  53 cb
01341  d8dd  0b d2 45          .byte $0b, 'RecorD'
01342  d8e0  43 4f 52
01343  d8e3  c4
01344  d8e4
01345  d8e4              errend = *-errtab
01346  d8e4
01347  d8e4
01348  d8e4  ===> Recursive error message routine <===
01349  d8e4
01350  d8e4  dd eb d7    moverr cmp errtab,x     A = BCD error number
01351  d8e7  f0 06             beq mer5
01352  d8e9  e8                inx
01353  d8ea  e0 f9             cpx #$f9
01354  d8ec  90 f6             bcc moverr
01355  d8ee  60                rts
01356  d8ef
01357  d8ef  e8          mer5  inx               skip past error numbers
01358  d8f0  bd eb d7          lda errtab,x
01359  d8f3  10 fa             bpl mer5
01360  d8f5  29 7f             and #$7f
01361  d8f7  c9 10       mer6  cmp #$10
01362  d8f9  90 15             bcc mer70         token
01363  d8fb  91 47             sta (cb+2),y      so store character
01364  d8fd  c8                iny
01365  d8fe  e8          mer65 inx
01366  d8ff  bd eb d7          lda errtab,x
01367  d902  10 f3             bpl mer6
01368  d904  48                pha               last character
01369  d905  29 7f             and #$7f
01370  d907  c9 10             cmp #$10
01371  d909  90 06             bcc mer7          token
01372  d90b  91 47             sta (cb+2),y
01373  d90d  c8                iny
01374  d90e  68                pla
01375  d90f  60                rts
01376  d910
01377  d910  48          mer70 pha               token process
01378  d911  48          mer7  pha
```

```
line   addr  object      source code

01379  d912  a9 20             lda #$20
01380  d914  91 47             sta (cb+2),y     implied leading space
01381  d916  c8                iny
01382  d917  68                pla
01383  d918  86 07             stx t3
01384  d91a  a2 c4             ldx #$c4
01385  d91c  20 e4 d8          jsr moverr       recursive for tokens
01386  d91f  a6 07             ldx t3
01387  d921  68                pla
01388  d922  10 da             bpl mer65
01389  d924  60                rts
01390  d925
01391  d925
01392  d925  ===> Handle errors reported by controller <===
01393  d925
01394  d925  .A = error number
01395  d925  .X = job#
01396  d925
01397  d925  48          error  pha              save error code
01398  d926  86 a1             stx jobnum       job number in .X
01399  d928  8a                txa              as error number
01400  d929  0a                asl a            multiply
01401  d92a  0a                asl a            by
01402  d92b  0a                asl a            8
01403  d92c  aa                tax
01404  d92d  bd 23 10          lda hdrs+2,x     set track
01405  d930  85 13             sta track        current track#
01406  d932  bd 24 10          lda hdrs+3,x     and sector
01407  d935  85 14             sta sector       current sector#
01408  d937  68                pla              convert EC from disk controller
01409  d938  29 0f             and #$0f         to DOS code ready for output
01410  d93a  d0 02             bne err1         if 0, handle codes 16-20
01411  d93c  a9 06             lda #$06
01412  d93e  09 20       err1  ora #$20         6 ORed with $20 then 2 subtracted
01413  d940  aa                tax
01414  d941  ca                dex
01415  d942  ca                dex
01416  d943  8a                txa              marks code for output
01417  d944  48                pha              save DOS error code to stack
01418  d945  ad 7a 43          lda cmdnum       command #
01419  d948  c9 01             cmp #$01         open or validate?
01420  d94a  d0 0f             bne err2         no
01421  d94c  a9 ff             lda #$ff
01422  d94e  8d 7a 43          sta cmdnum       set command number to 255
01423  d951  68                pla              pull error code from stack and
01424  d952  20 dd d9          jsr errmsg       transfer to error buffer
01425  d955  20 ff ec          jsr initdr       initialize drive and eliminate bad
                                                 BAM
01426  d958  4c 5f d9          jmp cmder3       complete the error handling
01427  d95b
01428  d95b  68          err2  pla              pull DOS code off stack
01429  d95c  20 dd d9    cmder2 jsr errmsg      transfer code to error buffer
01430  d95f  20 be db    cmder3 jsr clrcb       Clear command buffer
```

```
line   addr  object     source code

01431  d962  a9 00            lda #0          clear error flag -- tell DOS not to
                                               write bad
01432  d964  8d 73 43         sta erword          BAM copy to disk
01433  d967  ad 82 02         lda pbd2        set error LED red
01434  d96a  09 20            ora #errled
01435  d96c  8d 82 02         sta pbd2
01436  d96f  20 d3 f0         jsr freich      Free internal channels L17 & L18
01437  d972  a9 00            lda #0          clear pointers
01438  d974  85 45            sta cb          in command buffer
01439  d976  a2 ff            ldx #$ff
01440  d978  9a               txs             reset stack
01441  d979  a5 17            lda orgsa       mask bits 0-3
01442  d97b  29 1f            and #$1f        mark as
01443  d97d  85 16            sta sa          current secondary address
01444  d97f  c9 0f            cmp #cmdsa      command channel = 15?
01445  d981  f0 2b            beq err10       yes
01446  d983  78               sei             no interrups!
01447  d984  a5 0e            lda lsnact      if listen active flag not 0, we are
                                               an
01448  d986  d0 11            bne lsnerr      active listener, else
01449  d988  a5 0f            lda tlkact      if no 0 here,
01450  d98a  f0 22            beq err10       we're an active talker, so do next
                                               routine
01451  d98c
01452  d98c
01453  d98c  ===> Talker error recovery <===
01454  d98c       if command channel, release DAV
01455  d98c       if data channel, force not ready and release channel
01456  d98c
01457  d98c  20 6e ed   tlkerr jsr fndrch       find an unused read channel
01458  d98f  ad 80 02           lda pad2        IEEE control port
01459  d992  09 10              ora #davo       DAV
01460  d994  8d 80 02           sta pad2        out
01461  d997  d0 0d              bne tlerr       bra
01462  d999
01463  d999
01464  d999  ===> Listener error recovery <===
01465  d999       if command channel, release RFD
01466  d999       if data channel, force not ready and release channel
01467  d999
01468  d999  20 89 ed   lsnerr jsr fndwch       find an unused read channel
01469  d99c  a9 04              lda #rfdo       RFD
01470  d99e  0d 80 02           ora pad2        IEEE control port
01471  d9a1  29 fe              and #$fe        ATN lo
01472  d9a3  8d 80 02           sta pad2        out
01473  d9a6  20 a6 ed   tlerr  jsr typfil
01474  d9a9  b0 03              bcs err10       if direct access
01475  d9ab  20 a4 ee           jsr frechn      close channel
01476  d9ae  4c a7 d4   err10  jmp idle          and twiddle your thumbs a while!
01477  d9b1
01478  d9b1
```

```
line   addr  object      source code

01479  d9b1  ===> Convert hex to bcd <===
01480  d9b1
01481  d9b1  aa          hexdec tax              transfer hex T & S
01482  d9b2  a9 00              lda #0
01483  d9b4  f8                 sed              set decimal mode
01484  d9b5  e0 00       hex0   cpx #0
01485  d9b7  f0 07              beq hex5         Convert bcd to ascii
01486  d9b9  18                 clc              add (X) times
01487  d9ba  69 01              adc #$01
01488  d9bc  ca                 dex
01489  d9bd  4c b5 d9           jmp hex0
01490  d9c0
01491  d9c0  d8          hex5   cld              clear decimal mode
01492  d9c1
01493  d9c1
01494  d9c1  ===> Convert bcd to ascii <===
01495  d9c1
01496  d9c1  aa          bcddec tax              packed figure
01497  d9c2  4a                 lsr a            mask tenths — divide BCD value by
                                                 16
01498  d9c3  4a                 lsr a
01499  d9c4  4a                 lsr a
01500  d9c5  4a                 lsr a
01501  d9c6  20 ca d9           jsr bcd2         convert MSB to ASCII and transfer to
                                                 error buffer
01502  d9c9  8a                 txa              same with ones
01503  d9ca  29 0f       bcd2   and #$0f         mask off the higher order nibble
01504  d9cc  09 30              ora #$30         to convert to ASCII
01505  d9ce  91 47              sta (cb+2),y     this now contains ASCII number
01506  d9d0  c8                 iny
01507  d9d1  60                 rts              .X still contains BCD number
01508  d9d2
01509  d9d2
01510  d9d2  ===> Transfer error message to error buffer <===
01511  d9d2
01512  d9d2  20 4b da    okerr  jsr erroff        error LED green
01513  d9d5  a9 00              lda #0           no error
01514  d9d7  a0 00       errts0 ldy #0           set
01515  d9d9  84 13              sty track        track &
01516  d9db  84 14              sty sector       sector to 0
01517  d9dd  a0 00       errmsg ldy #0
01518  d9df  a2 dc              ldx #<errbuf     pointer to error buffer
01519  d9e1  86 47              stx cb+2
01520  d9e3  a2 43              ldx #>errbuf
01521  d9e5  86 48              stx cb+3
01522  d9e7  20 c1 d9           jsr bcddec       convert error number and store at
                                                 start of error buffer
01523  d9ea  a9 2c              lda #','         store "," after error message
01524  d9ec  91 47              sta (cb+2),y     in error buffer
01525  d9ee  c8                 iny              points into error buffer
01526  d9ef  ad dc 43           lda errbuf       copy 1st digit from error buffer
01527  d9f2  85 bc              sta chndat+errchn      to output register (channel
                                                 data area)
```

```
line   addr  object     source code

01528  d9f4  8a               txa             error number in .X
01529  d9f5  a2 00            ldx #0
01530  d9f7  20 e4 d8         jsr moverr      move error message to buffer
01531  d9fa  a9 2c      ermsg2 lda #','        store ","
01532  d9fc  91 47            sta (cb+2),y    in error buffer after error message
01533  d9fe  c8               iny             points into error buffer
01534  d9ff  a5 13            lda track       convert track
01535  da01  20 b1 d9         jsr hexdec      to bcd
01536  da04  a9 2c            lda #','
01537  da06  91 47            sta (cb+2),y
01538  da08  c8               iny
01539  da09  a5 14            lda sector      convert sector
01540  da0b  20 b1 d9         jsr hexdec      to bcd
01541  da0e  88               dey             point to last character
01542  da0f  98               tya
01543  da10  18               clc
01544  da11  69 dc            adc #<errbuf    add start address lo of error buffer
01545  da13  85 c4            sta lstchr+errchn      store as end pointer
01546  da15  e6 47            inc cb+2        increment error buffer pointer
01547  da17  a9 88            lda #rdytlk     indicate ready-to-talk error channel
                                               status
01548  da19  85 9f            sta chnrdy+errchn      set read flag, reset EOI
01549  da1b  60               rts
01550  da1c
01551  da1c
01552  da1c   ===> Mark track and sector as free in BAM <===
01553  da1c
01554  da1c  20 89 d7   frets  jsr setbmp      point BMPNT at BAM
01555  da1f  20 b4 eb         jsr freuse      Calculate BAM index for FRETS and
                                               USEDTS
01556  da22  38               sec             flag for no action
01557  da23  d0 0f            bne frerts      free already
01558  da25  b1 02            lda (bmpnt),y   not free -- free it
01559  da27  1d ce eb         ora bmask,x     BAM mask bytes
01560  da2a  91 02            sta (bmpnt),y   bit map pointer
01561  da2c  a4 04            ldy temp        index to free sectors counter
01562  da2e  b1 02            lda (bmpnt),y   add one (C=1)
01563  da30  69 00            adc #0
01564  da32  91 02            sta (bmpnt),y   bit map pointer
01565  da34  60         frerts rts
01566  da35
01567  da35
01568  da35  ===> Turn LED on/off for current drive <===
01569  da35
01570  da35  a9 e7      setlds lda #$e7
01571  da37  2d 82 02         and pbd2
01572  da3a  48               pha
01573  da3b  a5 12            lda drvnum      current drive
01574  da3d  f0 05            beq leds0
01575  da3f  68               pla
01576  da40  09 08            ora #led1
01577  da42  d0 03            bne leds1
01578  da44  68         leds0 pla
```

```
line   addr  object     source code

01579  da45  09 10            ora #led0
01580  da47  8d 82 02  ledsl   sta pbd2
01581  da4a  60                rts
01582  da4b
01583  da4b  ad 82 02  erroff  lda pbd2          turn off error LED
01584  da4e  29 df             and #$ff-errled
01585  da50  8d 82 02          sta pbd2
01586  da53  60                rts
01587  da54
01588  da54
01589  da54  ===> Start directory loading function; get buffer first <===
01590  da54
01591  da54  a9 00       stdir  lda #0
01592  da56  85 16              sta sa             current secondary address
01593  da58  a9 01              lda #$01           allocate channel and 1 buffer
01594  da5a  20 63 ee           jsr getrch         open a new read channel
01595  da5d  a9 00              lda #0
01596  da5f  20 c1 f0           jsr setpnt         Set up pointer into active data
                                                  buffer
01597  da62  a6 15              ldx lindx          logical index, channel number
01598  da64  a9 00              lda #0
01599  da66  95 bd              sta lstchr,x       channel last character pointer
01600  da68  20 95 fa           jsr getact         Get active buffer number
01601  da6b  aa                 tax
01602  da6c  a5 12              lda drvnum         current drive
01603  da6e  9d 4e 43           sta lstjob,x       last job by buffer
01604  da71  a9 01              lda #$01           put SAL in buffer
01605  da73  20 b6 ec           jsr putbyt
01606  da76  a9 04              lda #$04           put SAH in buffer
01607  da78  20 b6 ec           jsr putbyt
01608  da7b  a9 01              lda #$01           insert phoney program line links
                                                  ($0101)
01609  da7d  20 b6 ec           jsr putbyt
01610  da80  20 b6 ec           jsr putbyt
01611  da83  ad 77 43           lda nbtemp         temporary number of blocks
01612  da86  20 b6 ec           jsr putbyt         put in DRVNUM
01613  da89  a9 00              lda #0
01614  da8b  20 b6 ec           jsr putbyt         store as hi byte of line mumber
01615  da8e  20 0c db           jsr movbuf         get disk name name into buffer
01616  da91  20 95 fa           jsr getact         and active buffer number
01617  da94  0a                 asl a              multiply by 2
01618  da95  aa                 tax                and store in .X, then decrement
01619  da96  d6 29              dec buftab,x       buffer 0 pointer lo
01620  da98  d6 29              dec buftab,x       twice
01621  da9a  a9 00              lda #0             end-of-line null byte
01622  da9c  20 b6 ec           jsr putbyt
01623  da9f  a9 01       dir1   lda #$01           insert phoney links
01624  daa1  20 b6 ec           jsr putbyt
01625  daa4  20 b6 ec           jsr putbyt
01626  daa7  20 c9 e0           jsr getnam         Get number of buffers and file name
01627  daaa  90 2c              bcc dir3           test if last entry
01628  daac  ad 77 43           lda nbtemp         number of temp. blocks, use as lo
                                                  line number byte
```

```
line   addr  object      source code

01629  daaf  20 b6 ec           jsr putbyt
01630  dab2  ad 78 43           lda nbtemp+1      hi line number byte
01631  dab5  20 b6 ec           jsr putbyt
01632  dab8  20 0c db           jsr movbuf        move filename & type into buffer
01633  dabb  a9 00              lda #0            end of entry
01634  dabd  20 b6 ec           jsr putbyt        if on return the Z flag not set,
                                                  buffer is
01635  dac0  d0 dd              bne dir1          not yet full so do next file entry
01636  dac2  20 95 fa   dir10   jsr getact        Get active buffer number
01637  dac5  0a                 asl a
01638  dac6  aa                 tax
01639  dac7  a9 00              lda #0
01640  dac9  95 29              sta buftab,x      buffer 0 pointer lo
01641  dacb  a9 88              lda #rdytlk
01642  dacd  a4 15              ldy lindx         logical index, channel number
01643  dacf  8d 46 43           sta dirlst        directory listing flag
01644  dad2  99 98 00           sta chnrdy,y      directory list buffer full
01645  dad5  a5 18              lda data          temporary data byte
01646  dad7  60                 rts
01647  dad8
01648  dad8  ad 77 43   dir3    lda nbtemp        this is end of load: line number lo
01649  dadb  20 b6 ec           jsr putbyt
01650  dade  ad 78 43           lda nbtemp+1      line number hi
01651  dae1  20 b6 ec           jsr putbyt
01652  dae4  20 0c db           jsr movbuf        mobe filename, type
01653  dae7  20 95 fa           jsr getact        Get active buffer number
01654  daea  0a                 asl a
01655  daeb  aa                 tax
01656  daec  d6 29              dec buftab,x      buffer 0 pointer lo
01657  daee  d6 29              dec buftab,x      buffer 0 pointer lo
01658  daf0  a9 00              lda #0            end of listing (000)
01659  daf2  20 b6 ec           jsr putbyt
01660  daf5  20 b6 ec           jsr putbyt
01661  daf8  20 b6 ec           jsr putbyt
01662  dafb  20 95 fa           jsr getact        Get active buffer number
01663  dafe  0a                 asl a
01664  daff  a8                 tay
01665  db00  b9 29 00           lda buftab,y      buffer 0 pointer lo
01666  db03  a6 15              ldx lindx         logical index, channel number
01667  db05  95 bd              sta lstchr,x      channel last character pointer
01668  db07  d6 bd              dec lstchr,x      pointer to last character in buffer
01669  db09  4c c2 da           jmp dir10         set channel status, flags and exit
01670  db0c
01671  db0c
01672  db0c  ===> Transfer filename to listing buffer <===
01673  db0c
01674  db0c  a0 00      movbuf  ldy #0
01675  db0e  b9 b4 41   movb1   lda nambuf,y      directory buffer
01676  db11  20 b6 ec           jsr putbyt        Byte to active buffer of LINDEX
                                                  channel
01677  db14  c8                 iny
01678  db15  c0 1b              cpy #nbsiz        count off 28 bytes
01679  db17  d0 f5              bne movb1
```

```
line   addr  object      source code

01680  db19  60                rts
01681  db1a
01682  db1a  ===> Get character for directory loading <===
01683  db1a
01684  db1a  20 b8 ed   getdir jsr getbyt       Read one byte from the active buffer
01685  db1d  f0 01             beq getd3        end-of-file if Z flag is set
01686  db1f  60                rts
01687  db20
01688  db20  85 18       getd3 sta data         temporary data byte
01689  db22  a4 15             ldy lindx        logical index, channel number
01690  db24  b9 bd 00          lda lstchr,y     channel last character pointer, lo
                                                byte
01691  db27  f0 08             beq gdl          if 0 we have exhausted the current
                                                buffer
01692  db29  a9 80             lda #eoiout      musthave reached EOF
01693  db2b  99 98 00          sta chnrdy,y     write, read, eoi flags, channel
                                                status
01694  db2e  a5 18             lda data         temporary data byte
01695  db30  60                rts
01696  db31
01697  db31  4c 9f da    gdl   jmp dirl         create pseudo program listing
01698  db34
01699  db34
01700  db34  ===> Get number of blocks free in DRVNUM <===
01701  db34
01702  db34  a6 12       numfre ldx drvnum      current drive
01703  db36  bd e8 d2          lda ipbm,x       BAM address hi
01704  db39  85 05             sta t1
01705  db3b  a0 04             ldy #$04
01706  db3d  a9 00             lda #0
01707  db3f  85 04             sta temp         0 lo pointer
01708  db41  aa                tax              0 hi pointer
01709  db42  18          numf1 clc
01710  db43  71 04             adc (temp),y     temporary work area
01711  db45  90 01             bcc numf2
01712  db47  e8                inx
01713  db48  c8           numf2 iny
01714  db49  c8                iny
01715  db4a  c8                iny
01716  db4b  c8                iny
01717  db4c  c0 48             cpy #$48         don't count the directory
01718  db4e  f0 f8             beq numf2
01719  db50  c0 90             cpy #$90
01720  db52  d0 ee             bne numf1
01721  db54  8d 77 43          sta nbtemp       temporary number of blocks
01722  db57  8e 78 43          stx nbtemp+1
01723  db5a  60                rts
01723  db5b
01724  db5b                    .lib parsex
```

```
line   addr  object      source code

01726  db5b  ===> Parse & execute string in command buffer <===
01727  db5b
01728  db5b  20 d2 d9   parsxq jsr okerr        Transfer OK message to error buffer
01729  db5e  a5 17             lda orgsa        original secondary address
01730  db60  10 09             bpl ps05
01731  db62  29 0f             and #$0f
01732  db64  c9 0f             cmp #$0f         is it the command channel?
01733  db66  f0 03             beq ps05         (if it's the command channel)
01734  db68  4c 79 f2          jmp open         Open IEEE channel
01735  db6b
01736  db6b  20 b6 dc   ps05    jsr cmdset       Initialize cmd tables & pointers
01737  db6e  b1 45              lda (cb),y       cmd buffer pointer lo
01738  db70  8d 7b 43           sta char         character under parser
01739  db73  a2 0a              ldx #ncmds-1
01740  db75  bd a1 d2   ps10    lda cmdtbl,x     search command table
01741  db78  cd 7b 43           cmp char         for character under parser
01742  db7b  f0 08              beq ps20         o.k. — found
01743  db7d  ca                 dex              try
01744  db7e  10 f5              bpl ps10         again
01745  db80  a9 31              lda #badcmd      no such command
01746  db82  4c c9 db           jmp cmderr       Command level error handling
01747  db85
01748  db85  8e 7a 43   ps20    stx cmdnum       command number
01749  db88  e0 08              cpx #pcmd        is it a command that needs parsing?
01750  db8a  90 03              bcc ps30         if parsing is required,
01751  db8c  20 ef db           jsr tagcmd       Tag command string, set up cmd &
                                                  filestream pointers
01752  db8f  ae 7a 43   ps30    ldx cmdnum       move address of appropriate ROM
                                                  routine
01753  db92  bd ac d2           lda cjumpl,x     from tables to
01754  db95  85 04              sta temp         temporary storage
01755  db97  bd b7 d2           lda cjumph,x     (lo/hi)
01756  db9a  85 05              sta t1           then jump to routine
01757  db9c  6c 04 00           jmp (temp)       via vector now in (temp)
01758  db9f
01759  db9f
01760  db9f  ===> Terminate command successfully <===
01761  db9f
01762  db9f  ad 73 43   endcmd lda erword       error word for recovery. If not 0,
                                                  error, so
01763  dba2  d0 25              bne cmderr       report it!
01764  dba4  a0 00              ldy #0           If command completed with no errors,
01765  dba6  98                 tya              scratch entry: zero the
01766  dba7  84 13              sty track        current track number
01767  dba9  84 14      screnc sty sector       current sector number and
01768  dbab  84 45              sty cb           cmd buffer pointer lo to zero
01769  dbad  20 dd d9           jsr errmsg       then clear error status
01770  dbb0  20 4b da           jsr erroff
01771  dbb3  a5 12              lda drvnum       move current drive number to
01772  dbb5  8d 94 43           sta lstdrv       last drive without error
01773  dbb8  20 be db           jsr clrcb        Clear command buffer
01774  dbbb  4c d3 f0           jmp freich       Free both internal channels
01775  dbbe
```

```
line   addr  object      source code

01776  dbbe  ===> Clear command buffer <===
01777  dbbe
01778  dbbe  a0 39       clrcb  ldy #cmdlen-1
01779  dbc0  a9 00              lda #0            erase old commands in
01780  dbc2  99 00 43    clrb2  sta cmdbuf,y      command buffer
01781  dbc5  88                  dey
01782  dbc6  10 fa               bpl clrb2
01783  dbc8  60                  rts
01784  dbc9
01785  dbc9
01786  dbc9  ===> Command level error handling <===
01787  dbc9
01788  dbc9  a0 00       cmderr ldy #0
01789  dbcb  84 13              sty track         current track number
01790  dbcd  84 14              sty sector        current sector number
01791  dbcf  4c 5c d9           jmp cmder2
01792  dbd2
01793  dbd2
01794  dbd2  ===> Simple parser <===
01795  dbd2
01796  dbd2  a2 00       simprs ldx #0
01797  dbd4  8e 80 43           stx filtbl
01798  dbd7  a9 3a              lda #':'
01799  dbd9  20 69 dc           jsr parse         scan for colon. If found, Z=1
01800  dbdc  f0 05              beq sp10
01801  dbde  88                 dey               .Y points to its position in the
                                                  command
01802  dbdf  88                 dey
01803  dbe0  8c 80 43           sty filtbl
01804  dbe3  4c 64 dd    sp10   jmp setany        Set drive number from any
                                                  configuration
01805  dbe6
01806  dbe6
01807  dbe6  ===> Find colon in command string <===
01808  dbe6
01809  dbe6  a0 00       prscln ldy #0
01810  dbe8  a2 00              ldx #0
01811  dbea  a9 3a              lda #':'
01812  dbec  4c 69 dc           jmp parse         Store desired character in CHAR
01813  dbef
01814  dbef  ===> Tag command string, set up cmd & filestream pointers <===
01815  dbef
01816  dbef  ------------------------------------------------------------------
01817  dbef  Command structure (bit mapped)
01818  dbef  ----------------
01819  dbef  The commands RENAME, SCRATCH, NEW and LOAD are analyzed by this
01820  dbef  routine to determine the command structure. As the command is
01821  dbef  parsed, bits in IMAGE are set or cleared to indicate the presence
01822  dbef  or absence of various parts of the command. After analysis the
01823  dbef  structure image is checked against the correct structure for that
01824  dbef  command in STRUCT
01825  dbef
01826  dbef
```

```
line   addr  object      source code

01827  dbef  --- ----- ------------------------
01828  dbef   7  P1    Wild cards present (Y=1)
01829  dbef   6  G1    More than one file implied (Y=1)
01830  dbef   5  D1    Drive# specified (not default)
01831  dbef   4  N1    Filename1 given
01832  dbef   3  P2    Wild cards present (Y=1)
01833  dbef   2  G2    More than one file implied (Y=1)
01834  dbef   1  D2    Drive# specified (not default)
01835  dbef   0  N2     Filename2 given
01836  dbef
01837  dbef             Bits 7-4 refer to file# 1
01838  dbef             Bits 3-0 refer to file# 2
01839  dbef  ------------------------------------------------------------
01840  dbef
01841  dbef
01842  dbef  20 e6 db   tagcmd jsr prscln       Find colon in command string
01843  dbf2  d0 05             bne tc30
01844  dbf4  a9 34      tc25   lda #nofile      this is a bad command--no files!
01845  dbf6  4c c9 db          jmp cmderr       Command level error handling
01846  dbf9
01847  dbf9  88         tc30   dey
01848  dbfa  88                dey
01849  dbfb  8c 80 43          sty filtbl       if ":" found, filestream 1 starts at
                                                previous character
01850  dbfe  8a                txa              if .X > 0, bad syntax
01851  dbff  d0 f3             bne tc25
01852  dc01  a9 3d      tc35   lda #'='
01853  dc03  20 69 dc          jsr parse        Store desired character in CHAR
01854  dc06  8a                txa              .X=0 indicates no * or ? found, else
                                                set bit 6
01855  dc07  f0 02             beq tc40         of IMAGE to indicate that the
                                                command applies
01856  dc09  a9 40             lda #%01000000   to more than one file. In all cases
                                                set bit 5
01857  dc0b  09 21      tc40   ora #%00100001   to indicate that a second filename
                                                is given (is
01858  dc0d  8d 91 43          sta image        fixed later)
01859  dc10  e8                inx              use to set
01860  dc11  8e 7d 43          stx flcnt        size of filename1 and 2. Filename2
                                                will
01861  dc14  8e 7e 43          stx f2cnt        default to same length as fn1
01862  dc17  ad 90 43          lda patflg       has PARSE found wild cards? If so,
01863  dc1a  f0 0d             beq tc50
01864  dc1c  a9 80             lda #%10000000   set bit 7
01865  dc1e  0d 91 43          ora image        file stream image
01866  dc21  8d 91 43          sta image        file stream image
01867  dc24  a9 00             lda #0           prepare for parsing of
01868  dc26  8d 90 43          sta patflg       rest of command
01869  dc29  98         tc50   tya              any more commands to parse? if Y=0,
01870  dc2a  f0 29             beq tc75         no more, so check structure, else
01871  dc2c  9d 80 43          sta filtbl,x     value of Y in filtbl
01872  dc2f  ad 7d 43          lda flcnt        set pointer to start of filename2
                                                from
```

```
line   addr  object      source code

01873  dc32  8d 7f 43          sta f2ptr       current value of filename1
01874  dc35  a9 8d             lda #$8d        find shifted CR
01875  dc37  20 69 dc          jsr parse       Store desired character in CHAR
01876  dc3a  e8                inx
01877  dc3b  8e 7e 43          stx f2cnt       file stream 2 count
01878  dc3e  ca                dex             restore for test
01879  dc3f  ad 90 43          lda patflg      last pattern
01880  dc42  f0 02             beq tc60        if any wildcards found,
01881  dc44  a9 08             lda #%1000      set bit 3
01882  dc46  ec 7d 43   tc60    cpx flcnt       check if second filename. If only
                                                one
01883  dc49  f0 02             beq tc70        character long, branch, else
01884  dc4b  09 04             ora #%0100      set bit 2 (command implies more than
                                                one filename)
01885  dc4d  09 03      tc70    ora #%0011      bit 1 (more drives) and bit 0
                                                (more filenames). Then
01886  dc4f  4d 91 43          eor image       clear bit 0 and
01887  dc52  8d 91 43          sta image       restore
01888  dc55  ad 91 43   tc75    lda image       check image against
01889  dc58  ae 7a 43          ldx cmdnum      entry
01890  dc5b  3d bb d2          and struct,x    in command template
01891  dc5e  d0 01             bne tc80
01892  dc60  60                rts             if o.k.
01893  dc61
01894  dc61  8d 73 43   tc80    sta erword      error word for recovery
01895  dc64  a9 30             lda #badsyn
01896  dc66  4c c9 db          jmp cmderr      Command level error handling
01897  dc69
01898  dc69  ------------------------------------------------------------------
01899  dc69  Parse string:
01900  dc69  On entry, .A contains character to be found in the string.
01901  dc69            .Y points to that character where scan must begin
01902  dc69            .X points into the filetable.
01903  dc69  The routine also scans the string for special characters (* ? ,).
01904  dc69  When a wild card is found, the pattern flag is incremented. When
01905  dc69  a comma is found, or the end of the command is reached, the
01906  dc69  routine ends.If no wild cards are found, the pattern flag is set
01907  dc69  to $80, otherwise it remains unchanged.
01908  dc69  On exit, .Y=0 and the Z-flag =0 if the desired character has not
01909  dc69  been found. If it has been found, .Y= the position of the character
01910  dc69  and the Z flag is set.
01911  dc69  ------------------------------------------------------------------
01912  dc69
01913  dc69  ===> Store desired character in CHAR <===
01914  dc69
01915  dc69  8d 7b 43   parse  sta char        character under parser
01916  dc6c  cc 79 43   pr10   cpy cmdsiz      command string size
01917  dc6f  b0 2f             bcs pr30
01918  dc71  b1 45             lda (cb),y      cmd buffer pointer lo
01919  dc73  c8                iny
01920  dc74  cd 7b 43          cmp char        character under parser
01921  dc77  f0 29             beq pr35
01922  dc79  c9 2a             cmp #'*'        check for wild cards
```

```
line   addr  object      source code

01923  dc7b  f0 04             beq pr20        or
01924  dc7d  c9 3f             cmp #'?'
01925  dc7f  d0 03             bne pr25
01926  dc81  ee 90 43   pr20    inc patflg      pattern present flag, count # of
                                                wild cards
01927  dc84  c9 2c      pr25    cmp #','        do we have a comma?
01928  dc86  d0 e4             bne pr10        if not, get next command string
                                                character
01929  dc88  98                tya
01930  dc89  9d 81 43          sta filtbl+1,x  find out where filename ends
01931  dc8c  ad 90 43          lda patflg      save pattern for each file
01932  dc8f  29 7f             and #$7f        0 means no wild cards
01933  dc91  f0 08             beq pr28
01934  dc93  a9 80             lda #$80        wild cards present, so indicate this
01935  dc95  9d 86 43          sta filtrk,x    first link/track
01936  dc98  8d 90 43          sta patflg      clear the count
01937  dc9b  e8         pr28   inx             number of files
01938  dc9c  e0 04             cpx #mxfils-1   4 files allowed in string. If less,
01939  dc9e  90 cc             bcc pr10        continue scanning
01940  dca0  a0 00      pr30   ldy #0          desired character not found (Z=1)
01941  dca2  ad 79 43   pr35   lda cmdsiz      copy command string size
01942  dca5  9d 81 43          sta filtbl+1,x
01943  dca8  ad 90 43          lda patflg      pattern present flag
01944  dcab  29 7f             and #$7f        no wild cards if 0
01945  dcad  f0 05             beq pr40
01946  dcaf  a9 80             lda #$80        wild cards present. Indicate this
01947  dcb1  9d 86 43          sta filtrk,x    first link/track
01948  dcb4  98         pr40   tya             Z is set
01949  dcb5  60                rts
01950  dcb6
01951  dcb6
01952  dcb6  ===> Initialize cmd tables & pointers <===
01953  dcb6       Find length of command string and zero variables and pointers
01954  dcb6
01955  dcb6  a4 45      cmdset ldy cb          cmd buffer pointer lo (command size
                                                sent from computer)
01956  dcb8  f0 14             beq cs08
01957  dcba  88                dey
01958  dcbb  f0 10             beq cs07
01959  dcbd  b9 00 43          lda cmdbuf,y    character from command buffer
01960  dcc0  c9 0d             cmp #cr         carriage return?
01961  dcc2  f0 0a             beq cs08
01962  dcc4  88                dey
01963  dcc5  b9 00 43          lda cmdbuf,y    next character
01964  dcc8  c9 0d             cmp #cr
01965  dcca  f0 02             beq cs08
01966  dccc  c8                iny             if not a carriage return, increment
                                                pointer
01967  dccd  c8         cs07   iny             increment command buffer pointer,
                                                then store
01968  dcce  8c 79 43   cs08   sty cmdsiz      length. Compare with
01969  dcd1  c0 3b             cpy #cmdlen+1   maximum allowable
01970  dcd3  a0 ff             ldy #$ff
```

```
line   addr  object      source code

01971  dcd5  90 08             bcc cmdrst      Zero all important variables and
                                                pointers
01972  dcd7  8c 7a 43          sty cmdnum      command is oversize
01973  dcda  a9 32             lda #longln
01974  dcdc  4c c9 db          jmp cmderr      Command level error handling
01975  dcdf
01976  dcdf
01977  dcdf  ===> Zero all important variables and pointers <===
01978  dcdf
01979  dcdf  a0 00       cmdrst ldy #0
01980  dce1  98                 tya
01981  dce2  85 45              sta cb          cmd buffer pointer lo
01982  dce4  8d 4b 43           sta rec         record size
01983  dce7  85 c5              sta type        current file type
01984  dce9  8d 9c 43           sta typflg      match by type of file
01985  dcec  85 81              sta flptr       file stream 1 pointer
01986  dcee  8d 7f 43           sta f2ptr       file stream 2 pointer
01987  dcf1  8d 7d 43           sta flcnt       file stream 1 count
01988  dcf4  8d 7e 43           sta f2cnt       file stream 2 count
01989  dcf7  8d 90 43           sta patflg      pattern present flag
01990  dcfa  8d 73 43           sta erword      error word for recovery
01991  dcfd  a2 05              ldx #mxfils
01992  dcff  9d 7f 43    cs10   sta filtbl-1,x
01993  dd02  95 85              sta filent-1,x
01994  dd04  95 8a              sta fildat-1,x
01995  dd06  9d 85 43           sta filtrk-1,x
01996  dd09  9d 8a 43           sta filsec-1,x
01997  dd0c  ca                 dex
01998  dd0d  d0 f0              bne cs10
01999  dd0f  60                 rts
02000  dd10
02001  dd10
02002  dd10  ===> Set first drive & table pointers <===
02003  dd10
02004  dd10  ad 7e 43    onedrv lda f2cnt       change pointer to end of first
                                                filename
02005  dd13  8d 7d 43           sta flcnt       to point to the end of second
                                                filename,
02006  dd16  a9 01              lda #1          then clear these variables:
02007  dd18  8d 7e 43           sta f2cnt
02008  dd1b  8d 7f 43           sta f2ptr
02009  dd1e
02010  dd1e
02011  dd1e  ===> Set up all drives from F2CNT <===
02012  dd1e
02013  dd1e  ac 94 43    alldrs ldy lstdrv      set up drive numbers
02014  dd21  a2 00              ldx #0          into file entry table
02015  dd23  86 81       ad10   stx flptr       on sector pointer byte
02016  dd25  bd 80 43           lda filtbl,x
02017  dd28  20 3a dd           jsr setdrv      Set drive number from text or
                                                default to 0
02018  dd2b  a6 81              ldx flptr       file stream 1 pointer
02019  dd2d  9d 80 43           sta filtbl,x    increment past ":"
```

```
line   addr  object     source code

02020  dd30  98               tya             bits represent drives
02021  dd31  95 8b            sta fildat,x    bit 7: default
02022  dd33  e8               inx             bit 0: drive number
02023  dd34  ec 7e 43         cpx f2cnt       see if any more files specified
02024  dd37  90 ea            bcc ad10
02025  dd39  60               rts             no more
02026  dd3a
02027  dd3a
02028  dd3a  ===> Set drive number from text or default to 0 <===
02029  dd3a
02030  dd3a  aa         setdrv tax             .A = index into command buffer
02031  dd3b  a9 3a            lda #':'        hunt for colon
02032  dd3d  dd 01 43         cmp cmdbuf+1,x  in command string
02033  dd40  f0 0c            beq setnme      Set drive number from command
                                              string. Syntax X#:FILENAME
02034  dd42  dd 00 43         cmp cmdbuf,x    command buffer
02035  dd45  d0 16            bne setfle      Set drive# from command string.
                                              Syntax X#,FILE or xx=FILE
02036  dd47  e8               inx
02037  dd48  98         sd20  tya             set up default drive
02038  dd49  29 01      sd22  and #1          make sure drive number converted to
                                              $0 or $1
02039  dd4b  a8         sd24  tay            restore drive number
02040  dd4c  8a               txa             and index & xxxxfile
02041  dd4d  60               rts
02042  dd4e
02043  dd4e
02044  dd4e  ===> Set drive number from command string. Syntax X#:FILENAME <===
02045  dd4e
02046  dd4e            sd40
02047  dd4e  bd 00 43   setnme lda cmdbuf,x   command buffer
02048  dd51  e8               inx               xxx:file
02049  dd52  e8               inx                        points to first filename
                                              character
02050  dd53  c9 30            cmp #$30          xx0:file
02051  dd55  f0 f2            beq sd22
02052  dd57  c9 31            cmp #$31          xx1:file
02053  dd59  f0 ee            beq sd22
02054  dd5b  d0 eb            bne sd20        cmd: file   or   xx:file
02055  dd5d            sd50   =*
02056  dd5d
02057  dd5d  ===> Set drive# from command string. Syntax X#,FILE or xx=FILE <===
02058  dd5d
02059  dd5d  98         setfle tya             for xxx,file or  xx=file
02060  dd5e  09 80            ora #%10000000
02061  dd60  29 81            and #%10000001  drive is default, mask off odd bits
02062  dd62  d0 e7            bne sd24        terminate testing
02063  dd64
02064  dd64  ===> Set drive number from any configuration <===
02065  dd64
02066  dd64  a9 00      setany lda #0
02067  dd66  8d 91 43         sta image       file stream image
02068  dd69  ac 80 43         ldy filtbl
```

```
line   addr  object      source code

02069  dd6c  b1 45       sa05   lda (cb),y      cmd buffer pointer lo
02070  dd6e  20 bb dd            jsr tst0v1      Test for 0 or 1
02071  dd71  10 12               bpl sa20        (if drive number given)
02072  dd73  c8                  iny             point to end of command less 1 so we
                                                 can
02073  dd74  cc 79 43            cpy cmdsiz      pick up things like VO
02074  dd77  b0 06               bcs sa10
02075  dd79  ac 79 43            ldy cmdsiz      command string size
02076  dd7c  88                  dey
02077  dd7d  d0 ed               bne sa05
02078  dd7f  ce 91 43   sa10     dec image       (becomes $ff) to flag default:
02079  dd82  ad 94 43            lda lstdrv      last drive without error
02080  dd85  29 01      sa20     and #1
02081  dd87  85 12               sta drvnum      current drive number
02082  dd89  4c 35 da            jmp setlds      Turn on LED for current drive
02083  dd8c
02084  dd8c
02085  dd8c  ===> Toggle drive number <===
02086  dd8c
02087  dd8c  a5 12       togdrv  lda drvnum      current drive number
02088  dd8e  49 01               eor #1
02089  dd90  29 01               and #1
02090  dd92  85 12               sta drvnum      current drive number
02091  dd94  60                  rts
02092  dd95
02093  dd95
02094  dd95  ===> Set pointers to one file stream and check type <===
02095  dd95
02096  dd95  a0 00       fslset  ldy #0
02097  dd97  ad 7d 43            lda flcnt       pointer to end of filename1
02098  dd9a  cd 7e 43            cmp f2cnt       equals same for filename2?
02099  dd9d  f0 16               beq fsl5        if equal, there is no second file,
                                                 else
02100  dd9f  ce 7e 43            dec f2cnt
02101  dda2  ac 7e 43            ldy f2cnt       file stream 2 count
02102  dda5  b9 80 43            lda filtbl,y    pointer to filetype
02103  dda8  a8                  tay
02104  dda9  b1 45               lda (cb),y      cmd buffer pointer lo
02105  ddab  a0 04               ldy #ntypes-1
02106  ddad  d9 d4 d2   fsl0     cmp typlst,y    DEL, SEQ, PRG, USR, REL
02107  ddb0  f0 03               beq fsl5        if no match, assume DEL
02108  ddb2  88                  dey
02109  ddb3  d0 f8               bne fsl0
02110  ddb5  98         fsl5     tya             transfer file type to .A
02111  ddb6  0a                  asl a           and store
02112  ddb7  8d 9c 43            sta typflg
02113  ddba  60                  rts
```

```
line   addr  object     source code

02115  ddbb  ===> Test for 0 or 1 <===
02116  ddbb
02117  ddbb  c9 30      tst0v1 cmp #'0'
02118  ddbd  f0 06             beq t0v1
02119  ddbf  c9 31             cmp #'1'
02120  ddc1  f0 02             beq t0v1
02121  ddc3  09 80             ora #%10000000   set bit 7 if no match found
02122  ddc5  29 81      t0v1   and #%10000001   convert to hex and preserve bit 7
02123  ddc7  60                rts
02123  ddc8
02124  ddc8                    .lib autoit

02126  ddc8  ===> RSR test subroutines <===
02127  ddc8
02128  ddc8  Checks if drive number is initialized. This routine works if
02129  ddc8  CATALOG calls it before any header is transferred. It will end
02130  ddc8  in error if any error but disk ID occurs
02131  ddc8
02132  ddc8  a2 ff      autoit ldx #$ff          flag for error RTN
02133  ddca  8e 9e 43          stx jobrtn       job return flag
02134  ddcd  20 e4 ec          jsr initsu       .A = DRVNUM<>=error
02135  ddd0  c9 03             cmp #$03         check missing disk
02136  ddd2  f0 07             beq catid3
02137  ddd4  c9 02             cmp #$02         check for o.k.
02138  ddd6  90 16             bcc catid4
02139  ddd8  4c 25 d9   catid2 jmp error        must be error, so report
02140  dddb
02141  dddb  ac 92 43   catid3 ldy drvcnt       number of drive searches
02142  ddde  f0 f8             beq catid2
02143  dde0  a9 00             lda #0           only one good drive?
02144  dde2  8d 92 43          sta drvcnt       number of drive searches
02145  dde5  a5 12             lda drvnum       current drive number
02146  dde7  49 01             eor #1           flip to check the other
02147  dde9  85 12             sta drvnum       current drive number
02148  ddeb  4c c8 dd          jmp autoit       RSR test subroutines
02149  ddee
02150  ddee  8a         catid4 txa              preserve .X
02151  ddef  0a                asl a            multiply by 8
02152  ddf0  0a                asl a
02153  ddf1  0a                asl a
02154  ddf2  a8                tay
02155  ddf3  a5 12             lda drvnum       current drive number
02156  ddf5  0a                asl a
02157  ddf6  aa                tax
02158  ddf7  b9 21 10          lda hdrs,y       check disk ID
02159  ddfa  dd 40 43          cmp dskid,x      against old ID
02160  ddfd  d0 0e             bne catid1
02161  ddff  b9 22 10          lda hdrs+1,y     ID2
02162  de02  dd 41 43          cmp dskid+1,x
02163  de05  d0 06             bne catid1
02164  de07  a5 a1             lda jobnum       retrieve job
02165  de09  20 97 ec          jsr seth         so we can restore all
02166  de0c  60                rts              under same ID
02167  de0d
```

```
line   addr  object      source code

02168  de0d  4c ff ec   catidl jmp initdr       different, so initialize
02169  de10
02170  de10  ===> Determine optimal search for LOOKUP and FINFIL <===
02171  de10
02172  de10  a9 00       optsch lda #0
02173  de12  85 04              sta temp
02174  de14  8d 93 43          sta drvflg       init drive mask
02175  de17  48                 pha              $00
02176  de18  ae 7e 43          ldx f2cnt        file stream 2 count
02177  de1b  68          os10   pla
02178  de1c  05 04              ora temp
02179  de1e  48                 pha
02180  de1f  a9 01              lda #1
02181  de21  85 04              sta temp
02182  de23  ca                 dex
02183  de24  30 0f              bmi os30         ($ff if no files left)
02184  de26  b5 8b              lda fildat,x     drive number, pattern
02185  de28  10 04              bpl os15         if default drive
02186  de2a  06 04              asl temp
02187  de2c  06 04              asl temp
02188  de2e  4a          os15   lsr a            if drive number was 1, carry is set
02189  de2f  90 ea              bcc os10
02190  de31  06 04              asl temp         since it was 0
02191  de33  d0 e6              bne os10         branch
02192  de35  68          os30   pla
02193  de36  aa                 tax              index into
02194  de37  bd 6a de           lda schtbl-1,x   search table
02195  de3a  48                 pha
02196  de3b  29 03              and #$03
02197  de3d  8d 92 43           sta drvcnt       number of drive searches
02198  de40  68                 pla
02199  de41  0a                 asl a
02200  de42  10 23              bpl os40         (if bit 7 not set)
02201  de44  a5 8b              lda fildat       drive number, pattern
02202  de46  29 01       os35   and #1
02203  de48  85 12              sta drvnum       current drive number
02204  de4a  ad f3 10           lda autofg
02205  de4d  d0 15              bne ox0000
02206  de4f  20 c8 dd           jsr autoit       RSR test subroutines
02207  de52  ad 92 43           lda drvcnt       number of drive searches
02208  de55  f0 0d              beq ox0000
02209  de57  a5 12              lda drvnum       current drive number
02210  de59  48                 pha              save it
02211  de5a  49 01              eor #1           flip it
02212  de5c  85 12              sta drvnum       current drive number
02213  de5e  20 c8 dd           jsr autoit       RSR test subroutines
02214  de61  68                 pla              restore drive number
02215  de62  85 12              sta drvnum       current drive number
02216  de64  4c 35 da    ox0000 jmp setlds       Turn on LED for current drive
02217  de67
02218  de67
02219  de67  2a          os40   rol a
02220  de68  4c 46 de           jmp os35
```

```
line   addr  object     source code

02221  de6b
02222  de6b  ===> Search table <===
02223  de6b
02224  de6b  00 80 41   schtbl .byte $00, $80, $41
02225  de6e  01 01 01          .byte 1,1,1,1
02226  de71  01
02227  de72  81 81 81          .byte $81, $81, $81, $81
02228  de75  81
02229  de76  42 42 42          .byte $42, $42, $42, $42
02230  de79  42
02231  de7a
02232  de7a
02233  de7a  ===> Look up files in cmd string in dir. & fill tables <===
02234  de7a
02235  de7a  20 10 de   lookup jsr optsch       Determine optimal search for LOOKUP
                                                and FINFIL
02236  de7d  a9 00      lk05   lda #0           indicate not looking for DEL or
                                                unused entry
02237  de7f  8d 98 43          sta delind       index of first available entry
02238  de82  20 da df          jsr srchst       Initiate search of directory
02239  de85  d0 1a             bne lk25         (if valid name found)
02240  de87  ce 92 43   lk10   dec drvcnt       number of drive searches
02241  de8a  10 01             bpl lk15
02242  de8c  60                rts
02243  de8d
02244  de8d  a9 01      lk15   lda #1
02245  de8f  8d 93 43          sta drvflg       drive search flag
02246  de92  20 8c dd          jsr togdrv       Toggle drive number
02247  de95  20 35 da          jsr setlds       Turn on LED for current drive
02248  de98  f0 e3             beq lk05         branch always
02249  de9a  d0 e1             bne lk05
02250  de9c  20 43 e0   lk20   jsr search        Continue search of entries
02251  de9f  f0 10             beq lk30         (abandon if no valid name found)
02252  dea1  20 04 df   lk25   jsr compar        Compare files in cmd list with valid
                                                directory entries
02253  dea4  ad 95 43          lda found        found flag in directory searches
02254  dea7  f0 01             beq lk26         0 if not found
02255  dea9  60                rts
02256  deaa
02257  deaa  ad 45 43   lk26   lda entfnd       directory entry found flag
02258  dead  30 ed             bmi lk20         does it match? $ff=not
02259  deaf  10 f0             bpl lk25         (if found, try again)
02260  deb1  ad 95 43   lk30   lda found        found flag in directory searches
02261  deb4  f0 d1             beq lk10         (all files not yet found, continue)
02262  deb6  60                rts
02263  deb7
02264  deb7
02265  deb7  ===> Find next name in file stream and table entry <===
02266  deb7
02267  deb7  20 31 e0   ffre   jsr srre         set up and read in next entry block
02268  deba  f0 1a             beq ff10         (none found)
02269  debc  d0 28             bne ff25         found
02270  debe  a9 01      ff15   lda #1           switch to the other drive:
```

```
line   addr  object      source code

02271  dec0  8d 93 43          sta drvflg       drive search flag
02272  dec3  20 8c dd          jsr togdrv       Toggle drive number
02273  dec6  20 35 da          jsr setlds       Turn on LED for current drive
02274  dec9
02275  dec9  ===> Find starting entry in directory <===
02276  dec9
02277  dec9  a9 00       ffst  lda #0           not looking for DEL or unused
                                                entries!
02278  decb  8d 98 43          sta delind       index of first available entry
02279  dece  20 da df          jsr srchst       Initiate search of directory
02280  ded1  d0 13             bne ff25         if Z=0: valid name found
02281  ded3  8d 95 43          sta found
02282  ded6  ad 95 43    ff10  lda found        if not 0, all files found, so
02283  ded9  d0 28             bne ff40
02284  dedb  ce 92 43          dec drvcnt       nothing more on this drive, so
02285  dede  10 de             bpl ff15         try the other. If none left,
02286  dee0  60                rts              exit
02287  dee1
02288  dee1
02289  dee1  ===> Continue scan of directory <===
02290  dee1
02291  dee1  20 43 e0    fndfil jsr search      retrieve next valid filename
02292  dee4  f0 f0             beq ff10
02293  dee6  20 04 df    ff25  jsr compar       Compare files in cmd list with valid
                                                directory entries
02294  dee9  ae 45 43          ldx entfnd       directory entry found flag
02295  deec  10 07             bpl ff30
02296  deee  ad 95 43          lda found        found flag in directory searches
02297  def1  f0 ee             beq fndfil       continue scan of directory
02298  def3  d0 0e             bne ff40
02299  def5  ad 9c 43    ff30  lda typflg       match by type of file
02300  def8  f0 09             beq ff40
02301  defa  b5 8b             lda fildat,x     drive number, pattern
02302  defc  29 1e             and #$1e
02303  defe  cd 9c 43          cmp typflg       match by type of file
02304  df01  d0 de             bne fndfil       Continue scan of directory
02305  df03  60          ff40  rts
02306  df04
02307  df04
02308  df04  ===> Compare files in cmd list with valid directory entries <===
02309  df04
02310  df04  a2 ff       compar ldx #$ff
02311  df06  8e 45 43           stx entfnd       directory entry found flag
02312  df09  e8                 inx              zero the
02313  df0a  8e 90 43           stx patflg       pattern present flag
02314  df0d  20 b9 df           jsr cmpchk       Check table for unfound files
02315  df10  f0 06              beq cp10
02316  df12  60          cp02   rts
02317  df13
02318  df13  20 c4 df    cp05   jsr cc10         point to next file. If no more
                                                needed,
02319  df16  d0 fa              bne cp02         exit, else
02320  df18  a5 12       cp10   lda drvnum       current drive number
```

```
line   addr  object     source code

02321  df1a  55 8b           eor fildat,x     drive number, pattern
02322  df1c  4a              lsr a
02323  df1d  90 0b           bcc cp20         (if carry flag clear, drive number
                                               o.k.)
02324  df1f  29 40           and #$40         see if default to be used ($40 not
                                               $80 because
02325  df21  f0 f0           beq cp05         of LSR). If no can use default, set
                                               up next
02326  df23  a9 02           lda #$02         filename, else check the
02327  df25  cd 92 43        cmp drvcnt       number of drive searches. If equal,
02328  df28  f0 e9           beq cp05         don't use default, else we have a
                                               match
02329  df2a  bd 80 43  cp20  lda filtbl,x     on drive numbers. Now find a name.
02330  df2d  aa              tax
02331  df2e  20 a1 e0        jsr fndlmt       Find end of string in command buffer
02332  df31  a0 03           ldy #$03         to make it point past type, T & S
02333  df33  4c 49 df        jmp cp33
02334  df36
02335  df36  bd 00 43  cp30  lda cmdbuf,x     command buffer (filename)
02336  df39  d1 27           cmp (dirbuf),y   directory buffer pointer (direct.
                                               entry)
02337  df3b  f0 0a           beq cp32
02338  df3d  c9 3f           cmp #'?'         ? matches any character
02339  df3f  d0 d2           bne cp05
02340  df41  b1 27           lda (dirbuf),y   directory buffer pointer
02341  df43  c9 a0           cmp #$a0         reached shifted space (end of entry
                                               name)?
02342  df45  f0 cc           beq cp05
02343  df47  e8        cp32  inx              set up for next character
02344  df48  c8              iny
02345  df49  ec 7c 43  cp33  cpx limit        pointer limit in comparison
02346  df4c  b0 09           bcs cp34         (if at end of string)
02347  df4e  bd 00 43        lda cmdbuf,x     command buffer
02348  df51  c9 2a           cmp #'*'
02349  df53  f0 0c           beq cp40
02350  df55  d0 df           bne cp30         keep matching
02351  df57  c0 13     cp34  cpy #$13         end of name?
02352  df59  b0 06           bcs cp40
02353  df5b  b1 27           lda (dirbuf),y   directory buffer pointer
02354  df5d  c9 a0           cmp #$a0
02355  df5f  d0 b2           bne cp05         if not shifted space, try again
02356  df61  ae 7f 43  cp40  ldx f2ptr        names match, so store pointer into
                                               the
02357  df64  8e 45 43        stx entfnd       directory entry found flag and fill
                                               tables:
02358  df67  bd 86 43        lda filtrk,x     first link/track
02359  df6a  29 80           and #$80
02360  df6c  8d 90 43        sta patflg       pattern present flag
02361  df6f  9d 86 43        sta filtrk,x     first link/track
02362  df72  ad 9a 43        lda index        current index in buffer
02363  df75  29 e0           and #$e0
02364  df77  85 04           sta temp
02365  df79  a5 14           lda sector       current sector number
```

```
line   addr  object      source code

02366  df7b  05 04             ora temp
02367  df7d  95 86             sta filent,x     table of sector numbers in directory
02368  df7f  a0 00             ldy #0
02369  df81  b1 27             lda (dirbuf),y   .Y contains file type
02370  df83  c8                iny
02371  df84  48                pha
02372  df85  29 40             and #$40         found "<" indicating a "locked"
                                                 file?
02373  df87  85 04             sta temp
02374  df89  68                pla              file type
02375  df8a  0a                asl a
02376  df8b  29 1e             and #$1e
02377  df8d  b0 02             bcs cp42         replacement bit set?
02378  df8f  09 20             ora #$20
02379  df91  05 04       cp42  ora temp
02380  df93  85 04             sta temp
02381  df95  a9 80             lda #$80
02382  df97  35 8b             and fildat,x     drive number, pattern
02383  df99  05 12             ora drvnum       current drive number
02384  df9b  05 04             ora temp
02385  df9d  95 8b             sta fildat,x     drive number, pattern
02386  df9f  b1 27             lda (dirbuf),y   directory buffer pointer (.Y=1)
02387  dfa1  1d 86 43          ora filtrk,x     first link/track
02388  dfa4  9d 86 43          sta filtrk,x     first link/track
02389  dfa7  c8                iny
02390  dfa8  b1 27             lda (dirbuf),y   directory buffer pointer (.Y=2)
02391  dfaa  9d 8b 43          sta filsec,x     first link/sector
02392  dfad  ad 4b 43          lda rec          record size. If not 0,
02393  dfb0  d0 07             bne cmpchk       Check table for unfound files
02394  dfb2  a0 15             ldy #$15
02395  dfb4  b1 27             lda (dirbuf),y   move file entry's record size
02396  dfb6  8d 4b 43          sta rec          record size
02397  dfb9
02398  dfb9
02399  dfb9  ===> Check table for unfound files <===
02400  dfb9
02401  dfb9                cp50
02402  dfb9  a9 ff       cmpchk lda #$ff        set all-files-found flag
02403  dfbb  8d 95 43           sta found       found flag in directory searches
02404  dfbe  ad 7e 43           lda f2cnt       move to test
02405  dfc1  8d 7f 43           sta f2ptr
02406  dfc4  ce 7f 43     cc10  dec f2ptr       file count
02407  dfc7  10 01              bpl cc15        another file to find?
02408  dfc9  60                 rts             no
02409  dfca
02410  dfca  ae 7f 43     cc15  ldx f2ptr       file number to test
02411  dfcd  bd 86 43           lda filtrk,x    first link/track
02412  dfd0  30 02              bmi cc20        (if bit 7 still set, not found)
02413  dfd2  d0 f0              bne cc10        to test next file
02414  dfd4  a9 00       cc20   lda #0          all-files-found flag
02415  dfd6  8d 95 43           sta found       found flag in directory searches
02416  dfd9  60                 rts
02417  dfda
```

```
line   addr  object     source code

02418  dfda  ===> Initiate search of directory <===
02419  dfda                                     Returns with valid entry (DELIND=0)
                                                or with the first deleted
02420  dfda                                     entry (DELIND=1)
02421  dfda                                     SRCHST will initiate a search  --
                                                SEARCH will continue a search
02422  dfda  a0 00      srchst ldy #0           initiate deleted sector
02423  dfdc  8c 97 43          sty delsec       sector of first available entry
02424  dfdf  88                dey              $ff
02425  dfe0  8c 45 43          sty entfnd       directory entry found flag
02426  dfe3  a9 12             lda #18          start search at beginning
02427  dfe5  85 13             sta track        current track number
02428  dfe7  a9 01             lda #1           last-sector-in-file flag
02429  dfe9  85 14             sta sector       current sector number
02430  dfeb  8d 99 43          sta lstbuf       =0 if last block
02431  dfee  20 6c f0          jsr opnird       Open internal read channel (SA=17)
02432  dff1  ad 99 43  sr10    lda lstbuf       =0 if last block
02433  dff4  d0 01             bne sr15
02434  dff6  60                rts              (Z=1)
02435  dff7
02436  dff7  a9 07     sr15    lda #$07         8 entries (0 to 7) to examine
02437  dff9  8d 9b 43          sta filcnt       counter of file entries
02438  dffc  a9 00             lda #0           read track number
02439  dffe  20 ef f0          jsr drdbyt       Direct read of a byte (.A =
                                                position)
02440  e001  8d 99 43          sta lstbuf       =0 if last block
02441  e004  20 e1 f0  sr20    jsr getpnt       Get the active buffer pointer
02442  e007  ce 9b 43          dec filcnt       counter of file entries
02443  e00a  a0 00             ldy #0
02444  e00c  b1 27             lda (dirbuf),y   read file type
02445  e00e  d0 18             bne sr30
02446  e010  ad 97 43          lda delsec       deleted entry found?
02447  e013  d0 2e             bne search       deleted entry already found
02448  e015  20 3b f9          jsr curblk       get current sector
02449  e018  a5 14             lda sector       current sector number
02450  e01a  8d 97 43          sta delsec
02451  e01d  a5 27             lda dirbuf       get index to start of this entry
02452  e01f  ae 98 43          ldx delind       bit 1: we want a deleted entry
02453  e022  8d 98 43          sta delind       store pointer in .A
02454  e025  f0 1c             beq search       we need a valid entry, not deleted
                                                ones
02455  e027  60                rts              found what we wanted (Z=0) -- a
                                                deleted entry
02456  e028
02457  e028  a2 01     sr30    ldx #1           valid entry found
02458  e02a  ec 98 43          cpx delind       looking for deleted?
02459  e02d  d0 2c             bne sr50         no!
02460  e02f  f0 12             beq search       Continue search of entries
02461  e031
02462  e031
02463  e031
```

```
line   addr  object      source code

02464  e031  ===> Re-entry directory search <===
02465  e031
02466  e031  a9 12       srre   lda #18
02467  e033  85 13              sta track        current track number
02468  e035  ad 96 43           lda dirsec       directory sector last used
02469  e038  85 14              sta sector       current sector number
02470  e03a  20 6c f0           jsr opnird       Open internal read channel (SA=17)
02471  e03d  ad 9a 43           lda index        current index in buffer
02472  e040  20 c1 f0           jsr setpnt       Set up pointer into active data
                                                  buffer
02473  e043
02474  e043
02475  e043  ===> Continue search of entries <===
02476  e043
02477  e043  a9 ff       search lda #$ff
02478  e045  8d 45 43           sta entfnd       directory entry found flag
02479  e048  ad 9b 43           lda filcnt       adjust file count
02480  e04b  30 08              bmi sr40         (if none left)
02481  e04d  a9 20              lda #$20         at least one more. 32 chars per
                                                  entry!
02482  e04f  20 47 ee           jsr incptr       Increment the pointer of the active
                                                  buffer by .A
02483  e052  4c 04 e0           jmp sr20         process it
02484  e055
02485  e055  20 44 f0    sr40   jsr nxtbuf       get next buffer of entries and
02486  e058  4c f1 df           jmp sr10         start processing
02487  e05b
02488  e05b  a5 27       sr50   lda dirbuf       valid entry found: save how far we
                                                  got
02489  e05d  8d 9a 43           sta index        current index in buffer
02490  e060  20 3b f9           jsr curblk       Read track & sector from header
02491  e063  a5 14              lda sector       current sector number
02492  e065  8d 96 43           sta dirsec       directory sector
02493  e068  60                 rts
02493  e069
02494  e069                     .lib trnsfer
```

```
line   addr  object     source code

02496  e069  ===> Transfer filename from command string to buffer <===
02497  e069       On entry, .A = string size, .X = starting index, .Y = buffer
                                                    number
02498  e069
02499  e069  48         trname pha
02500  e06a  20 a1 e0           jsr fndlmt        Find end of string in command buffer
02501  e06d  20 83 e0           jsr trcmbf        Transfer cmd buffer to another
                                                    buffer
02502  e070  68                 pla
02503  e071  38                 sec
02504  e072  ed 3a 43           sbc strsiz        length of the string
02505  e075  aa                 tax               if 0, no padding needed
02506  e076  f0 0a              beq tn20
02507  e078  90 08              bcc tn20
02508  e07a  a9 a0              lda #$a0          string is short, so pad with shifted
                                                    spaces
02509  e07c  91 27       tn10   sta (dirbuf),y    directory buffer
02510  e07e  c8                 iny
02511  e07f  ca                 dex
02512  e080  d0 fa              bne tn10          as needed
02513  e082  60          tn20   rts
02514  e083
02515  e083
02516  e083  ===> Transfer command buffer to another buffer <===
02517  e083
02518  e083  98          trcmbf tya               multiply buffer number by 2
02519  e084  0a                 asl a
02520  e085  a8                 tay
02521  e086  b9 29 00           lda buftab,y      set the
02522  e089  85 27              sta dirbuf        directory buffer pointers
02523  e08b  b9 2a 00           lda buftab+1,y
02524  e08e  85 28              sta dirbuf+1
02525  e090  a0 00              ldy #$00
02526  e092  bd 00 43    tr10   lda cmdbuf,x      command buffer
02527  e095  91 27              sta (dirbuf),y    directory buffer pointer
02528  e097  c8                 iny               if .Y = 0, then abort
02529  e098  f0 06              beq tr20
02530  e09a  e8                 inx
02531  e09b  ec 7c 43           cpx limit         index to last character + 1 in
                                                    command buffer
02532  e09e  90 f2              bcc tr10
02533  e0a0  60          tr20   rts
02534  e0a1
02535  e0a1
02536  e0a1  ===> Find end of string in command buffer <===
02537  e0a1
02538  e0a1  a9 00       fndlmt lda #$00
02539  e0a3  8d 3a 43           sta strsiz        length of the string = 0
02540  e0a6  8a                 txa
02541  e0a7  48                 pha
02542  e0a8  bd 00 43    f105   lda cmdbuf,x      command buffer
02543  e0ab  c9 2c              cmp #','          end?
02544  e0ad  f0 14              beq f110
```

```
line   addr  object      source code

02545  e0af  c9 3d             cmp #'='        end?
02546  e0b1  f0 10             beq fl10
02547  e0b3  ee 3a 43          inc strsiz      length of the string
02548  e0b6  e8                inx
02549  e0b7  a9 0f             lda #15         maximum size
02550  e0b9  cd 3a 43          cmp strsiz      length of the string
02551  e0bc  90 05             bcc fl10
02552  e0be  ec 79 43          cpx cmdsiz      command string size
02553  e0c1  90 e5             bcc fl05
02554  e0c3  8e 7c 43   fl10   stx limit       last character + 1
02555  e0c6  68                pla             transfer original .X value
02556  e0c7  aa                tax             from stack
02557  e0c8  60                rts
02558  e0c9
02559  e0c9
02560  e0c9  ===> Get file name from directory <===
02561  e0c9
02562  e0c9  a5 16      getnam lda sa          save SA and current channel number
                                               to stack
02563  e0cb  48                pha
02564  e0cc  a5 15             lda lindx
02565  e0ce  48                pha
02566  e0cf  20 d9 e0          jsr gnsub      Get directory entry via int. read
                                               channel 17
02567  e0d2  68                pla            reset LINDX and SA
02568  e0d3  85 15             sta lindx
02569  e0d5  68                pla
02570  e0d6  85 16             sta sa
02571  e0d8  60                rts
02572  e0d9
02573  e0d9
02574  e0d9  ===> Get file entry subroutine <===
02575  e0d9
02576  e0d9  a9 11      gnsub  lda #$11        channel number 17
02577  e0db  85 16             sta sa         current secondary address
02578  e0dd  20 6e ed          jsr fndrch     Find the assigned read channel
02579  e0e0  20 e1 f0          jsr getpnt     Get the active buffer pointer
02580  e0e3  ad 45 43          lda entfnd     directory entry found flag
02581  e0e6  10 0c             bpl gn05       more files
02582  e0e8  ad 93 43          lda drvflg     do other drive?
02583  e0eb  d0 0c             bne gn050
02584  e0ed  20 fa e1          jsr msgfre     Set up message "blocks free"
02585  e0f0  18                clc            C=0: end
02586  e0f1  4c a8 e1          jmp gn45       terminate
02587  e0f4
02588  e0f4  ad 93 43   gn05   lda drvflg     drive search flag = 0
02589  e0f7  f0 1f             beq gn10       send filename
02590  e0f9  ce 93 43   gn050  dec drvflg     if -1, go do new directory
02591  e0fc  d0 0d             bne gn051
02592  e0fe  ce 93 43          dec drvflg     -1
02593  e101  20 8c dd          jsr togdrv     Toggle drive number
02594  e104  20 fa e1          jsr msgfre     Set up message "blocks free"
02595  e107  38                sec
```

```
line   addr  object      source code

02596  e108  4c 8c dd          jmp togdrv        Toggle drive number
02597  e10b
02598  e10b  a9 00       gn051  lda #$00         zero hi byte of
02599  e10d  8d 78 43           sta nbtemp+1      blocks counter and
02600  e110  8d 93 43           sta drvflg        drive flag
02601  e113  20 b4 e1           jsr newdir        New directory in listing
02602  e116  38                 sec
02603  e117  60                 rts
02604  e118
02605  e118  a2 18       gn10   ldx #dirlen      set number of blocks & adjust
                                                  spacing
02606  e11a  a0 1d              ldy #29           (e.g.  123 "program filename"  prg)
02607  e11c  b1 27              lda (dirbuf),y    hi byte of number of blocks
02608  e11e  8d 78 43           sta nbtemp+1
02609  e121  f0 02              beq gn12
02610  e123  a2 16              ldx #dirlen-2
02611  e125  88          gn12   dey               to point to lo of number blocks
02612  e126  b1 27              lda (dirbuf),y
02613  e128  8d 77 43           sta nbtemp        number of blocks (temporary)
02614  e12b  e0 16              cpx #dirlen-2
02615  e12d  f0 0a              beq gn14
02616  e12f  c9 0a              cmp #10           if <10
02617  e131  90 06              bcc gn14
02618  e133  ca                 dex               less padding needed: number is at
                                                  least 2 digits
02619  e134  c9 64              cmp #100          .A = number blocks lo
02620  e136  90 01              bcc gn14          if <100
02621  e138  ca                 dex               less padding needed: number is at
                                                  least 3 digits
02622  e139  20 a9 e1    gn14   jsr blknb         Blank the name buffer. Y=0
02623  e13c  b1 27              lda (dirbuf),y    save file type
02624  e13e  48                 pha
02625  e13f  0a                 asl a             set C if valid unclosed file, see
                                                  BCS in GN15)
02626  e140  10 05              bpl gn15          if .A<128
02627  e142  a9 3c              lda #'<'
02628  e144  9d b5 41           sta nambuf+1,x
02629  e147
02630  e147  ------------------------------------------------------------------
02631  e147  The above code produces "locked" filetypes like PRG< in the directory
02632  e147  which can't be scratched. The feature is NOT supported by any
02633  e147  Commodore DOS. To lock a file, change the file type in the directory
02634  e147  track from $8x to $Cx. Reverse this to unlock. Files CAN be renamed!
02635  e147  ------------------------------------------------------------------
02636  e147
02637  e147  68          gn15   pla               type
02638  e148  29 0f              and #$0f          mask hi bits
02639  e14a  a8                 tay               use as index
02640  e14b  b9 de d2           lda tp2lst,y      move last character of type
02641  e14e  9d b4 41           sta nambuf,x
02642  e151  ca                 dex
02643  e152  b9 d9 d2           lda tp1lst,y      move middle character
02644  e155  9d b4 41           sta nambuf,x
```

```
line   addr  object      source code

02645  e158  ca                dex               move first character
02646  e159  b9 d4 d2          lda typlst,y      DEL, SEQ, PRG, USR, REL
02647  e15c  9d b4 41          sta nambuf,x
02648  e15f  ca                dex
02649  e160  ca                dex
02650  e161  b0 05             bcs gn20          if C is set (see GN14) entry is
                                                  valid
02651  e163  a9 2a             lda #'*'          file not closed
02652  e165  9d b5 41          sta nambuf+1,x
02653  e168  a9 a0       gn20   lda #$a0          shifted space between name and type
02654  e16a  9d b4 41          sta nambuf,x
02655  e16d  ca                dex
02656  e16e  a0 12             ldy #18           to point to end of name
02657  e170  b1 27       gn22   lda (dirbuf),y    transfer 18-2 characters to
02658  e172  9d b4 41          sta nambuf,x      name buffer
02659  e175  ca                dex
02660  e176  88                dey
02661  e177  c0 03             cpy #$03
02662  e179  b0 f5             bcs gn22
02663  e17b  a9 22             lda #'"'          send name in quotes
02664  e17d  9d b4 41          sta nambuf,x
02665  e180  e8          gn30   inx                 scan name for quote or shifted
                                                  space
02666  e181  e0 20             cpx #$20            when found, or end of name
                                                  reached,
02667  e183  b0 0b             bcs gn35            store a " at that location, then
                                                  AND
02668  e185  bd b4 41          lda nambuf,x        any remaining characters with $7F
                                                  to
02669  e188  c9 22             cmp #'"'            clear bit 7
02670  e18a  f0 04             beq gn35
02671  e18c  c9 a0             cmp #$a0
02672  e18e  d0 f0             bne gn30
02673  e190  a9 22       gn35   lda #'"'
02674  e192  9d b4 41          sta nambuf,x
02675  e195  e8          gn37   inx
02676  e196  e0 20             cpx #$20
02677  e198  b0 0a             bcs gn40
02678  e19a  a9 7f             lda #$7f
02679  e19c  3d b4 41          and nambuf,x
02680  e19f  9d b4 41          sta nambuf,x
02681  e1a2  10 f1             bpl gn37
02682  e1a4  20 e1 de    gn40   jsr fndfil        Continue scan of directory
02683  e1a7  38                sec
02684  e1a8  60          gn45   rts
02685  e1a9
02686  e1a9
02687  e1a9  ===> Blank the name buffer <===
02688  e1a9
02689  e1a9  a0 1b       blknb  ldy #nbsiz        length of name buffer
02690  e1ab  a9 20              lda #$20
02691  e1ad  99 b3 41    blknb1 sta nambuf-1,y
02692  e1b0  88                 dey
```

```
line   addr  object      source code

02693  e1b1  d0 fa               bne blknb1
02694  e1b3  60                  rts
02695  e1b4
02696  e1b4
02697  e1b4  ===> New directory in listing <===
02698  e1b4
02699  e1b4  20 a9 e1   newdir   jsr blknb        Blank the name buffer
02700  e1b7  a9 ff               lda #$ff
02701  e1b9  85 04               sta temp         temporary work area
02702  e1bb  a6 12               ldx drvnum       current drive number
02703  e1bd  8e 77 43            stx nbtemp
02704  e1c0  a9 00               lda #$00
02705  e1c2  8d 78 43            sta nbtemp+1
02706  e1c5  bd e8 d2            lda ipbm,x       BAM address hi
02707  e1c8  85 28               sta dirbuf+1     current buffer pointer hi
02708  e1ca  a9 90               lda #$90         lo byte of pointer
02709  e1cc  85 27               sta dirbuf       directory buffer pointer
02710  e1ce  a0 16               ldy #22          name length
02711  e1d0  b1 27               lda (dirbuf),y   test for shifted blank
02712  e1d2  c9 a0               cmp #$a0
02713  e1d4  d0 0b               bne nd20
02714  e1d6  a9 31               lda #'1'         not shifted blank, so indicate
                                                  version #1
02715  e1d8  2c                  .byte $2c        to branch to ND20
02716  e1d9  b1 27       nd15    lda (dirbuf),y   test for shifted blank
02717  e1db  c9 a0               cmp #$a0
02718  e1dd  d0 02               bne nd20
02719  e1df  a9 20               lda #$20         not shifted, so load blank
02720  e1e1  99 b6 41    nd20    sta nambuf+2,y
02721  e1e4  88                  dey              .Y>=0: more characters to do
02722  e1e5  10 f2               bpl nd15
02723  e1e7  a9 12               lda #$12         RVS on
02724  e1e9  8d b4 41            sta nambuf
02725  e1ec  a9 22               lda #'"'         quote
02726  e1ee  8d b5 41            sta nambuf+1
02727  e1f1  8d c6 41            sta nambuf+18
02728  e1f4  a9 20               lda #$20         space
02729  e1f6  8d c7 41            sta nambuf+19
02730  e1f9  60                  rts
02731  e1fa
02732  e1fa
02733  e1fa  ===> Set up message "blocks free" <===
02734  e1fa
02735  e1fa  20 a9 e1   msgfre   jsr blknb        Blank the name buffer
02736  e1fd  a0 0b               ldy #msglen-1
02737  e1ff  b9 0b e2   msgl     lda fremsg,y     Message "BLOCKS FREE"
02738  e202  99 b4 41            sta nambuf,y
02739  e205  88                  dey
02740  e206  10 f7               bpl msgl
02741  e208  4c 34 db            jmp numfre
02742  e20b
02743  e20b  42 4c 4f   fremsg .byte 'blocks free.'
02744  e20e  43 4b 53
```

```
line   addr  object      source code

02745  e211  20 46 52
02746  e214  45 45 2e
02747  e217
02748  e217              msglen = *-fremsg
02749  e217
02750  e217
02751  e217  ===> NEW (HEADER) a disk <===
02752  e217
02753  e217  20 10 dd    new     jsr onedrv        set up drive and table pointers
02754  e21a  a5 8b               lda fildat        drive number, pattern
02755  e21c  10 05               bpl n101          if bit 7 set, legal drive number
02756  e21e  a9 33               lda #badfn
02757  e220  4c c9 db            jmp cmderr        Command level error handling
02758  e223
02759  e223  29 01       n101    and #$01          mask off non-drive bits
02760  e225  85 12               sta drvnum
02761  e227  20 35 da            jsr set1ds        Turn on LED for current drive
02762  e22a  20 89 d7            jsr setbmp        Set (indirect) BAM pointer
02763  e22d  a5 12               lda drvnum
02764  e22f  0a                  asl a
02765  e230  aa                  tax
02766  e231  ac 81 43            ldy filtbl+1      get disk ID
02767  e234  cc 79 43            cpy cmdsiz        new or clear?
02768  e237  f0 16               beq n108          end of command string
02769  e239  b9 00 43            lda cmdbuf,y      command buffer
02770  e23c  9d 40 43            sta dskid,x       store in proper drive
02771  e23f  b9 01 43            lda cmdbuf+1,y    (.Y=0)
02772  e242  9d 41 43            sta dskid+1,x
02773  e245  a9 01               lda #$01
02774  e247  85 13               sta track         current track number
02775  e249  20 20 e4            jsr format        transfer FORMAT to RAM
02776  e24c  4c 5e e2            jmp n110
02777  e24f
02778  e24f
02779  e24f  ===> Clear directory <===
02780  e24f
02781  e24f  20 ff ec    n108    jsr initdr
02782  e252  a0 02               ldy #$02
02783  e254  b1 02               lda (bmpnt),y     use current version number
02784  e256  cd 9f 10            cmp vernum        "a" - DOS version number
02785  e259  f0 03               beq n110
02786  e25b  4c 80 f1            jmp vnerr         Version error
02787  e25e
02788  e25e  a9 00       n110    lda #$00
02789  e260  a8                  tay
02790  e261  91 02       n111    sta (bmpnt),y     clear buffer
02791  e263  c8                  iny
02792  e264  d0 fb               bne n111
02793  e266  a5 12               lda drvnum        current drive number
02794  e268  18                  clc
02795  e269  69 0c               adc #bamjob       buffer number 13 + drive number
02796  e26b  85 a1               sta jobnum        current job number
02797  e26d  0a                  asl a             multiply job code
```

```
line   addr  object     source code

02798  e26e  aa               tax
02799  e26f  a9 90            lda #$90          disk name offset in BAM
02800  e271  95 29            sta buftab,x
02801  e273  a0 01            ldy #$01
02802  e275  84 14            sty sector        current sector number
02803  e277  a9 ff            lda #$ff
02804  e279  91 02            sta (bmpnt),y     bit map pointer
02805  e27b  a9 12            lda #18
02806  e27d  85 13            sta track         current track number
02807  e27f  20 5b f0         jsr drtwrt        clear directory
02808  e282  20 76 e7         jsr newmap        new BAM
02809  e285  a0 02            ldy #$02
02810  e287  ad 9f 10         lda vernum        "a" - DOS version number
02811  e28a  91 02            sta (bmpnt),y
02812  e28c  20 9f eb         jsr usedts        used 18.1
02813  e28f  c6 14            dec sector        current sector number
02814  e291  20 9f eb         jsr usedts        used 18.0
02815  e294  a4 a1            ldy jobnum        current job number
02816  e296  ae 80 43         ldx filtbl        filtbl ;table of filename pointers
02817  e299  a9 27            lda #$27
02818  e29b  20 69 e0         jsr trname        Transfer filename from command
                                                 string to buffer
02819  e29e  a0 12            ldy #$12
02820  e2a0  a5 12            lda drvnum        set up current ID
02821  e2a2  0a               asl a
02822  e2a3  aa               tax
02823  e2a4  bd 40 43         lda dskid,x
02824  e2a7  91 27            sta (dirbuf),y
02825  e2a9  c8               iny
02826  e2aa  bd 41 43         lda dskid+1,x
02827  e2ad  91 27            sta (dirbuf),y    directory buffer pointer
02828  e2af  c8               iny
02829  e2b0  c8               iny
02830  e2b1  a9 04            lda #dosver+2
02831  e2b3  91 27            sta (dirbuf),y
02832  e2b5  c8               iny
02833  e2b6  ad 9f 10         lda vernum        show version number
02834  e2b9  91 27            sta (dirbuf),y
02835  e2bb  20 5b f0         jsr drtwrt        write it out
02836  e2be  4c 9f db         jmp endcmd        Terminate command successfully
02836  e2c1
02837  e2c1                   .lib scrtch
```

```
line   addr  object      source code

02839  e2c1  ===> Scratch one or more files <===
02840  e2c1
02841  e2c1  20 95 dd    scrtch jsr fslset       Set pointers to one file stream and
                                                 check type
02842  e2c4  20 1e dd           jsr alldrs       Set up all drives from F2CNT
02843  e2c7  20 10 de           jsr optsch       Determine optimal search for LOOKUP
                                                 and FINFIL
02844  e2ca  a9 00              lda #$00
02845  e2cc  85 19              sta r0           use as file count
02846  e2ce  20 c9 de           jsr ffst         Find starting entry in directory
02847  e2d1  30 3f              bmi sc30
02848  e2d3  ------------------------------------------------------------------
02849  e2d3  The following code prevents freeing the sectors
02850  e2d3  of an unclosed file
02851  e2d3  ------------------------------------------------------------------
02852  e2d3  20 bf f8   sc15    jsr tstchn       Test for active files
02853  e2d6  90 35              bcc sc25         yes -- don't scratch
02854  e2d8  ------------------------------------------------------------------
02855  e2d8  The following code prevents scratching a locked filename
02856  e2d8  (bit 6 set)
02857  e2d8  ------------------------------------------------------------------
02858  e2d8  a0 00              ldy #$00
02859  e2da  b1 27              lda (dirbuf),y
02860  e2dc  29 40              and #%01000000   lock bit
02861  e2de  d0 2d              bne sc25         it's locked
02862  e2e0  ------------------------------------------------------------------
02863  e2e0  20 45 e3           jsr deldir       Delete the entry in the directory
02864  e2e3  a0 13              ldy #19          a relative?
02865  e2e5  b1 27              lda (dirbuf),y   with a side sector?
02866  e2e7  f0 0a              beq sc17         no
02867  e2e9  85 13              sta track        yes -- save track number
02868  e2eb  c8                 iny
02869  e2ec  b1 27              lda (dirbuf),y   get sector
02870  e2ee  85 14              sta sector
02871  e2f0  20 1d e3           jsr delfil       Delete by links
02872  e2f3  ------------------------------------------------------------------
02873  e2f3  This prevents freeing a file's sectors if its replacement was
02874  e2f3  incomplete (bit 5 set)
02875  e2f3  ------------------------------------------------------------------
02876  e2f3  ae 45 43   sc17    ldx entfnd       directory entry found flag
02877  e2f6  a9 20              lda #$20
02878  e2f8  35 8b              and fildat,x     drive number, pattern
02879  e2fa  d0 0f              bne sc20         created -- not closed
02880  e2fc  ------------------------------------------------------------------
02881  e2fc  bd 86 43           lda filtrk,x     delete by links
02882  e2ff  29 7f              and #%01111111
02883  e301  85 13              sta track
02884  e303  bd 8b 43           lda filsec,x
02885  e306  85 14              sta sector
02886  e308  20 1d e3           jsr delfil       free the file blocks by updating the
                                                 BAM
02887  e30b  e6 19      sc20    inc r0           set file counter to
02888  e30d  20 b7 de   sc25    jsr ffre         find next name in file stream and
                                                 table entry
```

```
line   addr  object      source code

02889  e310  10 c1              bpl scl5
02890  e312  a5 19       sc30   lda r0            finished, set
02891  e314  85 13              sta track         number of files that have been
                                                   scratched
02892  e316  a9 01              lda #$01
02893  e318  a0 00              ldy #$00
02894  e31a  4c a9 db           jmp scrend        Scratch entry
02895  e31d
02896  e31d
02897  e31d  ===> Delete a file by links <===
02898  e31d
02899  e31d  20 1c da   delfil jsr frets         Free track/sector in BAM
02900  e320  20 6c f0          jsr opnird        update BAM
02901  e323  a9 00       del2   lda #$00
02902  e325  20 c1 f0          jsr setpnt        Set up pointer into active data
                                                   buffer
02903  e328  20 d7 ed          jsr rdbyt         Read byte from file
02904  e32b  85 13              sta track         current track number
02905  e32d  20 d7 ed          jsr rdbyt         Read byte from file
02906  e330  85 14              sta sector        current sector number
02907  e332  a5 13              lda track         current track number
02908  e334  d0 06              bne del1
02909  e336  20 55 f6          jsr mapout        Write out BAM to drive specified in
                                                   LSTJOB
02910  e339  4c a4 ee          jmp frechn        Free channel associated with SA
02911  e33c
02912  e33c  20 1c da   del1   jsr frets         Free track/sector in BAM
02913  e33f  20 44 f0          jsr nxtbuf        read next block using t/s link
02914  e342  4c 23 e3          jmp del2          deallocate new block
02915  e345
02916  e345
02917  e345  ===> Delete the entry in the directory <===
02918  e345
02919  e345  a0 00       deldir ldy #$00
02920  e347  98                tya
02921  e348  91 27              sta (dirbuf),y    directory buffer pointer
02922  e34a  20 60 f9          jsr wrtout        store write job code $90
02923  e34d  4c 87 ec          jmp watjob        Wait until job is completed
02924  e350
02925  e350
02926  e350  ===> Duplicate disk <===
02927  e350
02928  e350  20 a8 e4   duplct jsr prseq
02929  e353  a5 8c              lda fildat+1
02930  e355  85 12              sta drvnum        current drive number
02931  e357  a9 18              lda #led0+led1
02932  e359  0d 82 02           ora pbd2
02933  e35c  8d 82 02           sta pbd2
02934  e35f  20 ff ec           jsr initdr
02935  e362  20 89 d7           jsr setbmp        Set (indirect) BAM pointer
02936  e365  a0 02              ldy #$02
02937  e367  b1 02              lda (bmpnt),y     bit map pointer
02938  e369  cd 9f 10           cmp vernum        "a" - DOS version number
```

```
line   addr  object      source code

02939  e36c  f0 03             beq dupl
02940  e36e  4c 80 f1          jmp vnerr        Version error
02941  e371
02942  e371  20 8c dd   dupl   jsr togdrv       Toggle drive number
02943  e374  0a               asl a
02944  e375  a8               tay
02945  e376  49 02            eor #$02
02946  e378  aa               tax
02947  e379  bd 40 43         lda dskid,x      current id drive #0
02948  e37c  99 40 43         sta dskid,y
02949  e37f  bd 41 43         lda dskid+1,x
02950  e382  99 41 43         sta dskid+1,y
02951  e385  20 89 d7         jsr setbmp       Set (indirect) BAM pointer
02952  e388  a0 02            ldy #$02
02953  e38a  ad 9f 10         lda vernum       "a" - DOS version number
02954  e38d  91 02            sta (bmpnt),y    bit map pointer
02955  e38f  20 54 ef         jsr cldchn       Close all channels except the
                                                command channel
02956  e392  a9 01            lda #$01
02957  e394  85 13            sta track        current track number
02958  e396  20 20 e4         jsr format       Format a diskette routine
02959  e399
02960  e399
02961  e399  ===> Copy blocks to the other drive <===
02962  e399
02963  e399  a5 13      cpyd1  lda track        current track number
02964  e39b  20 db d7         jsr maxsec       Tell how many sectors allowed for
                                                this track
02965  e39e  85 14            sta sector       current sector number
02966  e3a0  c6 14            dec sector
02967  e3a2  20 b3 e3         jsr cpytrk       Copy one track
02968  e3a5  e6 13            inc track        current track number
02969  e3a7  a5 13            lda track
02970  e3a9  c9 24            cmp #maxtrk
02971  e3ab  d0 ec            bne cpyd1        Copy blocks to the other drive
02972  e3ad  20 ff ec         jsr initdr
02973  e3b0  4c 9f db         jmp endcmd
02974  e3b3
02975  e3b3
02976  e3b3  ===> Copy one track <===
02977  e3b3
02978  e3b3  20 c1 e3   cpytrk jsr setrh
02979  e3b6  20 dc e3         jsr reads        Read 10 sectors
02980  e3b9  20 00 e4         jsr writes       Write 10 sectors
02981  e3bc  a5 14            lda sector       current sector number
02982  e3be  10 f3            bpl cpytrk       Copy one track
02983  e3c0  60               rts
02984  e3c1
02985  e3c1  a5 12      setrh  lda drvnum       current drive number
02986  e3c3  49 01            eor #1
02987  e3c5  8d 3c 43         sta cmd          temporary job command
02988  e3c8  a9 0a            lda #10
02989  e3ca  85 06            sta t2
```

```
line   addr  object     source code

02990  e3cc  a5 06      setr3  lda t2
02991  e3ce  20 97 ec          jsr seth
02992  e3d1  c6 14             dec sector
02993  e3d3  30 06             bmi setr6
02994  e3d5  c6 06             dec t2
02995  e3d7  10 f3             bpl setr3
02996  e3d9  e6 06             inc t2
02997  e3db  60         setr6  rts
02998  e3dc
02999  e3dc
03000  e3dc  ===> Read T2 blocks in <===
03001  e3dc
03002  e3dc  ad 3c 43   reads  lda cmd
03003  e3df  09 80             ora #read
03004  e3e1  8d 3c 43          sta cmd
03005  e3e4  a6 06             ldx t2
03006  e3e6  ad 3c 43   readsl lda cmd
03007  e3e9  20 16 f1          jsr setjob      Set up new job
03008  e3ec  e0 0a             cpx #10
03009  e3ee  f0 03             beq reads8
03010  e3f0  e8                inx
03011  e3f1  d0 f3             bne readsl
03012  e3f3  a6 06      reads8 ldx t2
03013  e3f5  20 87 ec   reads3 jsr watjob     Wait until job is completed
03014  e3f8  e0 0a             cpx #10
03015  e3fa  f0 03             beq read15
03016  e3fc  e8                inx
03017  e3fd  d0 f6             bne reads3
03018  e3ff  60         read15 rts
03019  e400
03020  e400
03021  e400  ===> Write T2 buffers out <===
03022  e400
03023  e400  a9 90      writes lda #write
03024  e402  05 12             ora drvnum      current drive number
03025  e404  8d 3c 43          sta cmd         temporary job command
03026  e407  a6 06             ldx t2
03027  e409  20 16 f1   writ0  jsr setjob      Set up new job
03028  e40c  e0 0a             cpx #10
03029  e40e  f0 03             beq writ5
03030  e410  e8                inx
03031  e411  d0 f6             bne writ0
03032  e413  a6 06      writ5  ldx t2
03033  e415  20 87 ec   writ10 jsr watjob     Wait until job is completed
03034  e418  e0 0a             cpx #10
03035  e41a  f0 03             beq writ20
03036  e41c  e8                inx
03037  e41d  d0 f6             bne writ10
03038  e41f  60         writ20 rts
03039  e420
03040  e420
```

```
line   addr  object      source code

03041  e420  ===> Format a diskette routine <===
03042  e420
03043  e420        Transfer format to buffer 0 and start Controller formatting
03044  e420
03045  e420  a0 00       format ldy #$00
03046  e422  b9 00 d0   fmt102 lda code,y
03047  e425  99 00 11          sta bufs,y
03048  e428  b9 00 d1          lda code+256,y
03049  e42b  99 00 12          sta bufs+256,y
03050  e42e  b9 00 d2          lda code+512,y
03051  e431  99 00 13          sta bufs+512,y
03052  e434  c8                iny
03053  e435  d0 eb             bne fmt102
03054  e437  a9 00             lda #$00
03055  e439  20 97 ec          jsr seth
03056  e43c  a5 12             lda drvnum        current drive number
03057  e43e  09 e0             ora #exec
03058  e440  8d 03 10          sta jobs          job queue definitions
03059  e443  ad 03 10   fmt105 lda jobs
03060  e446  30 fb             bmi fmt105
03061  e448  c9 01             cmp #$01
03062  e44a  f0 07             beq fmt110
03063  e44c  a9 03             lda #$03
03064  e44e  a2 00             ldx #$00
03065  e450  4c 25 d9          jmp error         Handle errors reported by controller
03066  e453
03067  e453  60          fmt110 rts
03068  e454
03069  e454
03070  e454  ===> Disk Copy <===  check for type and parse special case
03071  e454
03072  e454  20 e6 db   dskcpy jsr prscln        Find colon in command string
03073  e457  d0 1d             bne dx0000
03074  e459  20 a8 e4          jsr prseq
03075  e45c  a9 2a             lda #'*'          copy all
03076  e45e  a2 27             ldx #39           put at buffer end
03077  e460  8e 81 43          stx filtbl+1
03078  e463  9d 00 43          sta cmdbuf,x      place *
03079  e466  e8                inx
03080  e467  8e 79 43          stx cmdsiz        command string size
03081  e46a  a2 01             ldx #$01          set up counts
03082  e46c  8e 7d 43          stx f1cnt
03083  e46f  e8                inx
03084  e470  8e 7e 43          stx f2cnt
03085  e473  4c eb e4          jmp mov1p2        enter routine
03086  e476
03087  e476  20 f9 db   dx0000 jsr tc30          normal parse
03088  e479  20 1e dd          jsr alldrs        Set up all drives from F2CNT
03089  e47c  ad 91 43          lda image         get parse image
03090  e47f  29 55             and #%01010101    value for patern copy
03091  e481  d0 1b             bne dx0020        must be concat or normal
03092  e483  ae 80 43          ldx filtbl        check for *
03093  e486  bd 00 43          lda cmdbuf,x
```

```
line   addr  object      source code

03094  e489  c9 2a             cmp #'*'
03095  e48b  d0 11             bne dx0020
03096  e48d  a2 01             ldx #$01          set counts
03097  e48f  8e 7d 43          stx f1cnt
03098  e492  e8                inx
03099  e493  8e 7e 43          stx f2cnt
03100  e496  4c cf e4          jmp cpydtd        Copy disk to disk routines
03101  e499
03102  e499  a9 30       dx0010 lda #badsyn
03103  e49b  4c c9 db          jmp cmderr
03104  e49e
03105  e49e  ad 91 43    dx0020 lda image         check for normal
03106  e4a1  25 d9              and %11011001
03107  e4a3  d0 f4              bne dx0010
03108  e4a5  4c 8e e5          jmp copy          Copy file(s) to one file
03109  e4a8
03110  e4a8  a9 3d       prseq  lda #'='          special case
03111  e4aa  20 69 dc          jsr parse
03112  e4ad  d0 05              bne x0020
03113  e4af  a9 30       x0015  lda #badsyn
03114  e4b1  4c c9 db          jmp cmderr
03115  e4b4
03116  e4b4  b9 00 43    x0020  lda cmdbuf,y      command buffer
03117  e4b7  20 bb dd           jsr tst0v1        Test for 0 or 1
03118  e4ba  30 f3              bmi x0015
03119  e4bc  85 8c              sta fildat+1      source drive
03120  e4be  88                 dey
03121  e4bf  88                 dey
03122  e4c0  b9 00 43           lda cmdbuf,y
03123  e4c3  20 bb dd           jsr tst0v1
03124  e4c6  30 e7              bmi x0015
03125  e4c8  c5 8c              cmp fildat+1      can't be equal
03126  e4ca  f0 e3              beq x0015
03127  e4cc  85 8b              sta fildat        destination drive
03128  e4ce  60                 rts
03129  e4cf
03130  e4cf
03131  e4cf  ===> Copy disk to disk routines <===
03132  e4cf
03133  e4cf  ad 81 43   cpydtd lda filtbl+1      save in temp
03134  e4d2  85 04              sta temp
03135  e4d4  a0 28              ldy #40           40-character buffer
03136  e4d6  ae 79 43           ldx cmdsiz        prepare to move
03137  e4d9  8c 79 43           sty cmdsiz        end of filename2
03138  e4dc  88          movlp1 dey
03139  e4dd  ca                 dex
03140  e4de  bd 00 43           lda cmdbuf,x      move filename last-in, first-out
03141  e4e1  99 00 43           sta cmdbuf,y
03142  e4e4  e4 04              cpx temp          actual f2 value
03143  e4e6  d0 f4              bne movlp1
03144  e4e8  8c 81 43           sty filtbl+1      pointer to f2
03145  e4eb  20 10 de   movlp2  jsr optsch        Determine optimal search for LOOKUP
                                                  and FINFIL
```

```
line   addr  object     source code

03146  e4ee  20 71 e5          jsr pupsl       set-up first pass
03147  e4f1  20 c9 de          jsr ffst        first match
03148  e4f4  10 2d             bpl fixit       entry found?
03149  e4f6  30 28             bmi endit       no
03150  e4f8  68          exlp0 pla             pull needed variables
03151  e4f9  8d 96 43          sta dirsec
03152  e4fc  68                pla
03153  e4fd  8d 81 43          sta filtbl+1
03154  e500  68                pla
03155  e501  8d 99 43          sta lstbuf
03156  e504  68                pla
03157  e505  8d 9b 43          sta filcnt
03158  e508  68                pla
03159  e509  8d 9a 43          sta index
03160  e50c  68                pla
03161  e50d  8d 95 43          sta found
03162  e510  68                pla
03163  e511  8d 98 43          sta delind
03164  e514  68                pla
03165  e515  8d 93 43          sta drvflg
03166  e518  20 71 e5          jsr pupsl       set up variables
03167  e51b  20 b7 de          jsr ffre        next match
03168  e51e  10 03             bpl fixit       found one?
03169  e520  4c 9f db    endit jmp endcmd      no, so goodbye!
03170  e523
03171  e523  ad 93 43    fixit lda drvflg      push needed variables
03172  e526  48                pha
03173  e527  ad 98 43          lda delind
03174  e52a  48                pha
03175  e52b  ad 95 43          lda found
03176  e52e  48                pha
03177  e52f  ad 9a 43          lda index
03178  e532  48                pha
03179  e533  ad 9b 43          lda filcnt
03180  e536  48                pha
03181  e537  ad 99 43          lda lstbuf
03182  e53a  48                pha
03183  e53b  ad 81 43          lda filtbl+1
03184  e53e  48                pha
03185  e53f  ad 96 43          lda dirsec
03186  e542  48                pha
03187  e543  20 61 e5          jsr trfnme      Transfer name (DIRBUF) to CMDBUF
03188  e546  a9 01             lda #$01        fake out LOOKUP
03189  e548  8d 7d 43          sta flcnt
03190  e54b  8d 7e 43          sta f2cnt
03191  e54e  20 7a de          jsr lookup      Look up files in cmd string in
                                               directory & fill tables
03192  e551  a9 01             lda #$01
03193  e553  8d 7d 43          sta flcnt
03194  e556  a9 02             lda #$02        real
03195  e558  8d 7e 43          sta f2cnt
03196  e55b  20 da e5          jsr cy          copy it
03197  e55e  4c f8 e4          jmp exlp0       now do next one
```

```
line   addr  object      source code

03198  e561
03199  e561
03200  e561  ===> Transfer name (DIRBUF) to CMDBUF <===
03201  e561
03202  e561  a0 03       trfnme ldy #$03          both indices
03203  e563  8c 80 43          sty filtbl        beginning of filename1
03204  e566  b1 27       trf0   lda (dirbuf),y   move it
03205  e568  99 00 43          sta cmdbuf,y      command buffer
03206  e56b  c8                iny
03207  e56c  c0 13              cpy #19           all 16 characters passed?
03208  e56e  d0 f6              bne trf0
03209  e570  60                rts
03210  e571
03211  e571
03212  e571  ===> Set-up subroutine <===
03213  e571
03214  e571  a9 00       pups1  lda #$00
03215  e573  8d 92 43          sta drvcnt        number of drive searches
03216  e576  8d 86 43          sta filtrk        first link/track
03217  e579  8d 87 43          sta filtrk+1
03218  e57c  a5 8c              lda fildat+1      get drive number
03219  e57e  29 01              and #$01          only
03220  e580  85 12              sta drvnum        current drive number
03221  e582  09 01              ora #$01
03222  e584  8d 97 43          sta delsec        non-zero
03223  e587  ad 81 43          lda filtbl+1      fn1 = fn2
03224  e58a  8d 80 43          sta filtbl        table of filename pointers
03225  e58d  60                rts
03226  e58e
03227  e58e
03228  e58e  ===> Copy file(s) to one file <===
03229  e58e
03230  e58e  20 7a de    copy   jsr lookup        Look up files in cmd string in
                                                 directory & fill tables
03231  e591  ad 7e 43          lda f2cnt         number of filenames
03232  e594  c9 03              cmp #$03          fewer than 3, not concatenate
03233  e596  90 3c              bcc copl0         copy file
03234  e598  a5 8b              lda fildat        drive number, pattern
03235  e59a  c5 8c              cmp fildat+1      if unequal, not concatenate
03236  e59c  d0 36              bne copl0         copy file
03237  e59e  a5 86              lda filent        table of sector numbers in directory
03238  e5a0  c5 87              cmp filent+1      if unequal, not concatenate
03239  e5a2  d0 30              bne copl0         Copy file
03240  e5a4
03241  e5a4
03242  e5a4  ===> Concatenate files <===
03243  e5a4
03244  e5a4  20 c3 e6           jsr chkin         check if input file exists
03245  e5a7  a9 01              lda #$01
03246  e5a9  8d 7f 43          sta f2ptr         file stream 2 pointer
03247  e5ac  20 1e e6           jsr opirfl        Open internal read channel to read
                                                 file
03248  e5af  20 a6 ed           jsr typfil        Get current file type
```

```
line   addr  object      source code

03249  e5b2  f0 04             beq cop01          0 means a scratched file
03250  e5b4  c9 02             cmp #prgtyp        if not 2,
03251  e5b6  d0 05             bne cop05          not deleted
03252  e5b8  a9 64       cop01 lda #mistyp
03253  e5ba  20 c9 db           jsr cmderr         Command level error handling
03254  e5bd  a9 12       cop05 lda #iwsa
03255  e5bf  85 16             sta sa             current secondary address
03256  e5c1  a5 b3             lda lintab+irsa
03257  e5c3  85 b4             sta lintab+iwsa
03258  e5c5  a9 ff             lda #$ff           deactivate
03259  e5c7  85 b3             sta lintab+irsa
03260  e5c9  20 df f4           jsr append         Read file, then append info to the
                                                  end of it
03261  e5cc  a2 02             ldx #$02
03262  e5ce  20 ec e5           jsr cy10           copy 2nd file behind 1st
03263  e5d1  4c 9f db           jmp endcmd         Terminate command successfully
03264  e5d4
03265  e5d4  20 da e5    cop10  jsr cy
03266  e5d7  4c 9f db           jmp endcmd         Terminate command successfully
03267  e5da
03268  e5da
03269  e5da  ===> Copy file <===
03270  e5da
03271  e5da  20 e0 e6    cy     jsr chkio          Check existence of I/O file
03272  e5dd  a5 8b              lda fildat         drive number, pattern
03273  e5df  29 01              and #$01
03274  e5e1  85 12              sta drvnum         current drive number
03275  e5e3  20 7c f0           jsr opnirw         Open internal write channel (SA=18)
03276  e5e6  20 a9 f1           jsr addfil         Add new filename to directory
03277  e5e9  ae 7d 43           ldx f1cnt
03278  e5ec  8e 7f 43    cy10   stx f2ptr          set op read file
03279  e5ef  20 1e e6           jsr opirf1         Open internal read channel to read
                                                  file
03280  e5f2  a9 08              lda #eoisnd
03281  e5f4  85 a0              sta eoiflg         current EOI status
03282  e5f6  4c fc e5           jmp cy20
03283  e5f9
03284  e5f9  20 eb eb    cy15   jsr pibyte         Write byte to internal channel
03285  e5fc  20 5e e6    cy20   jsr gibyte         Get byte from internal read channels
03286  e5ff  a9 80              lda #lrf
03287  e601  20 ae f8           jsr tstflg         Test flag
03288  e604  f0 f3              beq cy15
03289  e606  20 a6 ed           jsr typfil         Get current file type
03290  e609  f0 03              beq cy30
03291  e60b  20 eb eb           jsr pibyte         Write byte to internal channel
03292  e60e  ae 7f 43    cy30   ldx f2ptr          check if more files to copy
03293  e611  e8                 inx
03294  e612  ec 7e 43           cpx f2cnt          if carry clear,
03295  e615  90 d5              bcc cy10           more files to copy
03296  e617  a9 12              lda #iwsa
03297  e619  85 16              sta sa             current secondary address
03298  e61b  4c ba f5           jmp clschn         Close file with specified secondary
                                                  address
```

```
line   addr  object     source code

03299  e61e
03300  e61e
03301  e61e  ===> Open internal read channel to read file <===
03302  e61e
03303  e61e  ae 7f 43   opirfl ldx f2ptr        file stream 2 pointer
03304  e621  b5 8b             lda fildat,x     drive number, pattern
03305  e623  29 01             and #$01
03306  e625  85 12             sta drvnum       current drive number
03307  e627  a9 12             lda #18
03308  e629  85 13             sta track        current track number
03309  e62b  b5 86             lda filent,x     table of sector numbers in directory
03310  e62d  29 1f             and #%11111
03311  e62f  85 14             sta sector       current sector number
03312  e631  20 6c f0          jsr opnird       Open internal read channel (SA=17)
03313  e634  ae 7f 43          ldx f2ptr        file stream 2 pointer
03314  e637  b5 86             lda filent,x     table of sector numbers in directory
03315  e639  29 70             and #%1110000
03316  e63b  09 02             ora #$02
03317  e63d  20 c1 f0          jsr setpnt       Set up pointer into active data
                                                buffer
03318  e640  ae 7f 43          ldx f2ptr        file stream 2 pointer
03319  e643  b5 8b             lda fildat,x     drive number, pattern
03320  e645  29 0e             and #typmsk
03321  e647  4a                lsr a
03322  e648  85 c5             sta type         current file type
03323  e64a  a9 00             lda #$00
03324  e64c  8d 4b 43          sta rec          not relative
03325  e64f  20 5b f4          jsr opread       Open a file to read
03326  e652  a0 01             ldy #$01
03327  e654  20 a6 ed          jsr typfil       Get current file type
03328  e657  f0 01             beq opirl0       if Z set (not relative file)
03329  e659  c8                iny
03330  e65a  98         opirl0 tya
03331  e65b  4c c1 f0          jmp setpnt       Set up pointer into active data
                                                buffer
03332  e65e
03333  e65e  ===> Get byte from internal read channels <===
03334  e65e
03335  e65e  a9 11      gibyte lda #irsa
03336  e660  85 16             sta sa           current secondary address
03337  e662  20 95 ef   gcbyte jsr gbyte        Get next byte from channel
03338  e665  85 18             sta data         temporary data byte
03339  e667  a6 15             ldx lindx        logical index, channel#
03340  e669  b5 98             lda chnrdy,x     write, read, eoi flags, channel
                                                status
03341  e66b  29 08             and #eoisnd
03342  e66d  85 a0             sta eoiflg       current EOI status
03343  e66f  d0 0a             bne gib20
03344  e671  20 a6 ed          jsr typfil       Get current file type
03345  e674  f0 05             beq gib20        if Z set: not a relative file
03346  e676  a9 80             lda #lrf         last record flag
03347  e678  20 9f f8          jsr setflg       Set flag
03348  e67b  60         gib20  rts
```

```
line   addr  object      source code

03349  e67c
03350  e67c
03351  e67c  ===> Rename file in directory <===
03352  e67c
03353  e67c  20 1e dd   rename jsr alldrs       Set up both drives from F2CNT
03354  e67f  a5 8c             lda fildat+1
03355  e681  29 01             and #$01
03356  e683  85 8c             sta fildat+1
03357  e685  c5 8b             cmp fildat       drive number, pattern
03358  e687  f0 02             beq rn10         same drive numbers
03359  e689  09 80             ora #$80         check both for name
03360  e68b  85 8b       rn10  sta fildat       drive number, pattern
03361  e68d  20 7a de           jsr lookup       Look up files in cmd string in
                                                directory & fill tables
03362  e690  20 e0 e6           jsr chkio        Check existence of I/O file
03363  e693  a5 8c             lda fildat+1
03364  e695  29 01             and #$01
03365  e697  85 12             sta drvnum       current drive number
03366  e699  a5 87             lda filent+1
03367  e69b  48                 pha
03368  e69c  29 1f             and #%11111
03369  e69e  85 14             sta sector       current sector number
03370  e6a0  20 59 f9           jsr rdab         read directory sector
03371  e6a3  20 87 ec           jsr watjob
03372  e6a6  68                 pla
03373  e6a7  29 70             and #%1110000    set sector index
03374  e6a9  09 05             ora #$05         ...+5
03375  e6ab  20 c1 f0           jsr setpnt       Set up pointer into active data
                                                buffer
03376  e6ae  20 95 fa           jsr getact       Get active buffer number
03377  e6b1  a8                 tay
03378  e6b2  ae 80 43           ldx filtbl       table of filename pointers
03379  e6b5  a9 10             lda #16          number of characters in name
03380  e6b7  20 69 e0           jsr trname       Transfer filename from command
                                                string to buffer
03381  e6ba  20 60 f9           jsr wrtout       write revised sector
03382  e6bd  20 87 ec           jsr watjob       Wait until job is completed
03383  e6c0  4c 9f db           jmp endcmd       Terminate command successfully
03384  e6c3
03385  e6c3
03386  e6c3  ===> Check existence of input file <===
03387  e6c3
03388  e6c3  a5 8c       chkin  lda fildat+1
03389  e6c5  29 0e             and #%1110
03390  e6c7  4a                 lsr a
03391  e6c8  85 c5             sta type         current file type
03392  e6ca  ae 7e 43           ldx f2cnt        file stream 2 count
03393  e6cd  ca           ck10  dex
03394  e6ce  ec 7d 43           cpx f1cnt        file stream 1 count
03395  e6d1  90 0c             bcc ck20         C clear: found
03396  e6d3  bd 86 43           lda filtrk,x     first link/track
03397  e6d6  29 7f             and #%01111111   if link not 0, file
03398  e6d8  d0 f3             bne ck10         is found
```

```
line   addr  object      source code

03399  e6da  a9 62             lda #flntfd
03400  e6dc  4c c9 db          jmp cmderr       Command level error handling
03401  e6df  60          ck20  rts
03402  e6e0
03403  e6e0
03404  e6e0  ===> Check existence of I/O file <===
03405  e6e0
03406  e6e0  20 c3 e6    chkio jsr chkin        Check existence of input file
03407  e6e3  bd 86 43    ck25  lda filtrk,x     first link/track
03408  e6e6  29 7f              and #%01111111
03409  e6e8  f0 05              beq ck30
03410  e6ea  a9 63              lda #flexst
03411  e6ec  4c c9 db           jmp cmderr       Command level error handling
03412  e6ef
03413  e6ef  ca          ck30  dex
03414  e6f0  10 f1              bpl ck25
03415  e6f2  60                 rts
03415  e6f3
03416  e6f3                     .lib verdir
```

```
line   addr  object      source code

03418  e6f3  ===> Validate files with BAM, update BAM <===
03419  e6f3
03420  e6f3             verdir
03421  e6f3  20 d2 db   valdat jsr simprs       extract drive number
03422  e6f6  20 ff ec          jsr initdr       for name, ID
03423  e6f9  20 73 e7          jsr newmpv       Set up new BAM
03424  e6fc  a9 00             lda #$00
03425  e6fe  8d 98 43          sta delind       index of first available entry
03426  e701  20 da df          jsr srchst       Initiate search of directory
03427  e704  d0 39             bne vd25         found a file
03428  e706  a9 00       vd10  lda #$00         set directory sectors
03429  e708  85 14             sta sector       in BAM
03430  e70a  a9 12             lda #18
03431  e70c  85 13             sta track        current track number
03432  e70e  20 4b e7          jsr mrkbam       Mark BAM with file sectors
03433  e711  a5 12             lda drvnum       current drive number
03434  e713  20 5c f6          jsr mo10         write out BAM
03435  e716  4c 9f db          jmp endcmd       Terminate command successfully
03436  e719
03437  e719  c8          vd15  iny
03438  e71a  b1 27             lda (dirbuf),y
03439  e71c  48                pha              save track
03440  e71d  c8                iny
03441  e71e  b1 27             lda (dirbuf),y
03442  e720  48                pha              save sector
03443  e721  a0 13             ldy #19          get SS track
03444  e723  b1 27             lda (dirbuf),y   is this relative?
03445  e725  f0 0a             beq vd17         no
03446  e727  85 13             sta track        yes -- save track number
03447  e729  c8                iny
03448  e72a  b1 27             lda (dirbuf),y   get SS sector
03449  e72c  85 14             sta sector
03450  e72e  20 4b e7          jsr mrkbam       validate SS by links
03451  e731  68          vd17  pla
03452  e732  85 14             sta sector       now do data blocks
03453  e734  68                pla
03454  e735  85 13             sta track
03455  e737  20 4b e7          jsr mrkbam       set bit used in BAM
03456  e73a  20 31 e0    vd20  jsr srre         search for more
03457  e73d  f0 c7             beq vd10         no more files
03458  e73f  a0 00       vd25  ldy #$00
03459  e741  b1 27             lda (dirbuf),y   directory buffer pointer
03460  e743  30 d4             bmi vd15
03461  e745  20 45 e3          jsr deldir       not closed - delete the entry in the
                                                 directory
03462  e748  4c 3a e7          jmp vd20
03463  e74b
03464  e74b
03465  e74b  ===> Mark BAM with file sectors <===
03466  e74b
03467  e74b  20 89 d7   mrkbam jsr setbmp       Set (indirect) BAM pointer
03468  e74e  20 9f eb          jsr usedts       mark track & sector as used
03469  e751  20 6c f0          jsr opnird       Open internal read channel (SA=17)
```

```
line   addr  object     source code

03470  e754  a9 00      mrk2   lda #$00
03471  e756  20 c1 f0           jsr setpnt       Set up pointer into active data
                                                 buffer
03472  e759  20 b8 ed           jsr getbyt       Read one byte from the active buffer
03473  e75c  85 13               sta track        current track number
03474  e75e  20 b8 ed           jsr getbyt       Read one byte from the active buffer
03475  e761  85 14               sta sector       current sector number
03476  e763  a5 13               lda track        current track number
03477  e765  d0 03               bne mrk1
03478  e767  4c a4 ee           jmp frechn       Free channel associated with SA
03479  e76a
03480  e76a  20 9f eb   mrk1     jsr usedts       mark track & sector as used
03481  e76d  20 44 f0           jsr nxtbuf
03482  e770  4c 54 e7           jmp mrk2
03483  e773
03484  e773
03485  e773  ===> Set up new BAM <===
03486  e773
03487  e773  20 89 d7   newmpv  jsr setbmp       Set (indirect) BAM pointer
03488  e776  a0 00      newmap  ldy #$00
03489  e778  a9 12               lda #18          set link to track 18 sector 1
03490  e77a  91 02               sta (bmpnt),y
03491  e77c  c8                  iny
03492  e77d  98                  tya
03493  e77e  91 02               sta (bmpnt),y
03494  e780  c8                  iny
03495  e781  c8                  iny
03496  e782  c8                  iny              .Y = 4
03497  e783  a9 00      nm10    lda #$00         clear track map
03498  e785  85 04               sta t0
03499  e787  85 05               sta t1
03500  e789  85 06               sta t2
03501  e78b  98                  tya
03502  e78c  4a                  lsr a
03503  e78d  4a                  lsr a            .A = track number
03504  e78e  20 db d7           jsr maxsec       Tell how many sectors allowed for
                                                 this track
03505  e791  91 02               sta (bmpnt),y    bit map pointer
03506  e793  c8                  iny
03507  e794  aa                  tax
03508  e795  38        nm20    sec              set map bits
03509  e796  26 04               rol t0
03510  e798  26 05               rol t1
03511  e79a  26 06               rol t2
03512  e79c  ca                  dex
03513  e79d  d0 f6               bne nm20
03514  e79f
03515  e79f  b5 04      nm30    lda temp,x       .X=0
03516  e7a1  91 02               sta (bmpnt),y    bit map pointer
03517  e7a3  c8                  iny
03518  e7a4  e8                  inx
03519  e7a5  e0 03               cpx #$03
03520  e7a7  90 f6               bcc nm30
```

```
line   addr  object     source code

03573  e7f1  a5 05            lda tl          hi byte
03574  e7f3  85 48            sta cb+3
03575  e7f5  4c 3b f0         jmp ge20        continue read
03576  e7f8
03577  e7f8  20 6e ed   m30    jsr fndrch      Find the assigned read channel
03578  e7fb  4c 32 f0         jmp ge15        terminate read
03579  e7fe
03580  e7fe  a9 31       memerr lda #badcmd
03581  e800  4c c9 db         jmp cmderr
03582  e803
03583  e803  b9 06 43   memwrt lda cmdbuf+6,y
03584  e806  91 04            sta (temp),y    transfer from cmdbuf
03585  e808  c8               iny
03586  e809  cc 05 43         cpy cmdbuf+5    number of bytes to write
03587  e80c  90 f5            bcc memwrt
03588  e80e  60               rts
03589  e80f
03590  e80f
03591  e80f  ===> User jump commands <===   U0 restores pointer to JMP table
03592  e80f
03593  e80f  ac 01 43   user   ldy cmdbuf+1
03594  e812  c0 30            cpy #'0'
03595  e814  d0 09            bne us10        0 resets pointer
03596  e816  a9 ea      usrint lda #<ublock    restores normal address ($FFEA)
03597  e818  85 00            sta usrjmp      USR-vector
03598  e81a  a9 ff            lda #>ublock
03599  e81c  85 01            sta usrjmp+1
03600  e81e  60               rts
03601  e81f
03602  e81f  20 25 e8   us10   jsr usrexc      execute code by table
03603  e822  4c 9f db         jmp endcmd      Terminate command successfully
03604  e825
03605  e825  88         usrexc dey             entry is (((index-1)AND$F)*2)
03606  e826  98               tya
03607  e827  29 0f            and #$0f        convert to hex
03608  e829  0a               asl a
03609  e82a  a8               tay
03610  e82b  b1 00            lda (usrjmp),y  USR-vector
03611  e82d  85 0a            sta ip          indirect pointer variable
03612  e82f  c8               iny
03613  e830  b1 00            lda (usrjmp),y  USR-vector
03614  e832  85 0b            sta ip+1
03615  e834  6c 0a 00         jmp (ip)        indirect pointer variable
03616  e837
03617  e837
03618  e837  ===> Open direct access buffer ("#") <===
03619  e837
03620  e837  ae 79 43   opnblk ldx cmdsiz      command string size
03621  e83a  ca               dex
03622  e83b  d0 0d            bne ob10
03623  e83d  a9 01            lda #$01        get any buffer
03624  e83f  20 63 ee         jsr getrch      Open a new read channel
03625  e842  4c 8e e8         jmp ob30
```

```
line   addr  object      source code

03521  e7a9  c0 90              cpy #$90          end of BAM
03522  e7ab  90 d6              bcc nm10
03523  e7ad  60                 rts
03524  e7ae
03525  e7ae  47          echksm .byte $47         checksum E-ROM
03526  e7af
03527  e7af
03528  e7af  ===> Memory access cmds: M-R, M-W, M-E <===
03529  e7af
03530  e7af  ad 01 43    mem    lda cmdbuf+1
03531  e7b2  c9 2d              cmp #'-'          must be second character
03532  e7b4  d0 48              bne memerr
03533  e7b6  ad 03 43           lda cmdbuf+3      set address in temp
03534  e7b9  85 04              sta temp
03535  e7bb  ad 04 43           lda cmdbuf+4
03536  e7be  85 05              sta t1
03537  e7c0  a0 00              ldy #$00
03538  e7c2  ad 02 43           lda cmdbuf+2      third character
03539  e7c5  c9 57              cmp #'w'          M-W
03540  e7c7  f0 3a              beq memwrt
03541  e7c9  c9 52              cmp #'r'          M-R
03542  e7cb  f0 07              beq memrd
03543  e7cd  c9 45              cmp #'e'          M-E
03544  e7cf  d0 2d              bne memerr
03545  e7d1  6c 04 00           jmp (temp)
03546  e7d4
03547  e7d4  b1 04       memrd  lda (temp),y
03548  e7d6  85 18              sta data
03549  e7d8  ad 79 43           lda cmdsiz        command string size
03550  e7db  c9 06              cmp #$06

03552  e7dd
03553  e7dd  ************************************************************
03554  e7dd  Commodore documentation implies that only one byte can be read
03555  e7dd  by M-R, as in PRINT#15,"M-R";chr$(lo);chr$(hi). This results in
03556  e7dd  a command length of five bytes. However, the cmp is for six
03557  e7dd  bytes. Tests have shown that the sixth byte may hold the number
03558  e7dd  of reads — up to 255 — to be carried out, as in
03559  e7dd       PRINT#15,"M-R";chr$(lo);chr$(hi);chr$(number of bytes)
03560  e7dd  ************************************************************
03561  e7dd
03562  e7dd  90 19              bcc m30
03563  e7df  ae 05 43           ldx cmdbuf+5      sixth character
03564  e7e2  ca                 dex               now $00 if only one to read
03565  e7e3  f0 13              beq m30
03566  e7e5  8a                 txa
03567  e7e6  18                 clc
03568  e7e7  65 04              adc temp          add lo byte of last character to be
                                                  sent
03569  e7e9  e6 04              inc temp          to point to second memory location
03570  e7eb  85 c4              sta lstchr+errchn
03571  e7ed  a5 04              lda temp          lo byte
03572  e7ef  85 47              sta cb+2
```

```
line   addr  object      source code

03626  e845
03627  e845  a9 70       ob05   lda #nochnl      no channel error
03628  e847  4c c9 db            jmp cmderr
03629  e84a
03630  e84a  a0 01       ob10   ldy #$01         buffer number requested
03631  e84c  20 17 e9           jsr bp05
03632  e84f  ae 8b 43           ldx filsec       buffer number
03633  e852  e0 0c              cpx #bamjob      must be less than 13
03634  e854  b0 ef              bcs ob05
03635  e856  a9 00              lda #$00
03636  e858  85 04              sta temp
03637  e85a  85 05              sta t1
03638  e85c  38                 sec
03639  e85d  26 04       ob15   rol temp         loop to shift a 1 to position
                                                 according to buf
03640  e85f  26 05              rol t1           e.g. T1 (00000000) TEMP (00000001) =
                                                 buffer 0
03641  e861  ca                 dex                   T1 (00000000) TEMP (00000100) =
                                                 buffer 2
03642  e862  10 f9              bpl ob15               T1 (00000001) TEMP (00000000) =
                                                 buffer 8
03643  e864  a5 04              lda temp
03644  e866  2d 3e 43           and bufuse       indicate buffer in use
03645  e869  d0 da              bne ob05         buffer in use
03646  e86b  a5 05              lda t1           find out which
03647  e86d  2d 3f 43           and bufuse+1     is in use
03648  e870  d0 d3              bne ob05         if not 0, abort
03649  e872  a5 04              lda temp         mark buffer used
03650  e874  0d 3e 43           ora bufuse
03651  e877  8d 3e 43           sta bufuse
03652  e87a  a5 05              lda t1
03653  e87c  0d 3f 43           ora bufuse+1
03654  e87f  8d 3f 43           sta bufuse+1
03655  e882  a9 00              lda #$00         set up channel
03656  e884  20 63 ee           jsr getrch       Open a new read channel
03657  e887  a6 15              ldx lindx        logical index, channel number
03658  e889  ad 8b 43           lda filsec       first link/sector
03659  e88c  95 49              sta buf0,x       channel buffer table 1
03660  e88e  a6 16       ob30   ldx sa           current secondary address
03661  e890  b5 a2              lda lintab,x     set LINDX table
03662  e892  09 40              ora #$40         read/write mode
03663  e894  95 a2              sta lintab,x     current status SA
03664  e896  a4 15              ldy lindx        logical index, channel number
03665  e898  a9 ff              lda #$ff         last character pointer
03666  e89a  99 bd 00           sta lstchr,y     channel last character pointer
03667  e89d  a9 89              lda #rndrdy      ready for random access flag
03668  e89f  99 98 00           sta chnrdy,y     write, read, eoi flags, channel
                                                 status
03669  e8a2  b9 49 00           lda buf0,y       channel buffer table 1
03670  e8a5  99 b5 00           sta chndat,y     buffer number as first character
03671  e8a8  0a                 asl a
03672  e8a9  aa                 tax
03673  e8aa  a9 01              lda #$01
```

```
line   addr  object     source code

03674  e8ac  95 29            sta buftab,x     buffer 0 pointer lo
03675  e8ae  a9 0e            lda #dirtyp+dirtyp
03676  e8b0  99 90 00         sta filtyp,y     set direct access file type
03677  e8b3  4c 9f db         jmp endcmd       Terminate command successfully
03678  e8b6
03679  e8b6
03680  e8b6  ===> Block commands: B-A, B-F, B-R, B-W, B-E, B-P <===
03681  e8b6
03682  e8b6  a0 00       block  ldy #$00
03683  e8b8  a2 00              ldx #$00
03684  e8ba  a9 2d              lda #'-'         separates from subcommand
03685  e8bc  20 69 dc           jsr parse        locate subcommand
03686  e8bf  d0 0a              bne blk40
03687  e8c1  a9 31       blk10  lda #badcmd
03688  e8c3  4c c9 db           jmp cmderr       Command level error handling
03689  e8c6
03690  e8c6  a9 30       blk30  lda #badsyn
03691  e8c8  4c c9 db           jmp cmderr       Command level error handling
03692  e8cb
03693  e8cb  8a          blk40  txa
03694  e8cc  d0 f8              bne blk30
03695  e8ce  a2 05              ldx #nbcmds-1    find command
03696  e8d0  b9 00 43           lda cmdbuf,y
03697  e8d3  dd f8 e8    blk50  cmp bctab,x      Block sub-command table
03698  e8d6  f0 05              beq blk60
03699  e8d8  ca                 dex
03700  e8d9  10 f8              bpl blk50
03701  e8db  30 e4              bmi blk10
03702  e8dd  8a          blk60  txa
03703  e8de  09 80              ora #$80
03704  e8e0  8d 7a 43           sta cmdnum       command#
03705  e8e3  20 0a e9           jsr blkpar       Parse the block parameters
03706  e8e6  ad 7a 43           lda cmdnum       command#
03707  e8e9  0a                 asl a
03708  e8ea  aa                 tax
03709  e8eb  bd ff e8           lda bcjmp+1,x
03710  e8ee  85 05              sta t1
03711  e8f0  bd fe e8           lda bcjmp,x
03712  e8f3  85 04              sta temp
03713  e8f5  6c 04 00           jmp (temp)
03714  e8f8
03715  e8f8
03716  e8f8  ===> Block sub-command table <===
03717  e8f8
03718  e8f8  41 46 52    bctab  .byte 'afrwep'
03719  e8fb  57 45 50
03720  e8fe
03721  e8fe              nbcmds = *-bctab
03722  e8fe
03723  e8fe  99 e9       bcjmp  .word blkalc     block-allocate
03724  e900  90 e9              .word blkfre     block-free
03725  e902  fc e9              .word blkrd      block-read
03726  e904  19 ea              .word blkwt      block-write
```

```
line   addr  object      source code

03727  e906  49 ea             .word blkexc     block execute
03728  e908  60 ea             .word blkptr     block-pointer
03729  e90a
03730  e90a
03731  e90a  ===> Parse the block parameters <===
03732  e90a
03733  e90a  a0 00       blkpar ldy #$00
03734  e90c  a2 00              ldx #$00
03735  e90e  a9 3a              lda #':'
03736  e910  20 69 dc           jsr parse        Store desired character in CHAR
03737  e913  d0 02              bne bp05         found :
03738  e915  a0 03              ldy #$03         else character 3 is beginning
03739  e917  b9 00 43   bp05    lda cmdbuf,y     command buffer
03740  e91a  c9 20              cmp #$20
03741  e91c  f0 08              beq bp10
03742  e91e  c9 1d              cmp #29          skip character
03743  e920  f0 04              beq bp10
03744  e922  c9 2c              cmp #','
03745  e924  d0 07              bne bp20
03746  e926  c8         bp10    iny
03747  e927  cc 79 43           cpy cmdsiz       command string size
03748  e92a  90 eb              bcc bp05
03749  e92c  60                 rts              That's all, folks!
03750  e92d
03751  e92d  20 3c e9   bp20    jsr aschex       Convert ASCII to hex & store
                                                 conversion in tables
03752  e930  ee 7d 43           inc f1cnt        file stream 1 count
03753  e933  ac 7f 43           ldy f2ptr        file stream 2 pointer
03754  e936  e0 04              cpx #$04
03755  e938  90 ec              bcc bp10
03756  e93a  b0 8a              bcs blk30
03757  e93c
03758  e93c
03759  e93c  ===> Convert ASCII to hex & store conversion in tables <===
03760  e93c
03761  e93c  a9 00       aschex lda #$00
03762  e93e  85 04              sta temp         temporary work area
03763  e940  85 05              sta t1
03764  e942  85 07              sta t3
03765  e944  a2 ff              ldx #$ff
03766  e946  b9 00 43   ah10    lda cmdbuf,y     command string byte
03767  e949  c9 40              cmp #$40         numeric?
03768  e94b  b0 18              bcs ah20         non-numeric terminates
03769  e94d  c9 30              cmp #$30         ASCII?
03770  e94f  90 14              bcc ah20
03771  e951  29 0f              and #$0f         mask off hi bits
03772  e953  48                 pha
03773  e954  a5 05              lda t1
03774  e956  85 06              sta t2
03775  e958  a5 04              lda temp
03776  e95a  85 05              sta t1
03777  e95c  68                 pla
03778  e95d  85 04              sta temp
```

```
line   addr  object      source code

03779  e95f  c8                iny
03780  e960  cc 79 43          cpy cmdsiz        if more commands left
03781  e963  90 el             bcc ah10
03782  e965  8c 7f 43   ah20    sty f2ptr         convert digits to binary by decimal
                                                  table
03783  e968  18                clc
03784  e969  a9 00             lda #$00
03785  e96b  e8          ah30   inx
03786  e96c  e0 03             cpx #$03          are we done?
03787  e96e  b0 0f             bcs ah40          yes
03788  e970  b4 04             ldy temp,x
03789  e972  88          ah35   dey
03790  e973  30 f6             bmi ah30
03791  e975  7d 8d e9          adc dectab,x      Decimal conversion table
03792  e978  90 f8             bcc ah35
03793  e97a  18                clc
03794  e97b  e6 07             inc t3
03795  e97d  d0 f3             bne ah35
03796  e97f  48          ah40   pha               .A = hex number
03797  e980  ae 7d 43           ldx flcnt         command segment counter
03798  e983  a5 07              lda t3            carry bit (thousands)
03799  e985  9d 86 43           sta filtrk,x      store result in table
03800  e988  68                 pla
03801  e989  9d 8b 43           sta filsec,x
03802  e98c  60                 rts
03803  e98d
03804  e98d
03805  e98d  ===> Decimal conversion table <===
03806  e98d
03807  e98d  01 0a 64    dectab .byte 1, 10, 100
03808  e990
03809  e990
03810  e990  ===> B-F de-allocate (free) a block in the BAM <===
03811  e990
03812  e990  20 98 ea    blkfre jsr blktst       test for legal block
03813  e993  20 1c da           jsr frets        Free track/sector in BAM
03814  e996  4c 9f db           jmp endcmd       Terminate command successfully
03815  e999
03816  e999  ===> B-A to allocate  block in the BAM <===
03817  e999
03818  e999  20 98 ea    blkalc jsr blktst       test for legal block
03819  e99c  a6 12              ldx drvnum       current drive number
03820  e99e  bd e8 d2           lda ipbm,x       BAM address hi
03821  e9a1  85 03              sta bmpnt+1
03822  e9a3  20 95 d7    ba10   jsr avail        Check BAM for available sector
03823  e9a6  20 b1 d7    ba20   jsr av2
03824  e9a9  b0 26              bcs ba40         search for next available sector
03825  e9ab  a6 14              ldx sector
03826  e9ad  e8                 inx
03827  e9ae  86 14              stx sector       current sector number
03828  e9b0  8e 73 43           stx erword       set not available flag
03829  e9b3  e4 19              cpx r0
03830  e9b5  90 ef              bcc ba20
```

```
line   addr  object     source code

03831  e9b7  a9 00            lda #$00
03832  e9b9  85 14            sta sector
03833  e9bb  a6 13            ldx track        current track number
03834  e9bd  e8               inx
03835  e9be  86 13            stx track
03836  e9c0  e0 24            cpx #36
03837  e9c2  b0 06            bcs ba30         track unavailable
03838  e9c4  20 ab ea         jsr bt05
03839  e9c7  4c a3 e9         jmp ba10
03840  e9ca
03841  e9ca  85 13      ba30  sta track        track not available:
03842  e9cc  a9 65      ba35  lda #noblk
03843  e9ce  4c 5c d9         jmp cmder2
03844  e9d1
03845  e9d1  ae 73 43   ba40  ldx erword       search finished
03846  e9d4  d0 f6            bne ba35         block wasn't available
03847  e9d6  20 9f eb         jsr usedts       mark track & sector as used
03848  e9d9  4c 9f db         jmp endcmd       Terminate command successfully
03849  e9dc
03850  e9dc
03851  e9dc  ===> B-R sub to test parameters <===
03852  e9dc
03853  e9dc  20 95 ea   blkrd2 jsr bkotst       Test all block operation parameters
03854  e9df  4c 57 f0         jmp drtrd        Direct block read
03855  e9e2
03856  e9e2
03857  e9e2  ===> B-R sub to get byte without increment <===
03858  e9e2
03859  e9e2  20 b0 ed   getsim jsr getpre       get byte without increment
03860  e9e5  a1 29            lda (buftab,x)   buffer 0 pointer lo
03861  e9e7  60               rts
03862  e9e8
03863  e9e8
03864  e9e8  ===> B-R sub to do read <===
03865  e9e8
03866  e9e8  20 dc e9   blkrd3 jsr blkrd2       B-R sub to test parameters
03867  e9eb  a9 00            lda #$00
03868  e9ed  20 c1 f0         jsr setpnt       Set up pointer into active data
                                                buffer
03869  e9f0  20 e2 e9         jsr getsim       B-R sub to get byte without
                                                increment
03870  e9f3  99 bd 00         sta lstchr,y     channel last character pointer
03871  e9f6  a9 89            lda #rndrdy      for random access ready
03872  e9f8  99 98 00         sta chnrdy,y     write, read, eoi flags, channel
                                                status
03873  e9fb  60               rts
03874  e9fc
03875  e9fc
03876  e9fc  ===> Block read a sector <===
03877  e9fc
03878  e9fc  20 e8 e9   blkrd  jsr blkrd3       B-R sub to do read
03879  e9ff  20 e5 ef         jsr rngetl       read in the sector
03880  ea02  4c 9f db         jmp endcmd       Terminate command successfully
```

```
line   addr  object     source code

03881  ea05
03882  ea05
03883  ea05  ===> U1 - block read a sector (preferred alternative) <===
03884  ea05
03885  ea05  The only real difference with a B-R command is that U1 moves
03886  ea05  the last byte into the data buffer and stores $FF as
03887  ea05  the last byte read
03888  ea05
03889  ea05  20 0a e9   ublkrd jsr blkpar       Parse the block parameters
03890  ea08  20 e8 e9          jsr blkrd3       B-R sub to do read
03891  ea0b  b9 bd 00          lda lstchr,y     channel last character pointer
03892  ea0e  99 b5 00          sta chndat,y     channel data byte
03893  ea11  a9 ff            lda #$ff         last character
03894  ea13  99 bd 00          sta lstchr,y     channel last character pointer
03895  ea16  4c 9f db          jmp endcmd       Terminate command successfully
03896  ea19
03897  ea19
03898  ea19  ===> Block write of a sector <===
03899  ea19
03900  ea19  20 95 ea   blkwt  jsr bkotst       Test all block operation parameters
03901  ea1c  20 e1 f0          jsr getpnt       Get the active buffer pointer
03902  ea1f  a8                tay
03903  ea20  88                dey
03904  ea21  c9 02             cmp #$02
03905  ea23  b0 02             bcs bw10
03906  ea25  a0 01             ldy #$01
03907  ea27  a9 00      bw10   lda #$00
03908  ea29  20 c1 f0          jsr setpnt       Set up pointer into active data
                                                buffer
03909  ea2c  98                tya
03910  ea2d  20 b6 ec          jsr putbyt       Byte to active buffer of LINDEX
                                                channel
03911  ea30  8a                txa
03912  ea31  48                pha
03913  ea32  20 5b f0   bw20   jsr drtwrt       Direct block write
03914  ea35  68                pla
03915  ea36  aa                tax
03916  ea37  20 e7 ef          jsr rnget2       set channel ready status and last
                                                character
03917  ea3a  4c 9f db          jmp endcmd       Terminate command successfully
03918  ea3d
03919  ea3d  ===> U2 - block write of a sector <===
03920  ea3d
03921  ea3d  20 0a e9   ublkwt jsr blkpar       Parse the block parameters
03922  ea40  20 95 ea          jsr bkotst       Test all block operation parameters
03923  ea43  20 5b f0          jsr drtwrt       Direct block write
03924  ea46  4c 9f db          jmp endcmd       Terminate command successfully
03925  ea49
03926  ea49
03927  ea49
```

```
line   addr  object      source code

03928  ea49  ===> Block execute a sector <===
03929  ea49
03930  ea49  20 dc e9   blkexc jsr blkrd2       B-R sub to test parameters
03931  ea4c  a9 00             lda #$00         lo byte of
03932  ea4e  85 04      be05   sta temp         JMP address
03933  ea50  a6 a1             ldx jobnum       current job number
03934  ea52  bd ff f0          lda bufind,x     hi byte table of pointers to data
                                                 buffer
03935  ea55  85 05             sta t1
03936  ea57  20 5d ea          jsr be10
03937  ea5a  4c 9f db          jmp endcmd       Terminate command successfully
03938  ea5d
03939  ea5d  6c 04 00   be10   jmp (temp)       temporary work area
03940  ea60
03941  ea60
03942  ea60  ===> B-P - Set the buffer pointer <===
03943  ea60
03944  ea60  20 75 ea   blkptr jsr buftst       Test whether a buffer is allocated
03945  ea63  a5 a1             lda jobnum       buffer number
03946  ea65  0a                asl a
03947  ea66  aa                tax
03948  ea67  ad 8c 43          lda filsec+1
03949  ea6a  95 29             sta buftab,x     buffer 0 pointer lo
03950  ea6c  20 b0 ed          jsr getpre       Set buffer pointers
03951  ea6f  20 e7 ef          jsr rnget2
03952  ea72  4c 9f db          jmp endcmd       Terminate command successfully
03953  ea75
03954  ea75
03955  ea75  ===> Test whether a buffer is allocated related to SA <===
03956  ea75
03957  ea75  a6 81      buftst ldx flptr        file stream 1 pointer
03958  ea77  e6 81             inc flptr
03959  ea79  bd 8b 43          lda filsec,x     first link/sector
03960  ea7c  a8                tay
03961  ea7d  88                dey
03962  ea7e  88                dey              eliminate reserved SA 0 and 1
03963  ea7f  c0 0c             cpy #bamjob      test passes if SA 2-14
03964  ea81  90 05             bcc bt20
03965  ea83  a9 70      bt15   lda #nochnl
03966  ea85  4c c9 db          jmp cmderr       Command level error handling
03967  ea88
03968  ea88  85 16      bt20   sta sa           current secondary address
03969  ea8a  20 6e ed          jsr fndrch       Find the assigned read channel
03970  ea8d  b0 f4             bcs bt15
03971  ea8f  20 95 fa          jsr getact       Get active buffer number
03972  ea92  85 a1             sta jobnum       buffer number
03973  ea94  60                rts
03974  ea95
03975  ea95
03976  ea95
```

```
line   addr  object     source code

03977  ea95  ===> Test all block operation parameters <===
03978  ea95
03979  ea95  20 75 ea   bkotst jsr buftst       Test whether a buffer is allocated
03980  ea98  a6 81      blktst ldx flptr        test for legal block, set up dr, tr
                                                & se
03981  ea9a  bd 8b 43          lda filsec,x
03982  ea9d  29 01             and #$01         mask off default drive bit
03983  ea9f  85 12             sta drvnum       current drive number
03984  eaa1  bd 8d 43          lda filsec+2,x
03985  eaa4  85 14             sta sector       current sector number
03986  eaa6  bd 8c 43          lda filsec+1,x
03987  eaa9  85 13             sta track        current track number
03988  eaab
03989  eaab  20 6e f1   bt05   jsr tschk        Check for bad track and sector
                                                values
03990  eaae  85 19             sta r0           temporary result
03991  eab0  4c 35 da          jmp setlds       Turn on LED for current drive
03992  eab3
03993  eab3
03994  eab3  ===> Find relative file <===  Version 2.5
03995  eab3
03996  eab3  ------------------------------------------------------------
03997  eab3  inputs:                           outputs:
03998  eab3  RECL    - record# lo              SSNUM   - side sector number
03999  eab3  RECH    - record# hi              SSIND   - index into SS
04000  eab3  RS      - record size             RELPTR  - pointer into sector
04001  eab3  RECPTR  - first byte wanted from record
04002  eab3  ------------------------------------------------------------
04003  eab3
04004  eab3  20 d1 ea   fndrel jsr mulply       result=RN*RS+RP
04005  eab6  20 13 eb          jsr div254       Divide-by-254 entry point
04006  eab9  a5 23             lda accum+1      save remainder
04007  eabb  85 85             sta relptr       relative file pointer to track
04008  eabd  20 16 eb          jsr div120
04009  eac0  e6 85             inc relptr       bypass two link bytes
04010  eac2  e6 85             inc relptr
04011  eac4  a5 1e             lda result       save quotient
04012  eac6  85 83             sta ssnum        side sector number
04013  eac8  a5 23             lda accum+1      save remainder
04014  eaca  0a                asl a            calculate index into SS
04015  eacb  18                clc
04016  eacc  69 10             adc #16          skip link table
04017  eace  85 84             sta ssind        (end) pointer in side sector
04018  ead0  60                rts
04019  ead1
04020  ead1
04021  ead1  ===> Calculate a record's location in bytes <===
04022  ead1       result = RECNUM*RS+RECPTR    destroys .A and .X
04023  ead1
04024  ead1  20 7e eb   mulply jsr zerres       Zero RESULT
04025  ead4  85 25             sta accum+3      .A=0
04026  ead6  a6 15             ldx lindx        get index
04027  ead8  b5 59             lda recl,x       move lo byte of record#
```

```
line   addr  object     source code

04028  eada  85 23             sta accum+1
04029  eadc  b5 61             lda rech,x        move hi
04030  eade  85 24             sta accum+2
04031  eae0  d0 04             bne mul25         if not 0
04032  eae2  a5 23             lda accum+1
04033  eae4  f0 0b             beq mul50         if =0, adjust for record#0, first
                                                 record
04034  eae6  a5 23       mul25 lda accum+1
04035  eae8  38                sec
04036  eae9  e9 01             sbc #$01
04037  eaeb  85 23             sta accum+1
04038  eaed  b0 02             bcs mul50         if carry still set
04039  eaef  c6 24             dec accum+2
04040  eaf1  b5 71       mul50 lda rs,x          copy to temp
04041  eaf3  85 04             sta temp
04042  eaf5  46 04       mul100 lsr temp         do an addition?
04043  eaf7  90 03             bcc mul200        if carry clear, no add this time
04044  eaf9  20 92 eb          jsr addres        Add ACCUM to RESULT
04045  eafc  20 8a eb    mul200 jsr accx2        Multiply ACCUM by 2
04046  eaff  a5 04             lda temp          done?
04047  eb01  d0 f2             bne mul100        no
04048  eb03  a5 82             lda recptr        pointer to start of record, add in
                                                 last bit
04049  eb05  18                clc
04050  eb06  65 1e             adc result        add byte pointer
04051  eb08  85 1e             sta result
04052  eb0a  90 06             bcc mul400        skip no carry
04053  eb0c  e6 1f             inc result+1
04054  eb0e  d0 02             bne mul400
04055  eb10  e6 20             inc result+2
04056  eb12  60          mul400 rts
04057  eb13
04058  eb13
04059  eb13  ===> Divide-by-254 entry point <===
04060  eb13       RESULT = quotient, ACCUM+1 = remainder.  Destroys A, X
04061  eb13
04062  eb13  a9 fe       div254 lda #254         divide by 254
04063  eb15  2c                 .byte $2c
04064  eb16  a9 78       div120 lda #120         divide by 120
04065  eb18  85 04              sta temp         save divisor
04066  eb1a  a2 03              ldx #$03         swap ACCUM+1,2,3 with RESULT,1,2
04067  eb1c  b5 22       div100 lda accum,x
04068  eb1e  48                 pha
04069  eb1f  b5 1d              lda r4,x
04070  eb21  95 22              sta accum,x
04071  eb23  68                 pla
04072  eb24  95 1d              sta r4,x
04073  eb26  ca                 dex
04074  eb27  d0 f3              bne div100
04075  eb29  20 7e eb           jsr zerres       RESULT = 0
04076  eb2c  a2 00       div150 ldx #$00
04077  eb2e  b5 23       div200 lda accum+1,x    divide by 256
04078  eb30  95 22              sta accum,x      accumulator
```

```
line   addr  object     source code

04079  eb32  e8                inx
04080  eb33  e0 04              cpx #$04          done?
04081  eb35  90 f7              bcc div200        no
04082  eb37  a9 00              lda #$00          zero hi byte
04083  eb39  85 25              sta accum+3
04084  eb3b  24 04              bit temp          A DIV120?
04085  eb3d  30 09              bmi div300        no
04086  eb3f  06 22              asl accum         only divide by 128
04087  eb41  08                 php               set carry
04088  eb42  46 22              lsr accum         normalize
04089  eb44  28                 plp               restore carry
04090  eb45  20 8b eb           jsr acc200        2*(X/256)=X/128
04091  eb48  20 92 eb   div300  jsr addres        total A quotient
04092  eb4b  20 8a eb           jsr accx2         A=2*A
04093  eb4e  24 04              bit temp          A DIV120?
04094  eb50  30 03              bmi div400        no
04095  eb52  20 87 eb           jsr accx4         A=4*(2*A)=8*A
04096  eb55  a5 22      div400  lda accum         add in remainder
04097  eb57  18                 clc
04098  eb58  65 23              adc accum+1
04099  eb5a  85 23              sta accum+1
04100  eb5c  90 06              bcc div500
04101  eb5e  e6 24              inc accum+2
04102  eb60  d0 02              bne div500
04103  eb62  e6 25              inc accum+3
04104  eb64  a5 25      div500  lda accum+3       test < 256
04105  eb66  05 24              ora accum+2
04106  eb68  d0 c2              bne div150        crunch some more
04107  eb6a  a5 23              lda accum+1       is remainder < divisor?
04108  eb6c  38                 sec
04109  eb6d  e5 04              sbc temp
04110  eb6f  90 0c              bcc div700        yes
04111  eb71  e6 1e              inc result        no -- fix RESULT
04112  eb73  d0 06              bne div600
04113  eb75  e6 1f              inc result+1
04114  eb77  d0 02              bne div600
04115  eb79  e6 20              inc result+2
04116  eb7b  85 23      div600  sta accum+1       new remainder
04117  eb7d  60         div700  rts
04118  eb7e
04119  eb7e
04120  eb7e  ===> Zero RESULT <===
04121  eb7e
04122  eb7e  a9 00      zerres  lda #$00
04123  eb80  85 1e              sta result
04124  eb82  85 1f              sta result+1
04125  eb84  85 20              sta result+2
04126  eb86  60                 rts
04127  eb87
04128  eb87  20 8a eb   accx4   jsr accx2        Multiply ACCUM by 2
04129  eb8a
04130  eb8a
04131  eb8a
```

```
line   addr  object     source code

04132  eb8a  ===> Multiply ACCUM by 2 <===
04133  eb8a
04134  eb8a  18         accx2  clc
04135  eb8b  26 23      acc200 rol accum+1
04136  eb8d  26 24             rol accum+2
04137  eb8f  26 25             rol accum+3
04138  eb91  60                rts
04139  eb92
04140  eb92
04141  eb92  ===> Add ACCUM to RESULT <===          RESULT=RESULT+ACCUM+1,2,3
04142  eb92
04143  eb92  18         addres clc
04144  eb93  a2 fd             ldx #$fd
04145  eb95  b5 21      add100 lda result+3,x
04146  eb97  75 26             adc accum+4,x
04147  eb99  95 21             sta result+3,x
04148  eb9b  e8                inx
04149  eb9c  d0 f7             bne add100
04150  eb9e  60                rts
04150  eb9f
04151  eb9f                    .lib tst2
```

```
line   addr  object      source code

04153  eb9f  ===> mark track & sector as used <===
04154  eb9f
04155  eb9f  20 b4 eb   usedts jsr freuse       Calculate BAM index for FRETS and
                                                USEDTS
04156  eba2  f0 0f             beq userts       used, no action
04157  eba4  b1 02             lda (bmpnt),y    get bits
04158  eba6  5d ce eb          eor bmask,x      mark sector used
04159  eba9  91 02             sta (bmpnt),y
04160  ebab  a4 04             ldy temp         index to free sector counter
04161  ebad  b1 02             lda (bmpnt),y    get count
04162  ebaf  e9 00             sbc #$00         decrement 1 (C=0)
04163  ebb1  91 02             sta (bmpnt),y    save it
04164  ebb3  60         userts rts
04165  ebb4
04166  ebb4
04167  ebb4  ===> Calculate BAM index for FRETS and USEDTS <===
04168  ebb4
04169  ebb4  a5 13      freuse lda track        .A= track*4
04170  ebb6  0a                asl a
04171  ebb7  0a                asl a            4 bytes in BAM per track
04172  ebb8  85 04             sta temp         save index
04173  ebba  a5 14             lda sector       .A= sector/8
04174  ebbc  4a                lsr a            divide by 8 to find out which
04175  ebbd  4a                lsr a            of the three bytes for this track
04176  ebbe  4a                lsr a            the sector is in
04177  ebbf  38                sec
04178  ebc0  65 04             adc temp         calculate index
04179  ebc2  a8                tay
04180  ebc3  a5 14             lda sector
04181  ebc5  29 07             and #%111        find bit position
04182  ebc7  aa                tax
04183  ebc8  b1 02             lda (bmpnt),y    get the byte
04184  ebca  3d ce eb          and bmask,x      test it
04185  ebcd  60                rts              Z=1: used   Z=0:free
04186  ebce
04187  ebce
04188  ebce  ===> BAM mask bytes <===
04189  ebce
04190  ebce  01 02 04   bmask  .byte 1,2,4,8,16,32,64,128
04191  ebd1  08 10 20
04192  ebd4  40 80
04193  ebd6
04194  ebd6
04195  ebd6  ===> Double buffering: toggle active buffer number in BUFNUM <===
04196  ebd6
04197  ebd6  a6 15      dblbuf ldx lindx
04198  ebd8  b5 49             lda buf0,x       toggle active flag
04199  ebda  49 80             eor #$80
04200  ebdc  95 49             sta buf0,x
04201  ebde  b5 51             lda buf1,x       toggle active flag
04202  ebe0  49 80             eor #$80
04203  ebe2  95 51             sta buf1,x
04204  ebe4  20 95 fa          jsr getact       Get active buffer number
```

```
line   addr  object      source code

04205  ebe7  aa                  tax
04206  ebe8  4c 87 ec            jmp watjob        Wait until buffer ready
04207  ebeb
04208  ebeb
04209  ebeb  ===> Write byte to internal channel <===
04210  ebeb
04211  ebeb  a2 12       pibyte ldx #iwsa         SA of internal write channel
04212  ebed  86 16              stx sa             current SA
04213  ebef  20 89 ed    pbyte  jsr fndwch         find unused write channel
04214  ebf2  20 35 da           jsr setlds         Turn on LED for current drive
04215  ebf5  a5 16              lda sa
04216  ebf7  c9 0f              cmp #15            using command channel?
04217  ebf9  f0 23              beq 142            yes
04218  ebfb  d0 08              bne 140            no
04219  ebfd
04220  ebfd
04221  ebfd  ===> Main routine to write to channel <===
04222  ebfd
04223  ebfd  a5 17       put    lda orgsa          command or data channel?
04224  ebff  29 8f              and #%10001111
04225  ec01  c9 0f              cmp #15            <15 means a data channel
04226  ec03  b0 19              bcs 142
04227  ec05  20 a6 ed   140     jsr typfil         Get current file type
04228  ec08  b0 05              bcs 141            branch if random
04229  ec0a  a5 18              lda data           seq file
04230  ec0c  4c 1e ee           jmp wrtbyt         Write character to the active
                                                  channel
04231  ec0f
04232  ec0f  d0 03       141    bne 146            if Z not set, we are writing
                                                  USR file
04233  ec11  4c 94 fb           jmp wrtrel         Write out relative records
04234  ec14
04235  ec14  a5 18       146    lda data           USR file -- write byte
04236  ec16  20 b6 ec           jsr putbyt         to new channel
04237  ec19  a4 15              ldy lindx          and prepare for next byte
04238  ec1b  4c e7 ef           jmp rnget2
04239  ec1e
04240  ec1e  a9 06       142    lda #cmdchn
04241  ec20  85 15              sta lindx
04242  ec22  20 e1 f0           jsr getpnt         test if command buffer full (>40)
04243  ec25  c9 29              cmp #41
04244  ec27  f0 05              beq 150
04245  ec29  a5 18              lda data           not yet, so store
04246  ec2b  20 b6 ec           jsr putbyt         byte
04247  ec2e  a5 a0       150    lda eoiflg         test if last byte of message
04248  ec30  f0 01              beq 145
04249  ec32  60                 rts
04250  ec33
04251  ec33  ee 47 43    145    inc cmdwat         set command waiting flag
04252  ec36  60                 rts
04253  ec37
04254  ec37
04255  ec37
```

```
line   addr  object      source code

04256  ec37  ===> Test if job (.X) is done yet and redo if necessary <===
04257  ec37
04258  ec37  bd 03 10  tstjob lda jobs,x       value from job queue >127?
04259  ec3a  30 49             bmi notyet
04260  ec3c  c9 02             cmp #$02        if <2, job is completed with no
                                               errors
04261  ec3e  90 3d             bcc ok
04262  ec40  de 5d 43          dec errcnt,x    redu until error count 0
04263  ec43  10 3a             bpl again
04264  ec45  2c 9e 43          bit jobrtn      return with .A=error?
04265  ec48  30 33             bmi ok
04266  ec4a  2c 5c 43  recov   bit revcnt       error recovery count
04267  ec4d  30 29             bmi rec4        no recovery
04268  ec4f  98                tya
04269  ec50  48                pha
04270  ec51  bd 4e 43          lda lstjob,x    restore head to track 1
04271  ec54  29 01             and #$01
04272  ec56  09 c0             ora #bump
04273  ec58  9d 03 10          sta jobs,x      queue
04274  ec5b  bd 03 10  rec1    lda jobs,x
04275  ec5e  30 fb             bmi rec1
04276  ec60  ad 5c 43          lda revcnt      error recovery count
04277  ec63  29 3f             and #%111111
04278  ec65  a8                tay
04279  ec66  bd 4e 43  rec2    lda lstjob,x    set last job
04280  ec69  9d 03 10          sta jobs,x
04281  ec6c  bd 03 10  rec3    lda jobs,x      wait
04282  ec6f  30 fb             bmi rec3
04283  ec71  c9 02             cmp #$02        recovery worked if error code <2
04284  ec73  90 06             bcc rec5
04285  ec75  88                dey             it didn't work, but try a few times
04286  ec76  d0 ee             bne rec2
04287  ec78  4c 25 d9  rec4    jmp error       give up!
04288  ec7b
04289  ec7b  68        rec5    pla             it worked! restore .Y
04290  ec7c  a8                tay
04291  ec7d  18        ok      clc
04292  ec7e  60                rts             C=0
04293  ec7f
04294  ec7f  bd 4e 43  again   lda lstjob,x
04295  ec82  9d 03 10          sta jobs,x
04296  ec85  38        notyet  sec
04297  ec86  60                rts             C=1
04298  ec87
04299  ec87
04300  ec87  ===> Wait until job is completed <===
04301  ec87
04302  ec87  20 37 ec  watjob jsr tstjob       Test if job (.X) is done yet and
                                               redo if necessary
04303  ec8a  b0 fb             bcs watjob      Wait until job is completed
04304  ec8c  48                pha             clear job return flag
04305  ec8d  a9 00             lda #$00
04306  ec8f  8d 9e 43          sta jobrtn      set job completed flag
```

```
line   addr  object     source code

04307  ec92  68               pla             recover error code
04308  ec93  60               rts
04309  ec94
04310  ec94
04311  ec94  ===> Set up header for active buffer <===
04312  ec94
04313  ec94  20 95 fa   sethdr jsr getact      Get active buffer number
04314  ec97  0a         seth   asl a
04315  ec98  0a                asl a
04316  ec99  0a                asl a
04317  ec9a  a8                tay
04318  ec9b  a5 13             lda track
04319  ec9d  99 23 10          sta hdrs+2,y    set track &
04320  eca0  a5 14             lda sector
04321  eca2  99 24 10          sta hdrs+3,y    sector
04322  eca5  a5 12             lda drvnum      get proper ID
04323  eca7  0a                asl a
04324  eca8  aa                tax
04325  eca9  bd 40 43          lda dskid,x     now move ID numbers
04326  ecac  99 21 10          sta hdrs,y
04327  ecaf  bd 41 43          lda dskid+1,x
04328  ecb2  99 22 10          sta hdrs+1,y
04329  ecb5  60                rts
04330  ecb6
04331  ecb6
04332  ecb6  ===> Byte to active buffer of LINDEX channel <===
04333  ecb6
04334  ecb6  48         putbyt pha
04335  ecb7  20 95 fa          jsr getact      Get active buffer number
04336  ecba  10 06             bpl putbl       branch if there is one
04337  ecbc  68                pla             no buffer error
04338  ecbd  a9 61             lda #filnop
04339  ecbf  4c c9 db          jmp cmderr      command level error
04340  ecc2
04341  ecc2  0a         putbl  asl a           save buffer number
04342  ecc3  aa                tax
04343  ecc4  68                pla             save data byte
04344  ecc5  81 29             sta (buftab,x)  lo
04345  ecc7  f6 29             inc buftab,x    increment buffer pointer
04346  ecc9  60                rts             Z=1 if last character slot in buffer
04347  ecca
04348  ecca
04349  ecca  ===> Initialize drive(s) -- disk command <===
04350  ecca       and find active buffer number (LINDX)
04351  ecca
04352  ecca  20 d2 db   intdrv jsr simprs      parse the disk command
04353  eccd  20 ff ec          jsr initdr      initialize drives
04354  ecd0  ad 91 43          lda image       flag for both drives
04355  ecd3  10 0c             bpl id20
04356  ecd5  20 8c dd          jsr togdrv      Toggle drive number
04357  ecd8  20 35 da          jsr setlds      Turn on LED for current drive
04358  ecdb  20 ff ec          jsr initdr
04359  ecde  20 8c dd          jsr togdrv      Toggle drive number
```

```
line   addr  object      source code

04360  ece1  4c 9f db    id20    jmp endcmd        Terminate command successfully
04361  ece4
04362  ece4  a5 12       initsu  lda drvnum       current drive number
04363  ece6  18                  clc
04364  ece7  69 0c               adc #$0c
04365  ece9  85 a1               sta jobnum        current job number
04366  eceb  a2 12               ldx #18           prepare to read BAM
04367  eced  86 13               stx track
04368  ecef  a2 00               ldx #0
04369  ecf1  86 14               stx sector
04370  ecf3  20 97 ec            jsr seth          set up seek image of BAM header
04371  ecf6  a6 a1               ldx jobnum        current job number
04372  ecf8  a5 12               lda drvnum        current drive number
04373  ecfa  09 b0               ora #seek
04374  ecfc  4c 9d f1            jmp doit          Do job, set up error count and exit
                                                   if error returns
04375  ecff
04376  ecff  20 54 ef    initdr  jsr cldchn        Close all channels except the
                                                   command channel
04377  ed02  20 e4 ec            jsr initsu
04378  ed05  a9 00               lda #$00
04379  ed07  99 24 10            sta hdrs+3,y      sector
04380  ed0a  a5 12               lda drvnum        get proper ID
04381  ed0c  09 80               ora #$80
04382  ed0e  20 9d f1            jsr doit          Do job, set up error count and exit
                                                   if error returns
04383  ed11  a5 12               lda drvnum        set master ID for diskette
04384  ed13  0a                  asl a
04385  ed14  aa                  tax
04386  ed15  b9 21 10            lda hdrs,y
04387  ed18  9d 40 43            sta dskid,x
04388  ed1b  b9 22 10            lda hdrs+1,y
04389  ed1e  9d 41 43            sta dskid+1,x
04390  ed21  60                  rts
04391  ed22
04392  ed22
04393  ed22  ===> Start double buffering (reading ahead) <===
04394  ed22
04395  ed22  20 94 ec    strdbl  jsr sethdr        Set up header for active buffer
04396  ed25  20 46 ed            jsr rdbuf         read next block into data buffer
04397  ed28  20 87 ec            jsr watjob        Wait until job is completed
04398  ed2b  20 b8 ed            jsr getbyt        read track link in the active buffer
04399  ed2e  85 13               sta track
04400  ed30  20 b8 ed            jsr getbyt        read sector link in the active
                                                   buffer
04401  ed33  85 14               sta sector
04402  ed35  a5 13               lda track         if not 0, we're not at the end of
                                                   file
04403  ed37  d0 01               bne strl
04404  ed39  60                  rts
04405  ed3a
04406  ed3a  20 d6 eb    strl    jsr dblbuf        set up buffers & pointers for double
                                                   buffering
```

```
line   addr  object      source code

04407  ed3d  20 94 ec          jsr sethdr        set up header for active buffer
04408  ed40  20 46 ed          jsr rdbuf         read next block into data buffer
04409  ed43  4c d6 eb          jmp dblbuf        set up buffers and pointers
04410  ed46
04411  ed46  a9 80       rdbuf  lda #read
04412  ed48  d0 02              bne strtit
04413  ed4a
04414  ed4a  a9 90       wrtbuf lda #write
04415  ed4c  8d 3c 43    strtit sta cmd           temporary job command
04416  ed4f  20 95 fa          jsr getact        Get active buffer number
04417  ed52  aa                tax
04418  ed53  20 0e f1          jsr setljb        Set up job using last job's drive
04419  ed56  8a                txa               .A = job number,  .X = buffer
                                                 number. Transfer and
04420  ed57  48                pha               save
04421  ed58  0a                asl a             multiply by 2
04422  ed59  aa                tax               and use as
04423  ed5a  a9 00             lda #$00          index in
04424  ed5c  95 29             sta buftab,x      buffer table
04425  ed5e  20 a6 ed          jsr typfil        Get current file type
04426  ed61  c9 04             cmp #$04          sequential?
04427  ed63  b0 06             bcs wrtcl         no
04428  ed65  f6 59             inc nbkl,x        block count lo
04429  ed67  d0 02             bne wrtcl
04430  ed69  f6 61             inc nbkh,x        block count hi
04431  ed6b  68          wrtcl  pla               original buffer#
04432  ed6c  aa                tax
04433  ed6d  60                rts
04434  ed6e
04435  ed6e
04436  ed6e  ===> Find the assigned read channel <===
04437  ed6e
04438  ed6e  a5 16       fndrch lda sa            current not too high (>19)?
04439  ed70  c9 13              cmp #maxsa+1
04440  ed72  90 02              bcc fndc20
04441  ed74  29 0f              and #%1111
04442  ed76  c9 0f       fndc20 cmp #$0f          mask off high order bits -- $11=$1,
                                                 $12=$2
04443  ed78  d0 02              bne fndc25
04444  ed7a  a9 10              lda #errsa
04445  ed7c  aa          fndc25 tax               SA
04446  ed7d  38                 sec
04447  ed7e  b5 a2              lda lintab,x      channel number
04448  ed80  30 06              bmi fndc30        bit 7 set -- no channel
04449  ed82  29 0f              and #%1111
04450  ed84  85 15              sta lindx         current channel number
04451  ed86  aa                 tax
04452  ed87  18                 clc
04453  ed88  60          fndc30 rts
04454  ed89
04455  ed89
04456  ed89
```

```
line   addr  object     source code

04457  ed89  ===> Find the assigned write channel <===
04458  ed89
04459  ed89  a5 16      fndwch lda sa           current SA >19?
04460  ed8b  c9 13             cmp #maxsa+1
04461  ed8d  90 02             bcc fndwl3
04462  ed8f  29 0f             and #%1111
04463  ed91  aa        fndwl3 tax
04464  ed92  b5 a2             lda lintab,x    channel number
04465  ed94  a8                tay
04466  ed95  0a                asl a
04467  ed96  90 0a             bcc fndwl5      channel assigned  (bit 7 set)
04468  ed98  30 0a             bmi fndw20      not assigned      (7 and 6 set)
04469  ed9a  98        fndwl0 tya              original SA
04470  ed9b  29 0f             and #%1111      mask off hi
04471  ed9d  85 15             sta lindx       currently active channel
04472  ed9f  aa                tax
04473  eda0  18                clc
04474  eda1  60                rts
04475  eda2
04476  eda2  30 f6      fndwl5 bmi fndwl0      bit 6 set: inactive channel
04477  eda4  38        fndw20 sec              abort after setting the carry flag
04478  eda5  60                rts
04479  eda6
04480  eda6
04481  eda6  ===> Get current file type <===
04482  eda6
04483  eda6  a6 15      typfil ldx lindx       logical index, channel number
04484  eda8  b5 90             lda filtyp,x    file type flags, channel 0-7
04485  edaa  4a                lsr a           divide by 2, mask hi bits
04486  edab  29 07             and #%111
04487  edad  c9 04             cmp #reltyp     if relative, set Z
04488  edaf  60                rts
04489  edb0
04490  edb0
04491  edb0  ===> Set buffer pointers <===
04492  edb0
04493  edb0  20 95 fa   getpre jsr getact      Get active buffer number
04494  edb3  0a                asl a
04495  edb4  aa                tax
04496  edb5  a4 15             ldy lindx       current channel number
04497  edb7  60                rts
04497  edb8
04498  edb8                    .lib getbyt
```

```
line   addr  object     source code

04500  edb8  ===> Read one byte from the active buffer <===
04501  edb8       Set flag if last data byte, then yes: Z=1 else Z=0
04502  edb8
04503  edb8  20 b0 ed   getbyt jsr getpre       Set buffer pointers
04504  edbb  b9 bd 00          lda lstchr,y     channel last character pointer
04505  edbe  f0 12             beq getb1
04506  edc0  a1 29             lda (buftab,x)   data byte to stack
04507  edc2  48                pha
04508  edc3  b5 29             lda buftab,x     compare pointer to
04509  edc5  d9 bd 00          cmp lstchr,y     character read
04510  edc8  d0 04             bne getb2
04511  edca  a9 ff             lda #$ff
04512  edcc  95 29             sta buftab,x
04513  edce  68          getb2  pla              if last byte, set Z
04514  edcf  f6 29             inc buftab,x
04515  edd1  60                rts
04516  edd2  a1 29       getb1  lda (buftab,x)
04517  edd4  f6 29             inc buftab,x
04518  edd6  60                rts
04519  edd7
04520  edd7
04521  edd7  ===> Read byte from file <===
04522  edd7       and read next block of file if needed
04523  edd7
04524  edd7  20 b8 ed   rdbyt  jsr getbyt       Read one byte from the active buffer
04525  edda  d0 36             bne rd3
04526  eddc  85 18             sta data
04527  edde  b9 bd 00          lda lstchr,y     channel last character pointer
04528  ede1  f0 08             beq rd1
04529  ede3  a9 80             lda #eoiout      last byte read
04530  ede5  99 98 00          sta chnrdy,y     channel status
04531  ede8  a5 18             lda data
04532  edea  60                rts
04533  edeb
04534  edeb  20 d6 eb   rd1    jsr dblbuf       Double buffer: switch
                                                active/inactive buffers
04535  edee  a9 00             lda #$00
04536  edf0  20 c1 f0          jsr setpnt       Set up pointer into active data
                                                buffer
04537  edf3  20 b8 ed          jsr getbyt       Read one byte from the active buffer
04538  edf6  c9 00             cmp #$00
04539  edf8  f0 19             beq rd4          no next block
04540  edfa  85 13             sta track
04541  edfc  20 b8 ed          jsr getbyt       Read one byte from the active buffer
04542  edff  85 14             sta sector
04543  ee01  20 d6 eb          jsr dblbuf
04544  ee04  20 54 ee          jsr setdrn       Set drive number
04545  ee07  20 94 ec          jsr sethdr       Set up header for active buffer
04546  ee0a  20 46 ed          jsr rdbuf        read in next block
04547  ee0d  20 d6 eb          jsr dblbuf       toggle buffers
04548  ee10  a5 18             lda data
04549  ee12  60          rd3    rts
04550  ee13
```

```
line   addr  object      source code

04551  ee13  20 b8 ed   rd4    jsr getbyt       Read one byte from the active buffer
04552  ee16  a4 15             ldy lindx        current channel#
04553  ee18  99 bd 00           sta lstchr,y     channel last character pointer
04554  ee1b  a5 18              lda data
04555  ee1d  60                 rts
04556  ee1e
04557  ee1e
04558  ee1e  ===> Write character to the active channel <===
04559  ee1e       If this fills the buffer, write buffer to disk
04560  ee1e
04561  ee1e  20 b6 ec   wrtbyt jsr putbyt       Byte to active buffer of LINDEX
                                                channel
04562  ee21  f0 01              beq wrt0         buffer full
04563  ee23  60                 rts
04564  ee24
04565  ee24  20 54 ee   wrt0   jsr setdrn       Set drive number
04566  ee27  20 b0 d6           jsr nxtts        Find the next available track and
                                                sector
04567  ee2a  a9 00              lda #$00
04568  ee2c  20 c1 f0           jsr setpnt       Set up pointer into active data
                                                buffer
04569  ee2f  a5 13              lda track
04570  ee31  20 b6 ec           jsr putbyt       store links
04571  ee34  a5 14              lda sector
04572  ee36  20 b6 ec           jsr putbyt
04573  ee39  20 4a ed           jsr wrtbuf       write out buffer
04574  ee3c  20 d6 eb           jsr dblbuf       toggle
04575  ee3f  20 94 ec           jsr sethdr       Set up header for active buffer
04576  ee42  a9 02              lda #$02         bypass track/sector link
04577  ee44  4c c1 f0           jmp setpnt       Set up pointer into active data
                                                buffer
04578  ee47
04579  ee47
04580  ee47  ===> Increment the pointer of the active buffer by .A <===
04581  ee47
04582  ee47            incpnt
04583  ee47  85 04     incptr sta temp
04584  ee49  20 e1 f0           jsr getpnt       Get the active buffer pointer
04585  ee4c  18                 clc
04586  ee4d  65 04              adc temp
04587  ee4f  95 29              sta buftab,x     lo
04588  ee51  85 27              sta dirbuf       directory buffer pointer
04589  ee53  60                 rts
04590  ee54
04591  ee54
04592  ee54  ===> Set drive number same as last job <===
04593  ee54
04594  ee54  20 95 fa   setdrn jsr getact       Get active buffer number
04595  ee57  aa                 tax
04596  ee58  bd 4e 43           lda lstjob,x     last job by buffer
04597  ee5b  29 01              and #$01
04598  ee5d  85 12              sta drvnum       current drive number
04599  ee5f  60                 rts
```

```
line   addr  object      source code

04600  ee60
04601  ee60
04602  ee60  ===> Open a new write channel <===
04603  ee60
04604  ee60  38          getwch sec              we want a write channel: C=1
04605  ee61  b0 01             bcs getr2
04606  ee63
04607  ee63
04608  ee63  ===> Open a new read channel <===
04609  ee63
04610  ee63  18          getrch clc              C=0 for read
04611  ee64  08          getr2  php              save R/W flag
04612  ee65  85 04             sta temp         save buffers we need
04613  ee67  20 a4 ee          jsr frechn       Free channel associated with SA
04614  ee6a  20 79 ef          jsr fndlnx       find next free channel, allocate it
04615  ee6d  85 15             sta lindx        use as current channel#
04616  ee6f  a6 16             ldx sa           current SA
04617  ee71  28                plp              check R/W
04618  ee72  90 02             bcc getr55       read
04619  ee74  09 80             ora #%10000000   write
04620  ee76  95 a2       getr55 sta lintab,x     bit 7 set for a write channel
04621  ee78  29 3f             and #%00111111   mask off write channel bit
04622  ee7a  a8                tay
04623  ee7b  a9 ff             lda #$ff         deallocate associated buffers
04624  ee7d  99 49 00          sta buf0,y
04625  ee80  99 51 00          sta buf1,y
04626  ee83  c6 04             dec temp         number of buffers to allocate
04627  ee85  30 1c             bmi getr4
04628  ee87  20 03 ef          jsr getbuf       Get a free buffer
04629  ee8a  10 08             bpl getr5        no more
04630  ee8c  20 cf ee    gberr  jsr relbuf       no buffers available!
04631  ee8f  a9 70             lda #nochnl
04632  ee91  4c c9 db          jmp cmderr       Command level error handling
04633  ee94
04634  ee94  99 49 00    getr5  sta buf0,y       channel buffer table 1
04635  ee97  c6 04             dec temp
04636  ee99  30 08             bmi getr4
04637  ee9b  20 03 ef          jsr getbuf       Get a free buffer
04638  ee9e  30 ec             bmi gberr
04639  eea0  99 51 00          sta buf1,y       channel buffer table 2
04640  eea3  60          getr4  rts
04641  eea4
04642  eea4
04643  eea4  ===> Free channel associated with SA <===
04644  eea4
04645  eea4  a5 16       frechn lda sa           current SA
04646  eea6  c9 0f             cmp #15          command channel
04647  eea8  d0 01             bne freco        Free data channel associated with SA
04648  eeaa  60                rts              must not free command channel!
04649  eeab
04650  eeab
```

```
line   addr  object     source code

04651  eeab  ===> Free data channel associated with SA <===
04652  eeab
04653  eeab  a6 16      freco  ldx sa           current SA
04654  eead  b5 a2             lda lintab,x     current status SA
04655  eeaf  c9 ff             cmp #$ff
04656  eeb1  f0 1b             beq fre25
04657  eeb3  29 3f             and #%00111111   mask off hi bits
04658  eeb5  85 15             sta lindx        channel#
04659  eeb7  a9 ff             lda #$ff         free it
04660  eeb9  95 a2             sta lintab,x
04661  eebb  20 cf ee          jsr relbuf       Release buffers associated with
                                                channel
04662  eebe  a6 15             ldx lindx        channel#
04663  eec0  a9 01             lda #$01
04664  eec2  ca        rel15  dex
04665  eec3  30 03             bmi rel10        no lower channel numbers
04666  eec5  0a                asl a            shift a bit
04667  eec6  d0 fa             bne rel15        .A<>0 (always)
04668  eec8  0d 48 43  rel10  ora linuse        free channel (1=free)
04669  eecb  8d 48 43          sta linuse       LINDX use word
04670  eece  60        fre25  rts
04671  eecf
04672  eecf
04673  eecf  ===> Release buffers associated with channel <===
04674  eecf
04675  eecf  a6 15      relbuf ldx lindx        logical index, channel#
04676  eed1  b5 49             lda buf0,x       channel buffer table 1
04677  eed3  c9 ff             cmp #$ff         free?
04678  eed5  f0 09             beq rel1
04679  eed7  48                pha              save it
04680  eed8  a9 ff             lda #$ff         free it
04681  eeda  95 49             sta buf0,x
04682  eedc  68                pla              buffer#
04683  eedd  20 34 ef          jsr frebuf       Free buffer in BUFUSE
04684  eee0  a6 15      rel1   ldx lindx        channel#
04685  eee2  b5 51             lda buf1,x       buffer#
04686  eee4  c9 ff             cmp #$ff         free?
04687  eee6  f0 09             beq rel2
04688  eee8  48                pha              save it
04689  eee9  a9 ff             lda #$ff         free the
04690  eeeb  95 51             sta buf1,x       buffer
04691  eeed  68                pla              buffer#
04692  eeee  20 34 ef          jsr frebuf       free it
04693  eef1  a6 15      rel2   ldx lindx        channel#
04694  eef3  b5 79             lda ss,x         side sector for this channel
04695  eef5  c9 ff             cmp #$ff         free?
04696  eef7  f0 09             beq rel3
04697  eef9  48                pha              save side sector
04698  eefa  a9 ff             lda #$ff         free pointer in
04699  eefc  95 79             sta ss,x         side sectors table
04700  eefe  68                pla              side sector
04701  eeff  20 34 ef          jsr frebuf       free buffers in BUFUSE
04702  ef02  60        rel3   rts
```

```
line   addr  object      source code

04703  ef03
04704  ef03
04705  ef03  ===> Get a free buffer <===
04706  ef03
04707  ef03  a9 ff       getbuf lda #$ff
04708  ef05  85 05              sta t1
04709  ef07  a2 0f              ldx #15          buffer count
04710  ef09  2e 3e 43    getbu1 rol bufuse       find which buffer in use
04711  ef0c  2e 3f 43           rol bufuse+1
04712  ef0f  b0 05              bcs getbu2       found one
04713  ef11  86 05              stx t1
04714  ef13  38                 sec
04715  ef14  b0 13              bcs getbu3
04716  ef16  ca          getbu2 dex              try the next one
04717  ef17  10 f0              bpl getbu1
04718  ef19  a6 05       getbu4 ldx t1
04719  ef1b  30 0a              bmi getb5
04720  ef1d  a9 00              lda #$00
04721  ef1f  9d 03 10           sta jobs,x       queue
04722  ef22  a5 12              lda drvnum       current drive number
04723  ef24  9d 4e 43           sta lstjob,x     last job by buffer
04724  ef27  8a          getb5  txa
04725  ef28  60                 rts
04726  ef29
04727  ef29  2e 3e 43    getbu3 rol bufuse       buffer allocation
04728  ef2c  2e 3f 43           rol bufuse+1
04729  ef2f  ca                 dex
04730  ef30  10 f7              bpl getbu3
04731  ef32  30 e5              bmi getbu4
04732  ef34
04733  ef34
04734  ef34  ===> Free buffer in BUFUSE <===
04735  ef34
04736  ef34  29 0f       frebuf and #$0f          mask off hi bits
04737  ef36  a8                 tay
04738  ef37  c8                 iny
04739  ef38  a2 10              ldx #16          2*8 bits -- loop 16 times
04740  ef3a  6e 3f 43    freb1  ror bufuse+1
04741  ef3d  6e 3e 43           ror bufuse
04742  ef40  88                 dey              to count down to 0. When .Y=0, the
                                                  bit that
04743  ef41  d0 01              bne freb2        corresponds to the buffer we want is
                                                  in the
04744  ef43  18                 clc              carry flag, so we clear the carry to
                                                  free
04745  ef44  ca          freb2  dex              that buffer. When .X reaches $FF,
                                                  the bits
04746  ef45  10 f3              bpl freb1        are all back in the right places
04747  ef47  60                 rts
04748  ef48
04749  ef48
04750  ef48
```

```
line   addr  object      source code

04751  ef48  ===> Clear all channels except the command channel <===
04752  ef48
04753  ef48  a9 0e       clrchn lda #14
04754  ef4a  85 16              sta sa           current SA
04755  ef4c  20 a4 ee    clrcl  jsr frechn       Free channel associated with SA
04756  ef4f  c6 16              dec sa           current SA
04757  ef51  d0 f9              bne clrcl
04758  ef53  60                 rts
04759  ef54
04760  ef54
04761  ef54  ===> Close all channels except the command channel <===
04762  ef54
04763  ef54  a9 14       cldchn lda #$14
04764  ef56  85 16              sta sa           current SA
04765  ef58  a6 16       clsd   ldx sa           use as index
04766  ef5a  b5 a2              lda lintab,x     current status SA
04767  ef5c  c9 ff              cmp #$ff         none assigned
04768  ef5e  f0 14              beq cld2
04769  ef60  29 3f              and #%00111111   mask off hi bits
04770  ef62  85 15              sta lindx        current channel#
04771  ef64  20 95 fa           jsr getact       Get active buffer number
04772  ef67  aa                 tax
04773  ef68  bd 4e 43           lda lstjob,x     last job by buffer
04774  ef6b  29 01              and #$01
04775  ef6d  c5 12              cmp drvnum       current drive number
04776  ef6f  d0 03              bne cld2
04777  ef71  20 a4 ee           jsr frechn       Free channel associated with SA
04778  ef74  c6 16       cld2   dec sa           current SA
04779  ef76  10 e0              bpl clsd
04780  ef78  60                 rts
04781  ef79
04782  ef79  a0 00       fndlnx ldy #$00
04783  ef7b  a9 01              lda #$01
04784  ef7d  2c 48 43    fnd10  bit linuse       test if same bit set in LINUSE and
                                                 .A
04785  ef80  d0 09              bne fnd30        when set, corresponding channel is
                                                 free
04786  ef82  c8                 iny              counter
04787  ef83  0a                 asl a            test next bit on the left
04788  ef84  d0 f7              bne fnd10        more testing needed
04789  ef86  a9 70              lda #nochnl
04790  ef88  4c c9 db           jmp cmderr       Command level error handling
04791  ef8b
04792  ef8b  49 ff       fnd30  eor #$ff         toggle bit mask
04793  ef8d  2d 48 43           and linuse       mark bit used
04794  ef90  8d 48 43           sta linuse
04795  ef93  98                 tya
04796  ef94  60                 rts
04797  ef95
04798  ef95
```

```
line   addr  object     source code

04799  ef95  ===> Get next byte from channel <===
04800  ef95
04801  ef95  20 6e ed   gbyte  jsr fndrch       Find an unused read channel
04802  ef98  20 35 da          jsr setlds       Turn on LED for current drive
04803  ef9b  20 a3 ef          jsr get          Get next byte from any type of file
04804  ef9e  a6 15              ldx lindx        current channel#
04805  efa0  b5 b5              lda chndat,x     channel data byte
04806  efa2  60                 rts
04807  efa3
04808  efa3
04809  efa3  ===> Get next byte from any type of file <===
04810  efa3
04811  efa3  a6 15       get    ldx lindx        logical index, channel#
04812  efa5  20 a6 ed           jsr typfil       Get current file type
04813  efa8  d0 03              bne get00
04814  efaa  4c 01 fc           jmp rdrel        Read relative records
04815  efad
04816  efad  a5 16       get00  lda sa           current SA
04817  efaf  c9 0f              cmp #$0f
04818  efb1  f0 59              beq geterc       Get byte from error channel
04819  efb3  b5 98              lda chnrdy,x     was last character just sent?
04820  efb5  29 08              and #eoisnd      just sent EOI
04821  efb7  d0 13              bne get1         no, not this time
04822  efb9  20 a6 ed           jsr typfil       Get current file type
04823  efbc  c9 07              cmp #dirtyp
04824  efbe  d0 07              bne get0         not direct type
04825  efc0  a9 89              lda #rndrdy      direct files stays active
04826  efc2  95 98              sta chnrdy,x     talker listener no EOI
04827  efc4  4c d7 ef           jmp rndget       prepare next character
04828  efc7
04829  efc7  a9 00       get0   lda #notrdy      last character sent, not ready
04830  efc9  95 98              sta chnrdy,x
04831  efcb  60                 rts
04832  efcc
04833  efcc  a5 16       get1   lda sa           test if a load
04834  efce  f0 31              beq get6
04835  efd0  20 a6 ed           jsr typfil       test for random file
04836  efd3  c9 04              cmp #$04
04837  efd5  90 22              bcc seqget
04838  efd7  20 b0 ed   rndget  jsr getpre       Set buffer pointers
04839  efda  b5 29              lda buftab,x     data byte pointer
04840  efdc  d9 bd 00           cmp lstchr,y     last character?
04841  efdf  d0 04              bne rnget1
04842  efe1  a9 00              lda #$00         yes, so wrap pointer around to start
                                                 again
04843  efe3  95 29              sta buftab,x
04844  efe5  f6 29       rnget1 inc buftab,x     point to next character
04845  efe7  a1 29       rnget2 lda (buftab,x)   data byte to
04846  efe9  99 b5 00           sta chndat,y     channel data byte
04847  efec  b5 29              lda buftab,x     is this the
04848  efee  d9 bd 00           cmp lstchr,y     last character we're supposed to
                                                 get?
04849  eff1  d0 05              bne rnget3
```

```
line   addr  object     source code

04850  eff3  a9 81            lda #rndeoi      yes,
04851  eff5  99 98 00         sta chnrdy,y     send EOI with it
04852  eff8  60         rnget3 rts
04853  eff9
04854  eff9  20 d7 ed   seqget jsr rdbyt       Read byte from file
04855  effc  a6 15      get3   ldx lindx       channel#
04856  effe  95 b5            sta chndat,x     channel data byte
04857  f000  60               rts
04858  f001
04859  f001  ad 46 43   get6   lda dirlst      looks like a load. Or a directory
                                               listing?
04860  f004  f0 f3            beq seqget       no
04861  f006  20 1a db         jsr getdir       Get character for directory loading
04862  f009  4c fc ef         jmp get3
04863  f00c
04864  f00c
04865  f00c  ===> Get byte from error channel <===
04866  f00c
04867  f00c  20 e1 f0   geterc jsr getpnt       Get the active buffer pointer
04868  f00f  c9 db            cmp #<errbuf-1
04869  f011  d0 18            bne ge10
04870  f013  a5 28            lda dirbuf+1     current buffer pointer hi
04871  f015  c9 43            cmp #>errbuf
04872  f017  d0 12            bne ge10
04873  f019  a9 0d            lda #cr          carriage return
04874  f01b  85 18            sta data
04875  f01d  20 4b da         jsr erroff       turn off error led
04876  f020  a9 00            lda #$00
04877  f022  20 d7 d9         jsr errts0       transfer message to error buffer
04878  f025  c6 47            dec cb+2
04879  f027  a9 80            lda #eoiout
04880  f029  d0 12            bne ge30         jump
04881  f02b  20 b8 ed   ge10   jsr getbyt       Read one byte from the active buffer
04882  f02e  85 18            sta data
04883  f030  d0 09            bne ge20
04884  f032  a9 db      ge15   lda #<errbuf-1
04885  f034  20 c1 f0         jsr setpnt       Set up pointer into active data
                                               buffer
04886  f037  a9 43            lda #>errbuf
04887  f039  95 2a            sta buftab+1,x   hi
04888  f03b  a9 88      ge20   lda #rdytlk
04889  f03d  85 9f      ge30   sta chnrdy+errchn
04890  f03f  a5 18            lda data
04891  f041  85 bc            sta chndat+errchn
04892  f043  60               rts
04893  f044
04894  f044  follow track/sector links, set End-Of-File when track link is zero
04895  f044
04896  f044  20 95 fa   nxtbuf jsr getact       Get active buffer number
04897  f047  0a               asl a
04898  f048  aa               tax
04899  f049  a9 00            lda #$00
04900  f04b  95 29            sta buftab,x     lo
```

```
line   addr  object     source code

04901  f04d  a1 29             lda (buftab,x)   track link zero?
04902  f04f  f0 05             beq nxtbl        no more blocks
04903  f051  d6 29             dec buftab,x     $FF, force read of next sector
04904  f053  4c d7 ed          jmp rdbyt        Read byte from file
04905  f056
04906  f056  60          nxtbl  rts
04907  f057
04908  f057
04909  f057  ===> Direct block read <===
04910  f057
04911  f057  a9 80       drtrd  lda #read
04912  f059  d0 02              bne drt
04913  f05b
04914  f05b
04915  f05b  ===> Direct block write <===
04916  f05b
04917  f05b  a9 90       drtwrt lda #write
04918  f05d  05 12       drt    ora drvnum       current drive number
04919  f05f  8d 3c 43           sta cmd          temporary job command
04920  f062  a5 a1              lda jobnum       current job number
04921  f064  20 97 ec           jsr seth
04922  f067  a6 a1              ldx jobnum       current job number
04923  f069  4c a0 f1           jmp doit2
04924  f06c
04925  f06c
04926  f06c  ===> Open internal read channel (SA=17) <===
04927  f06c
04928  f06c  a9 11       opnird lda #irsa
04929  f06e  85 16              sta sa           current SA
04930  f070  a9 01              lda #$01         prg
04931  f072  85 c5       opntyp sta type          --- entry point for any file type

04932  f074  20 47 f7           jsr opnrch       Open a read channel with two buffers
04933  f077  a9 02              lda #$02         point past tr/sec link
04934  f079  4c c1 f0           jmp setpnt       Set up pointer into active data
                                                  buffer
04935  f07c
04936  f07c
04937  f07c  ===> Open internal write channel (SA=18) <===
04938  f07c
04939  f07c  a9 12       opnirw lda #iwsa
04940  f07e  85 16              sta sa           current SA
04941  f080  4c e6 f7           jmp opnwch       Open a write channel with two
                                                  buffers
04942  f083
04943  f083
04944  f083  ===> Allocate next directory block <===
04945  f083
04946  f083  20 3b f9    nxdrbk jsr curblk       Read track & sector from header
04947  f086  a9 01              lda #$01
04948  f088  85 04              sta temp
04949  f08a  ad 44 43           lda secinc       sector increment for SEQ routine
04950  f08d  48                 pha
```

```
line   addr  object     source code

04951  f08e  a9 03            lda #3          increment for directory
04952  f090  8d 44 43         sta secinc      sector increment for SEQ routine
04953  f093  20 b7 d6         jsr nxtds       determine next available
                                               track/sector
04954  f096  68               pla
04955  f097  8d 44 43         sta secinc
04956  f09a  a9 00            lda #$00
04957  f09c  20 c1 f0         jsr setpnt      Set up pointer into active data
                                               buffer
04958  f09f  a5 13            lda track
04959  f0a1  20 b6 ec         jsr putbyt      store links in buffer
04960  f0a4  a5 14            lda sector
04961  f0a6  20 b6 ec         jsr putbyt
04962  f0a9  20 4a ed         jsr wrtbuf      write buffer out
04963  f0ac  20 87 ec         jsr watjob      Wait until job is completed
04964  f0af  a9 00            lda #$00        point to track byte
04965  f0b1  20 c1 f0         jsr setpnt
04966  f0b4  20 b6 ec  nxdbl  jsr putbyt      zero the buffer
04967  f0b7  d0 fb            bne nxdbl
04968  f0b9  20 b6 ec         jsr putbyt      store 0 as next track link
04969  f0bc  a9 ff            lda #$ff
04970  f0be  4c b6 ec         jmp putbyt      and $FF as sector link
04971  f0c1
04972  f0c1
04973  f0c1  ===> Set up pointer into active data buffer <===
04974  f0c1
04975  f0c1  85 04      setpnt sta temp
04976  f0c3  20 95 fa         jsr getact      Get active buffer number
04977  f0c6  0a               asl a
04978  f0c7  aa               tax
04979  f0c8  b5 2a            lda buftab+1,x  move hi byte of buffer pointer
04980  f0ca  85 28            sta dirbuf+1
04981  f0cc  a5 04            lda temp        store new buffer pointer
04982  f0ce  95 29            sta buftab,x
04983  f0d0  85 27            sta dirbuf
04984  f0d2  60               rts
04985  f0d3
04986  f0d3
04987  f0d3  ===> Free both internal channels <===
04988  f0d3
04989  f0d3  a9 11      freich lda #irsa
04990  f0d5  85 16            sta sa          current SA
04991  f0d7  20 a4 ee         jsr frechn      free internal read channel
04992  f0da  a9 12            lda #iwsa
04993  f0dc  85 16            sta sa          current SA
04994  f0de  4c a4 ee         jmp frechn      free internal write channel
04995  f0e1
04996  f0e1  ===> Get the active buffer pointer <===
04997  f0e1
04998  f0e1  20 95 fa   getpnt jsr getact      Get active buffer number
04999  f0e4             setdir
05000  f0e4  0a         gpl    asl a
05001  f0e5  aa               tax
```

```
line   addr  object     source code

05002  f0e6  b5 2a             lda buftab+1,x   hi
05003  f0e8  85 28             sta dirbuf+1     current buffer pointer hi
05004  f0ea  b5 29             lda buftab,x     lo
05005  f0ec  85 27             sta dirbuf       directory buffer pointer
05006  f0ee  60                 rts
05007  f0ef
05008  f0ef
05009  f0ef  ===> Direct read of a byte (.A = position) <===
05010  f0ef
05011  f0ef  85 06       drdbyt sta t2
05012  f0f1  20 95 fa           jsr getact       Get active buffer number
05013  f0f4  aa                 tax
05014  f0f5  bd ff f0           lda bufind,x     hi byte table of pointers to data
                                                 buffer
05015  f0f8  85 07             sta t3           create pointer to (T2)
05016  f0fa  a0 00             ldy #$00
05017  f0fc  b1 06             lda (t2),y       byte we want
05018  f0fe  60                 rts
05019  f0ff
05020  f0ff
05021  f0ff  ===> hi byte table of pointers to data buffer <===
05022  f0ff
05023  f0ff  11 12 13   bufind .byte $11, $12, $13
05024  f102  20 21 22           .byte $20, $21, $22, $23
05025  f105  23
05026  f106  30 31 32           .byte $30, $31, $32, $33
05027  f109  33
05028  f10a  40 41 42           .byte $40, $41, $42, $43
05029  f10d  43
05029  f10e
05030  f10e                     .lib jobs
```

```
line   addr  object     source code

05032  f10e  ===> Set up job using last job's drive, job code in CMD <===
05033  f10e
05034  f10e  bd 4e 43   setljb lda lstjob,x     last job by buffer
05035  f111  29 01             and #$01         leaves just the drive number
05036  f113  0d 3c 43          ora cmd          temporary job command
05037  f116
05038  f116
05039  f116  ===> Set up new job <===
05040  f116
05041  f116  48         setjob pha
05042  f117  86 a1             stx jobnum       current job number
05043  f119  8a                txa              transfer buffer number
05044  f11a  0a                asl a            multiply
05045  f11b  0a                asl a            by
05046  f11c  0a                asl a            8
05047  f11d  aa                tax
05048  f11e  bd 24 10          lda hdrs+3,x     move desired sector
05049  f121  8d 3c 43          sta cmd          temporary job command
05050  f124  bd 23 10          lda hdrs+2,x     then load the track
05051  f127  f0 2c             beq tserr
05052  f129  c5 24             cmp maxtrk       highest possible track number is 36
05053  f12b  b0 28             bcs tserr
05054  f12d  aa                tax              track number
05055  f12e  68                pla              check for write
05056  f12f  48                pha
05057  f130  29 f0             and #$f0         mask off drive bits
05058  f132  c9 90             cmp #write       is it a job code for write?
05059  f134  d0 52             bne sjb1
05060  f136  68                pla              job code
05061  f137  48                pha
05062  f138  4a                lsr a            find drive to use
05063  f139  b0 05             bcs sjb2         if not drive #1,
05064  f13b  ad 02 41          lda bam0+2       use drive #0. Load DOS version
05065  f13e  90 03             bcc sjb3
05066  f140  ad 02 42   sjb2   lda bam1+2       use drive #1. Load DOS version
05067  f143  f0 05      sjb3   beq sjb4         if 00 (no number) it's OK
05068  f145
05069  f145  ********************************************************************
05070  f145  Since the DOS version code - normally 65 (a) - is stored in RAM on
05071  f145  the 4040 and 8x50, it can be soft-set by the user. On the 1541,
05072  f145  however, this is impossible because the code is stored in ROM.
05073  f145  A DOS version of $00 is OK.
05074  f145  ********************************************************************
05075  f145
05076  f145  cd 9f 10           cmp vernum       "a" - DOS version number
05077  f148  d0 36              bne vnerr        Version error
05078  f14a  8a         sjb4    txa              track number
05079  f14b  20 db d7           jsr maxsec       Tell how many sectors allowed for
                                                 this track
05080  f14e  cd 3c 43           cmp cmd          temporary job command
05081  f151  f0 02              beq tserr
05082  f153  b0 33              bcs sjb1
05083  f155  20 5d f1   tserr   jsr hed2ts       Set desired track and sector values
```

```
line   addr  object     source code

05084  f158  a9 66      tser1  lda #badts
05085  f15a  4c 5c d9           jmp cmder2
05086  f15d
05087  f15d
05088  f15d  ===> Set desired track and sector values <===
05089  f15d
05090  f15d  a5 a1      hed2ts lda jobnum       current job number
05091  f15f  0a                asl a           multiply
05092  f160  0a                asl a           by
05093  f161  0a                asl a           8
05094  f162  aa                tax
05095  f163  bd 23 10          lda hdrs+2,x
05096  f166  85 13             sta track       current track number
05097  f168  bd 24 10          lda hdrs+3,x
05098  f16b  85 14             sta sector      current sector number
05099  f16d  60                rts
05100  f16e
05101  f16e
05102  f16e  ===> Check for bad track and sector values <===
05103  f16e
05104  f16e  a5 13      tschk  lda track       current track number
05105  f170  f0 e6             beq tser1
05106  f172  c5 24             cmp maxtrk      36 is tops
05107  f174  b0 e2             bcs tser1
05108  f176  20 db d7          jsr maxsec      Tell how many sectors allowed for
                                               this track
05109  f179  c5 14             cmp sector      current sector number
05110  f17b  f0 db             beq tser1
05111  f17d  90 d9             bcc tser1
05112  f17f  60                rts
05113  f180
05114  f180
05115  f180  ===> Version error <===
05116  f180
05117  f180  20 5d f1   vnerr  jsr hed2ts      Set desired track and sector values
05118  f183  a9 73             lda #cbmv2      version error
05119  f185  4c 5c d9          jmp cmder2
05120  f188
05121  f188
05122  f188  ===> Conclude job set up <===
05123  f188
05124  f188  a6 a1      sjb1   ldx jobnum       current job number
05125  f18a  ad 5c 43          lda revcnt      error recovery count
05126  f18d  29 1f             and #$1f
05127  f18f  9d 5d 43          sta errcnt,x    sector of directory entry by buffer
05128  f192  68                pla
05129  f193  8d 3c 43          sta cmd         temporary job command
05130  f196  9d 03 10          sta jobs,x      queue
05131  f199  9d 4e 43          sta lstjob,x    last job by buffer
05132  f19c  60                rts
05133  f19d
05134  f19d
05135  f19d
```

```
line   addr  object      source code

05136  f19d  ===> Do job, set up error count and exit if error returns <===
05137  f19d
05138  f19d  8d 3c 43   doit   sta cmd          temporary job command
05139  f1a0  ad 3c 43   doit2  lda cmd          temporary job command
05140  f1a3  20 16 f1          jsr setjob       Set up new job
05141  f1a6  4c 87 ec          jmp watjob       Wait until job is completed
05142  f1a9
05143  f1a9
05144  f1a9  ===> Add new filename to directory <===
05145  f1a9
05146  f1a9  a5 16      addfil lda sa           current secondary address
05147  f1ab  48                pha
05148  f1ac  a5 15              lda lindx        channel
05149  f1ae  48                pha
05150  f1af  a5 14              lda sector       sector
05151  f1b1  48                pha
05152  f1b2  a5 13              lda track        track
05153  f1b4  48                pha              save these to stack
05154  f1b5  a9 11              lda #irsa        17 = internal read channel
05155  f1b7  85 16              sta sa           current secondary address
05156  f1b9  20 3b f9           jsr curblk       Read track & sector from header
05157  f1bc  a5 c5              lda type         current file type
05158  f1be  48                pha
05159  f1bf  a5 8b              lda fildat       drive in table
05160  f1c1  29 01              and #$01
05161  f1c3  85 12              sta drvnum       current drive number
05162  f1c5  a6 a1              ldx jobnum       current job number
05163  f1c7  5d 4e 43           eor lstjob,x     same drive?
05164  f1ca  4a                 lsr a            divide by 2
05165  f1cb  90 0c              bcc af08         if clear, same drive as last job

05166  f1cd  a2 01              ldx #$01         we're searching for a deleted entry
05167  f1cf  8e 98 43           stx delind       pointer in directory for first
                                                 available entry
05168  f1d2  20 da df           jsr srchst       Initiate search of directory
05169  f1d5  f0 1d              beq af15         entry found? no, start a new sector
05170  f1d7  d0 28              bne af20         found a spot, so find pointers to
                                                 first character
05171  f1d9  ad 97 43   af08   lda delsec       sector of first available entry
05172  f1dc  f0 0c              beq af10         0? no deleted entry found at last
                                                 read
05173  f1de  c5 14              cmp sector       is this sector already in memory?
05174  f1e0  f0 1f              beq af20         compare sector numbers. If equal,
                                                 get pointers
05175  f1e2  85 14              sta sector       current sector number
05176  f1e4  20 57 f0           jsr drtrd        Direct block read
05177  f1e7  4c 01 f2           jmp af20
05178  f1ea
05179  f1ea  a9 01      af10    lda #$01         we're looking for a deleted entry
05180  f1ec  8d 98 43           sta delind       index of first available entry
05181  f1ef  20 43 e0           jsr search       Continue search of entries
05182  f1f2  d0 0d              bne af20         found?
05183  f1f4  20 83 f0   af15    jsr nxdrbk       Allocate next directory block
```

```
line   addr  object      source code

05184  f1f7  a5 14             lda sector      current sector number
05185  f1f9  8d 97 43          sta delsec      sector of first available entry
05186  f1fc  a9 02             lda #$02
05187  f1fe  8d 98 43          sta delind      pointer in dir block
05188  f201  ad 98 43   af20   lda delind      index of first available entry
05189  f204  20 c1 f0          jsr setpnt      Set up pointer into active data
                                                buffer
05190  f207  68                pla
05191  f208  85 c5             sta type        current file type
05192  f20a  c9 04             cmp #reltyp     relative file?
05193  f20c  d0 02             bne af25
05194  f20e  09 80             ora #$80        set bit 7
05195  f210  20 b6 ec   af25   jsr putbyt      Byte to active buffer of LINDEX
                                                channel
05196  f213  68                pla             track link
05197  f214  8d 86 43          sta filtrk
05198  f217  20 b6 ec          jsr putbyt      Byte to active buffer of LINDEX
                                                channel
05199  f21a  68                pla             sector link
05200  f21b  8d 8b 43          sta filsec      set link in table & entry
05201  f21e  20 b6 ec          jsr putbyt
05202  f221  20 95 fa          jsr getact      Get active buffer number
05203  f224  a8                tay
05204  f225  ad 80 43          lda filtbl      pointer to requested drive
05205  f228  aa                tax
05206  f229  a9 10             lda #16         name length
05207  f22b  20 69 e0          jsr trname      Transfer filename from command
                                                string to buffer
05208  f22e  a0 10             ldy #16
05209  f230  a9 00             lda #$00        pad with 00s
05210  f232  91 27       af30  sta (dirbuf),y  & replace links
05211  f234  c8                iny
05212  f235  c0 1b             cpy #27
05213  f237  90 f9             bcc af30
05214  f239  a5 c5             lda type        current file type
05215  f23b  c9 04             cmp #reltyp     rel?
05216  f23d  d0 13             bne af50        no
05217  f23f  a0 10             ldy #16         only for relative files
05218  f241  ad 4c 43          lda trkss       track link for side sect to
05219  f244  91 27             sta (dirbuf),y  directory buffer pointer
05220  f246  c8                iny
05221  f247  ad 4d 43          lda secss       same with sector link
05222  f24a  91 27             sta (dirbuf),y  directory buffer pointer
05223  f24c  c8                iny
05224  f24d  ad 4b 43          lda rec         and record length
05225  f250  91 27             sta (dirbuf),y  directory buffer pointer
05226  f252  20 5b f0   af50   jsr drtwrt      Direct block write
05227  f255  68                pla
05228  f256  85 15             sta lindx       logical index, channel number
05229  f258  aa                tax
05230  f259  68                pla
05231  f25a  85 16             sta sa          current secondary address
05232  f25c  ad 97 43          lda delsec      sector of first available entry
```

```
line   addr  object     source code

05233  f25f  29 1f            and #$1f
05234  f261  85 86            sta filent       table of sector numbers in directory
05235  f263  ad 98 43         lda delind       index of first available entry
05236  f266  29 e0            and #$e0
05237  f268  05 86            ora filent
05238  f26a  85 86            sta filent
05239  f26c  9d 6b 43         sta dirent,x     index of directory entry by buffer
05240  f26f  a5 c5            lda type         current file type
05241  f271  0a               asl a
05242  f272  29 1e            and #$1e
05243  f274  05 12            ora drvnum       current drive number
05244  f276  85 8b            sta fildat       drive number, pattern
05245  f278  60               rts
05246  f279
05247  f279
05248  f279  ===> Open a channel <===
05249  f279
05250  f279  a5 16      open   lda sa           current secondary address
05251  f27b  8d 3b 43         sta tempsa       temp for 1f
05252  f27e  20 b6 dc         jsr cmdset       Initialize cmd tables & pointers
05253  f281  8e 7a 43         stx cmdnum
05254  f284  ae 00 43         ldx cmdbuf       1st input character
05255  f287  ad 3b 43         lda tempsa       temporary secondary address
05256  f28a  d0 29            bne op021        if not 0 (LOAD)
05257  f28c  e0 2a            cpx #'*'
05258  f28e  d0 25            bne op021        no
05259  f290  a5 11            lda prgtrk       last track or 0
05260  f292  f0 49            beq op0415       if 0
05261  f294  4a               lsr a
05262  f295  85 13            sta track        current track number
05263  f297  a9 00            lda #$00
05264  f299  2a               rol a
05265  f29a  85 12            sta drvnum       current drive number
05266  f29c  09 04            ora #prgtyp+prgtyp
05267  f29e  85 8b            sta fildat       drive number, pattern
05268  f2a0  20 35 da         jsr setlds       Turn on LED for current drive
05269  f2a3  ad 74 43         lda prgsec       last program sector
05270  f2a6  85 14            sta sector       current sector number
05271  f2a8  20 47 f7         jsr opnrch       Open a read channel with two buffers
05272  f2ab  a5 8b            lda fildat       drive number, pattern
05273  f2ad  a6 15      endrd ldx lindx        logical index, channel number
05274  f2af  99 90 00         sta filtyp,y     file type flags, channel 0-7
05275  f2b2  4c 9f db         jmp endcmd       Terminate command successfully
05276  f2b5
05277  f2b5  e0 24      op021  cpx #'$'         load the directory
05278  f2b7  d0 1d            bne op041
05279  f2b9  ad 3b 43         lda tempsa       temporary secondary address
05280  f2bc  d0 03            bne op04
05281  f2be  4c 09 f5         jmp loadir       Load the directory ($)
05282  f2c1
05283  f2c1
05284  f2c1
```

```
line   addr  object      source code

05285  f2c1  ===> Open directory as a sequential file <===
05286  f2c1
05287  f2c1  20 d2 db   op04   jsr simprs        Simple parser
05288  f2c4  a9 12             lda #18          directory track
05289  f2c6  85 13             sta track        current track number
05290  f2c8  a9 00             lda #$00
05291  f2ca  85 14             sta sector       current sector number
05292  f2cc  20 47 f7          jsr opnrch       Open a read channel with two buffers
05293  f2cf  a5 12             lda drvnum       current drive number
05294  f2d1  09 02             ora #seqtyp+seqtyp     (2* the sequential file type)
05295  f2d3  4c ad f2          jmp endrd
05296  f2d6
05297  f2d6  e0 23       op041 cpx #'#'         open for direct access
05298  f2d8  d0 12             bne op042
05299  f2da  4c 37 e8          jmp opnblk       Open direct access buffer ("#")
05300  f2dd
05301  f2dd  a9 04       op0415 lda #prgtyp+prgtyp     program type
05302  f2df  8d 9c 43          sta typflg       match by type of file
05303  f2e2  a9 00             lda #$00
05304  f2e4  85 12             sta drvnum       current drive number
05305  f2e6  8d 94 43          sta lstdrv       last drive without error: default
05306  f2e9  20 ff ec          jsr initdr
05307  f2ec  20 e6 db   op042  jsr prscln       Find colon in command string
05308  f2ef  d0 04             bne op049        found ":"?
05309  f2f1  a2 00             ldx #$00
05310  f2f3  f0 0c             beq op20
05311  f2f5  8a         op049  txa              found comma?
05312  f2f6  f0 05             beq op10
05313  f2f8  a9 30             lda #badsyn      something amiss
05314  f2fa  4c c9 db          jmp cmderr       Command level error handling
05315  f2fd
05316  f2fd  88         op10   dey              so it points to the ":"
05317  f2fe  f0 01             beq op20
05318  f300  88                dey              character preceeding the colon
05319  f301  8c 80 43   op20   sty filtbl       pointer to drive in cmd
05320  f304  a9 8d             lda #$8d         look for shifted CR
05321  f306  20 69 dc          jsr parse        Store desired character in CHAR
05322  f309  e8                inx              comma counter
05323  f30a  8e 7e 43          stx f2cnt        file stream 2 count
05324  f30d  20 10 dd          jsr onedrv       Set first drive & table pointers
05325  f310  20 10 de          jsr optsch       Determine optimal search for LOOKUP
                                                 and FINFIL
05326  f313  20 c9 de          jsr ffst         Find starting entry in directory
05327  f316  a2 00             ldx #$00         default
05328  f318  8e 4b 43          stx rec          record size
05329  f31b  8e 9d 43          stx mode         active file mode (R/W)
05330  f31e  86 c5             stx type         current file type
05331  f320  e8                inx
05332  f321  ec 7d 43          cpx flcnt
05333  f324  b0 10             bcs op40         if 0, no wild cards
05334  f326  20 bf f4          jsr cktm         determine S P U R
05335  f329  e8                inx
05336  f32a  ec 7d 43          cpx flcnt        another wild card?
```

```
line   addr  object     source code

05337  f32d  b0 07            bcs op40        if 01, there's only one
05338  f32f  c0 04            cpy #reltyp     REL?
05339  f331  f0 37            beq op60        if yes, set record size
05340  f333  20 bf f4         jsr cktm        determine R W A M
05341  f336  ae 3b 43  op40    ldx tempsa      temporary secondary address
05342  f339  86 16            stx sa          current secondary address
05343  f33b  e0 02            cpx #2          SA >1?
05344  f33d  b0 0b            bcs op45        not load or save
05345  f33f  8e 9d 43          stx mode        0=READ/LOAD   1=WRITE/SAVE
05346  f342  a5 c5            lda type        current file type
05347  f344  d0 1a            bne op50        type from parameters
05348  f346  a9 02            lda #prgtyp     'prg'
05349  f348  85 c5            sta type        current file type
05350  f34a  a5 c5     op45    lda type        current file type
05351  f34c  d0 12            bne op50        type from parameters
05352  f34e  a5 8b            lda fildat      drive number, pattern
05353  f350  29 0e            and #typmsk
05354  f352  4a               lsr a
05355  f353  85 c5            sta type        type from file
05356  f355  ad 86 43         lda filtrk
05357  f358  29 3f            and #$3f
05358  f35a  d0 04            bne op50        yes, it exists
05359  f35c  a9 01            lda #seqtyp     'seq'
05360  f35e  85 c5            sta type        default is seq
05361  f360  ad 9d 43  op50    lda mode        active file mode (R/W)
05362  f363  c9 01            cmp #wtmode     'W'?
05363  f365  f0 1a            beq op75
05364  f367  4c f9 f3         jmp op90
05365  f36a
05366  f36a                                   handle relative file:
05367  f36a  bc 80 43  op60    ldy filtbl,x    get record size
05368  f36d  b9 00 43          lda cmdbuf,y    command buffer
05369  f370  8d 4b 43          sta rec         record size
05370  f373  ad 86 43          lda filtrk
05371  f376  29 3f            and #$3f
05372  f378  d0 bc            bne op40        it's here -- read
05373  f37a  a9 01            lda #wtmode     use write to open
05374  f37c  8d 9d 43          sta mode        active file mode (R/W)
05375  f37f  d0 b5            bne op40        branch always
05376  f381  ad 86 43  op75    lda filtrk
05377  f384  29 80            and #$80        deleted file?
05378  f386  aa               tax             if not,
05379  f387  d0 16            bne op81        "replace" character in file name?
05380  f389  a9 20            lda #$20
05381  f38b  24 8b            bit fildat      was file closed properly?
05382  f38d  f0 06            beq op80
05383  f38f  20 45 e3         jsr deldir      Delete the entry in the directory
05384  f392  4c 9b f4         jmp opwrt       Open a file to write
05385  f395
05386  f395  ad 86 43  op80    lda filtrk
05387  f398  29 3f            and #$3f
05388  f39a  d0 03            bne op81
05389  f39c  4c 9b f4         jmp opwrt       Open a file to write
```

```
line   addr  object      source code

05390  f39f
05391  f39f                                  *** "replace file" command detected
                                             here
05392  f39f  ad 00 43   op81   lda cmdbuf     1st character input
05393  f3a2  c9 40             cmp #'@'      replace character
05394  f3a4  f0 0d             beq op82
05395  f3a6  8a                txa           joker flag
05396  f3a7  d0 05             bne op815     set?
05397  f3a9  a9 63             lda #flexst   FILE EXISTS error
05398  f3ab  4c c9 db          jmp cmderr    Command level error handling
05399  f3ae
05400  f3ae  a9 33       op815 lda #badfn
05401  f3b0  4c c9 db          jmp cmderr    Command level error handling
05402  f3b3                                  ***************
                                             ***************
05403  f3b3                                  This routine may have a bug!
05404  f3b3  a5 8b       op82  lda fildat     is directory file type
05405  f3b5  29 0e             and #typmsk
05406  f3b7  4a                lsr a
05407  f3b8  c5 c5             cmp type      same as in command line?
05408  f3ba  d0 64             bne op115
05409  f3bc  c9 04             cmp #reltyp   REL?
05410  f3be  f0 60             beq op115
05411  f3c0  20 e6 f7          jsr opnwch    Open a write channel with two
                                             buffers
05412  f3c3  a5 15             lda lindx     logical index, channel number
05413  f3c5  8d 75 43          sta wlindx    write LINDX
05414  f3c8  a9 11             lda #irsa     to read internal channel
05415  f3ca  85 16             sta sa        current secondary address
05416  f3cc  20 6e ed          jsr fndrch    Find the assigned read channel
05417  f3cf  ad 9a 43          lda index     current index in buffer
05418  f3d2  20 c1 f0          jsr setpnt    Set up pointer into active data
                                             buffer
05419  f3d5  a0 00             ldy #$00
05420  f3d7  b1 27             lda (dirbuf),y  set bit 5 in file type byte (file
                                             open)
05421  f3d9  09 20             ora #$20      set replace bit
05422  f3db  91 27             sta (dirbuf),y  directory buffer pointer
05423  f3dd  a0 1a             ldy #26
05424  f3df  a5 13             lda track     current track number
05425  f3e1  91 27             sta (dirbuf),y  at position 26
05426  f3e3  c8                iny           #27
05427  f3e4  a5 14             lda sector    current sector number
05428  f3e6  91 27             sta (dirbuf),y  at position 27 in directory entry
05429  f3e8  a5 86             lda filent    table of sector numbers in directory
05430  f3ea  ae 75 43          ldx wlindx    write LINDX
05431  f3ed  9d 6b 43          sta dirent,x  index of directory entry by buffer
05432  f3f0  20 3b f9          jsr curblk    Read track & sector from header
05433  f3f3  20 5b f0          jsr drtwrt    Direct block write
05434  f3f6  4c a7 f4          jmp opfin
05435  f3f9                                  ***************
                                             ***************
05436  f3f9  ad 86 43   op90   lda filtrk
```

```
line   addr  object     source code

05437  f3fc  29 3f            and #$3f        test if file exists — 0 if not
05438  f3fe  d0 05            bne op100
05439  f400  a9 62            lda #flntfd     FILE NOT FOUND error
05440  f402  4c c9 db         jmp cmderr      Command level error handling
05441  f405
05442  f405  ad 9d 43   op100  lda mode        active file mode (R/W)
05443  f408  c9 03            cmp #mdmode     'M' (modify)
05444  f40a  f0 0b            beq op110
05445  f40c  a9 20            lda #$20        check bit 5
05446  f40e  24 8b            bit fildat      file closed properly?
05447  f410  f0 05            beq op110
05448  f412  a9 60            lda #filopn     write FILE OPEN error
05449  f414  4c c9 db         jmp cmderr      Command level error handling
05450  f417
05451  f417  a5 8b      op110 lda fildat      drive number, pattern
05452  f419  4a               lsr a           mask
05453  f41a  29 0f            and #$0f        type is in index table
05454  f41c  c5 c5            cmp type        current file type
05455  f41e  f0 05            beq op120
05456  f420  a9 64      op115 lda #mistyp     FILE TYPE MISMATCH error
05457  f422  4c c9 db         jmp cmderr      Command level error handling
05458  f425
05459  f425  a0 00      op120 ldy #$00
05460  f427  8c 7f 43         sty f2ptr       file stream 2 pointer
05461  f42a  ae 9d 43         ldx mode        active file mode (R/W)
05462  f42d  e0 02            cpx #apmode     'A' (Append)?
05463  f42f  d0 1a            bne op125
05464  f431  c9 04            cmp #reltyp     REL?
05465  f433  f0 eb            beq op115
05466  f435  b1 27            lda (dirbuf),y  flag for open
05467  f437  29 4f            and #$4f
05468  f439  91 27            sta (dirbuf),y  directory buffer pointer
05469  f43b  a5 16            lda sa          current secondary address
05470  f43d  48               pha
05471  f43e  a9 11            lda #irsa       internal read channel
05472  f440  85 16            sta sa          current secondary address
05473  f442  20 3b f9         jsr curblk      Read track & sector from header
05474  f445  20 5b f0         jsr drtwrt      Direct block write
05475  f448  68               pla             restore lf
05476  f449  85 16            sta sa          current secondary address
05477  f44b  20 5b f4   op125 jsr opread      Open a file to read
05478  f44e  ad 9d 43         lda mode        active file mode (R/W)
05479  f451  c9 02            cmp #apmode     'A'?
05480  f453  d0 52            bne opfin
05481  f455  20 df f4         jsr append      Read file, then append info to the
                                               end of it
05482  f458  4c 9f db         jmp endcmd      Terminate command successfully
05483  f45b
05484  f45b
05485  f45b
```

```
line   addr  object      source code

05486  f45b  ===> Open a file to read <===
05487  f45b
05488  f45b  a0 13       opread ldy #19          track first side sector
05489  f45d  b1 27              lda (dirbuf),y   directory buffer pointer
05490  f45f  8d 4c 43           sta trkss        side sector track number
05491  f462  c8                 iny
05492  f463  b1 27              lda (dirbuf),y   sector first side sector
05493  f465  8d 4d 43           sta secss        side sector sector number
05494  f468  c8                 iny
05495  f469  b1 27              lda (dirbuf),y   record length
05496  f46b  ae 4b 43           ldx rec          record size
05497  f46e  8d 4b 43           sta rec          record size
05498  f471  8a                 txa              previous length
05499  f472  f0 0a              beq op130        zero?
05500  f474  cd 4b 43           cmp rec          same as current length?
05501  f477  f0 05              beq op130
05502  f479  a9 50              lda #norec       RECORD NOT PRESENT error
05503  f47b  20 c9 db           jsr cmderr       Command level error handling
05504  f47e
05505  f47e  ae 7f 43   op130  ldx f2ptr         file stream 2 pointer
05506  f481  bd 86 43           lda filtrk,x
05507  f484  29 3f              and #$3f
05508  f486  85 13              sta track        current track number
05509  f488  bd 8b 43           lda filsec,x
05510  f48b  85 14              sta sector       current sector number
05511  f48d  20 47 f7           jsr opnrch       Open a read channel with two buffers
05512  f490  a4 15              ldy lindx        logical index, channel number
05513  f492  ae 7f 43           ldx f2ptr        file stream 2 pointer
05514  f495  b5 86              lda filent,x
05515  f497  99 6b 43           sta dirent,y
05516  f49a  60                 rts
05517  f49b
05518  f49b
05519  f49b  ===> Open a file to write <===
05520  f49b
05521  f49b  a5 8b       opwrt  lda fildat       drive number, pattern
05522  f49d  29 01              and #$01         mask off non-drive bits
05523  f49f  85 12              sta drvnum       current drive number
05524  f4a1  20 e6 f7           jsr opnwch       Open a write channel with two
                                                 buffers
05525  f4a4  20 a9 f1           jsr addfil       Add new filename to directory
05526  f4a7  a5 16       opfin  lda sa           current secondary address
05527  f4a9  c9 02              cmp #$02         >1, then not a program file
05528  f4ab  b0 0f              bcs opf1
05529  f4ad  20 3e f9           jsr gethdr
05530  f4b0  a5 13              lda track        current track link
05531  f4b2  0a                 asl a
05532  f4b3  05 12              ora drvnum       current drive number
05533  f4b5  85 11              sta prgtrk       last program accessed
05534  f4b7  a5 14              lda sector       current sector link
05535  f4b9  8d 74 43           sta prgsec       last program sector
05536  f4bc  4c 9f db   opf1    jmp endcmd       Terminate command successfully
05537  f4bf
```

```
line   addr  object      source code

05538  f4bf
05539  f4bf  ===> Check mode or file type <===
05540  f4bf
05541  f4bf  bc 80 43   cktm   ldy filtbl,x     get pointer
05542  f4c2  b9 00 43          lda cmdbuf,y     get character
05543  f4c5  a0 04             ldy #nmodes
05544  f4c7  88         ckml   dey
05545  f4c8  30 08             bmi ckm2
05546  f4ca  d9 cb d2          cmp modlst,y     file modes R W A M
05547  f4cd  d0 f8             bne ckml
05548  f4cf  8c 9d 43          sty mode                   0 1 2 3
05549  f4d2  a0 05      ckm2   ldy #ntypes
05550  f4d4  88         ckt1   dey
05551  f4d5  30 07             bmi ckt2         no valid type
05552  f4d7  d9 cf d2          cmp tplst,y      file types D S P U L
05553  f4da  d0 f8             bne ckt1
05554  f4dc  84 c5             sty type                   0 1 2 3 4
05555  f4de  60         ckt2   rts
05556  f4df
05557  f4df
05558  f4df  ===> Read file, then append info to the end of it <===
05559  f4df
05560  f4df  20 62 e6   append jsr gcbyte       get a byte from the data channel
05561  f4e2  a9 80             lda #1rf
05562  f4e4  20 ae f8          jsr tstflg       is bit 7 in ($90,x) set (signals
05563  f4e7  f0 f6             beq append       last byte in file)?
05564  f4e9  20 97 f9          jsr rdlnk        Set TRACK & SECTOR from link in
                                                 buffer
05565  f4ec  a6 14             ldx sector       sector link >255?
05566  f4ee  e8                inx
05567  f4ef  8a                txa
05568  f4f0  d0 05             bne ap30         no
05569  f4f2  20 24 ee          jsr wrt0         get another block
05570  f4f5  a9 02             lda #$02         buffer pointer = 2
05571  f4f7  20 c1 f0   ap30   jsr setpnt       Set up pointer into active data
                                                 buffer
05572  f4fa  a6 15             ldx lindx        logical index, channel number
05573  f4fc  a9 01             lda #$01         set write flag
05574  f4fe  95 98             sta chnrdy,x     write, read, eoi flags, channel
                                                 status
05575  f500  a9 80             lda #$80         channel bit
05576  f502  05 15             ora lindx        channel nr bit 7 set
05577  f504  a6 16             ldx sa           current secondary address
05578  f506  95 a2             sta lintab,x     to drive control table
05579  f508  60                rts
05580  f509
05581  f509
05582  f509  ===> Load the directory ($) <===
05583  f509
05584  f509  a9 0b      loadir lda #ldcmd
05585  f50b  8d 7a 43          sta cmdnum
05586  f50e  ae 79 43          ldx cmdsiz
05587  f511  ca                dex              -1
```

```
line   addr  object     source code

05588  f512  d0 17            bne ld01        =0?
05589  f514  a9 2a            lda #'*'        general joker as 1st character to
                                               input buffer
05590  f516  8d 00 43         sta cmdbuf      command buffer
05591  f519  a9 80            lda #$80        set joker flag
05592  f51b  8d 86 43         sta filtrk      first file link (track)
05593  f51e  0d 94 43         ora lstdrv      last drive without error: default
05594  f521  85 8b            sta fildat      default flag drive number
05595  f523  ee 7d 43         inc flcnt       =1
05596  f526  ee 7e 43         inc f2cnt       =1
05597  f529  d0 41            bne ld10        branch always
05598  f52b
05599  f52b  ca          ld01  dex             more than one extra character
                                               following?
05600  f52c  d0 26            bne ld03        yes
05601  f52e  ad 01 43         lda cmdbuf+1    should be drive number
05602  f531  20 bb dd         jsr tst0v1      Test for 0 or 1
05603  f534  30 1e            bmi ld03        0 or 1 selected? no
05604  f536  29 01            and #$01        load one directory
05605  f538  85 8b            sta fildat      default flag drive number
05606  f53a  85 12            sta drvnum      current drive number
05607  f53c  20 ff ec         jsr initdr
05608  f53f  ee 7d 43         inc flcnt       =1
05609  f542  ee 7e 43         inc f2cnt       =1
05610  f545  ee 80 43         inc filtbl      =1
05611  f548  a9 80            lda #$80        set joker flag
05612  f54a  8d 86 43         sta filtrk      first file link (track)
05613  f54d  a9 2a            lda #'*'
05614  f54f  8d 01 43         sta cmdbuf+1    as 2nd chr in input buffer
05615  f552  d0 18            bne ld10        always
05616  f554
05617  f554  20 e6 db    ld03  jsr prscln      Find colon in command string
05618  f557  d0 05            bne ld05        ':' found
05619  f559  20 df dc         jsr cmdrst      Zero all important variables and
                                               pointers
05620  f55c  a0 03            ldy #$03
05621  f55e  88          ld05  dey
05622  f55f  88              .  dey
05623  f560  8c 80 43         sty filtbl      points to drive in cmd line
05624  f563  20 01 dc         jsr tc35        parse and set tables
05625  f566  20 95 dd         jsr fslset      Set pointers to one file stream and
                                               check type
05626  f569  20 1e dd         jsr alldrs      Set up all drives from F2CNT
05627  f56c  20 10 de    ld10 jsr optsch      Determine optimal search for LOOKUP
                                               and FINFIL
05628  f56f  20 b4 e1         jsr newdir      New directory in listing
05629  f572  20 c9 de         jsr ffst        Find starting entry in directory
05630  f575  20 54 da         jsr stdir       Start directory loading
05631  f578  20 b8 ed         jsr getbyt      Read one byte from the active buffer
05632  f57b  a6 15            ldx lindx       logical index, channel number
05633  f57d  95 b5            sta chndat,x    data byte in output register
05634  f57f  a5 12            lda drvnum      current drive number
05635  f581  8d 94 43         sta lstdrv      current drive number
```

```
line   addr  object     source code

05636  f584  09 04            ora #$04          'prg' flag (shifted left)
05637  f586  95 90            sta filtyp,x      file type flags, channel 0-7
05638  f588  a9 00            lda #$00          reset input buffer pointer
05639  f58a  85 45            sta buftab+cbptr
05640  f58c  60               rts
05640  f58d
05641  f58d                   .lib close
```

```
line   addr  object     source code

05643  f58d  ===> Close the file related to the specified sec. address <===
05644  f58d
05645  f58d  a5 16      close  lda sa           current secondary address
05646  f58f  d0 0b             bne cls10        =0 (LOAD flag)?
05647  f591  a9 00             lda #$00
05648  f593  8d 46 43          sta dirlst       directory listing flag
05649  f596  20 a4 ee          jsr frechn       Free channel associated with SA
05650  f599  4c d3 f0   cls05  jmp freich       Free both internal channels
05651  f59c
05652  f59c  c9 0f      cls10  cmp #15
05653  f59e  f0 0c             beq clsall       yes: close all channels
05654  f5a0  20 ba f5          jsr clschn       Close file with specified secondary
                                                address
05655  f5a3  a5 16             lda sa           current secondary address
05656  f5a5  c9 02             cmp #$02         load, save
05657  f5a7  90 f0             bcc cls05
05658  f5a9  4c 9f db          jmp endcmd       Terminate command successfully
05659  f5ac
05660  f5ac
05661  f5ac  ===> Close all <===
05662  f5ac
05663  f5ac  a9 0e      clsall lda #14          init counter for lf
05664  f5ae  85 16             sta sa           current secondary address
05665  f5b0  20 ba f5   cls20  jsr clschn       Close file with specified secondary
                                                address
05666  f5b3  c6 16             dec sa           decrement counter
05667  f5b5  10 f9             bpl cls20        while not negative
05668  f5b7  4c 9f db          jmp endcmd       Terminate command successfully
05669  f5ba
05670  f5ba
05671  f5ba  ===> Close file with specified secondary address <===
05672  f5ba
05673  f5ba  a6 16      clschn ldx sa           current secondary address
05674  f5bc  b5 a2             lda lintab,x
05675  f5be  c9 ff             cmp #$ff         channel allocated?
05676  f5c0  d0 01             bne clsc28       yes
05677  f5c2  60                rts
05678  f5c3
05679  f5c3  29 0f      clsc28 and #$0f         mask channel
05680  f5c5  85 15             sta lindx        logical index, channel number
05681  f5c7  20 a6 ed          jsr typfil       Get current file type
05682  f5ca  c9 07             cmp #dirtyp      direct access?
05683  f5cc  f0 0f             beq clsc30
05684  f5ce  c9 04             cmp #reltyp
05685  f5d0  f0 11             beq clsrel       Sub to close relative file
05686  f5d2  20 89 ed          jsr fndwch       Find the assigned write channel
05687  f5d5  b0 09             bcs clsc31       not a write file
05688  f5d7  20 12 f6          jsr clswrt       Close a sequential file write
                                                channel
05689  f5da  20 a4 f6          jsr clsdir       Close directory after writing file
05690  f5dd  20 55 f6   clsc30 jsr mapout       Write out BAM to drive specified in
                                                LSTJOB
05691  f5e0  4c a4 ee   clsc31 jmp frechn       Free channel associated with SA
```

```
line   addr  object     source code

05692  f5e3
05693  f5e3
05694  f5e3  ===> Sub to close relative file <===
05695  f5e3
05696  f5e3  20 f1 f8   clsrel jsr scrub        Write out buffer if dirty
05697  f5e6  20 d6 eb          jsr dblbuf       Double buffer: switch
                                                active/inactive buffers
05698  f5e9  20 ae fc          jsr ssend        Set SS & BUFTAB to end of last
                                                record
05699  f5ec  a6 83             ldx ssnum        side sector number
05700  f5ee  86 08             stx t4
05701  f5f0  e6 08             inc t4
05702  f5f2  a9 00             lda #$00         pointer to side sector value
05703  f5f4  85 05             sta t1
05704  f5f6  85 06             sta t2
05705  f5f8  a5 84             lda ssind        (end) pointer in side sector
05706  f5fa  38                sec
05707  f5fb  e9 0e             sbc #ssioff-2    preceeding bytes
05708  f5fd  85 07             sta t3
05709  f5ff  20 53 fa          jsr sscalc       how many ss blocks needed?
05710  f602  a6 15             ldx lindx        logical index, channel number
05711  f604  a5 05             lda t1
05712  f606  95 59             sta nbkl,x       block count lo, channel 0-7
05713  f608  a5 06             lda t2
05714  f60a  95 61             sta nbkh,x       block count hi, channel 0-7
05715  f60c  20 a4 f6          jsr clsdir       Close directory after writing file
05716  f60f                                     bit 6 in (90,x) set?
05717  f60f  4c a4 ee          jmp frechn       Free channel associated with SA
05718  f612
05719  f612
05720  f612  ===> Close a sequential file write channel <===
05721  f612
05722  f612  a6 15      clswrt ldx lindx        logical index, channel number
05723  f614  b5 59             lda nbkl,x       block count lo, channel 0-7
05724  f616  15 61             ora nbkh,x       block count hi, channel 0-7
05725  f618  d0 0c             bne clsw10       at least one block written
05726  f61a  20 e1 f0          jsr getpnt       Get the active buffer pointer
05727  f61d  c9 02             cmp #$02
05728  f61f  d0 05             bne clsw10       at least one byte written
05729  f621  a9 0d             lda #cr
05730  f623  20 b6 ec          jsr putbyt       Byte to active buffer of LINDEX
                                                channel
05731  f626  20 e1 f0   clsw10 jsr getpnt       Get the active buffer pointer
05732  f629  c9 02             cmp #$02
05733  f62b  d0 0f             bne clsw20       not an empty buffer
05734  f62d  20 d6 eb          jsr dblbuf       switch buffers
05735  f630  a6 15             ldx lindx        logical index, channel number
05736  f632  b5 59             lda nbkl,x       block count lo, channel 0-7
05737  f634  d0 02             bne clsw15       decrement block count hi & lo
05738  f636  d6 61             dec nbkh,x       block count hi, channel 0-7
05739  f638  d6 59      clsw15 dec nbkl,x       block count lo, channel 0-7
05740  f63a  a9 00             lda #$00
05741  f63c  38          clsw20 sec              calculate end pointer
```

```
line   addr  object      source code

05742  f63d  e9 01             sbc #$01          back up one
05743  f63f  48                pha
05744  f640  a9 00             lda #$00          buffer pointer = 0
05745  f642  20 c1 f0          jsr setpnt        Set up pointer into active data
                                                  buffer
05746  f645  20 b6 ec          jsr putbyt        last character count
05747  f648  68                pla
05748  f649  20 b6 ec          jsr putbyt        Byte to active buffer of LINDEX
                                                  channel
05749  f64c  20 4a ed          jsr wrtbuf        write out last buffer
05750  f64f  20 87 ec          jsr watjob        Wait until job is completed
05751  f652  4c d6 eb          jmp dblbuf        make sure both buffers are OK
05752  f655
05753  f655
05754  f655  ===> Write out BAM to drive specified in LSTJOB <===
05755  f655
05756  f655  20 95 fa    mapout jsr getact       Get active buffer number
05757  f658  aa                 tax
05758  f659  bd 4e 43           lda lstjob,x      last job by buffer
05759  f65c  29 01       mo10   and #$01
05760  f65e  48                 pha               check BAM before writing
05761  f65f  aa                 tax
05762  f660  a9 00              lda #$00
05763  f662  85 14              sta sector        current sector number
05764  f664  bd e8 d2           lda ipbm,x        BAM address hi
05765  f667  85 09              sta temp+5
05766  f669  a9 00              lda #$00
05767  f66b  85 08              sta t4
05768  f66d  a9 01              lda #$01
05769  f66f  85 13              sta track         current track number
05770  f671
05771  f671
05772  f671  ===> Verify that BAM block count matches the bits <===
05773  f671
05774  f671  a5 13       mapchk lda track         current track number
05775  f673  0a                 asl a
05776  f674  0a                 asl a
05777  f675  a8                 tay
05778  f676  b1 08              lda (t4),y
05779  f678  85 07              sta t3
05780  f67a  c8                 iny
05781  f67b  b1 08              lda (t4),y
05782  f67d  85 04              sta t0            temporary work area
05783  f67f  c8                 iny
05784  f680  b1 08              lda (t4),y
05785  f682  85 05              sta t1
05786  f684  c8                 iny
05787  f685  b1 08              lda (t4),y
05788  f687  85 06              sta t2
05789  f689  20 bd d7           jsr avck          check bit map validity
05790  f68c  e6 13              inc track         next track
05791  f68e  a5 13              lda track         current track number
05792  f690  c9 24              cmp #maxtrk
```

```
line   addr  object     source code

05793  f692  d0 dd            bne mapchk       verify that BAM block count matches
                                                the bits
05794  f694  a9 12            lda #18
05795  f696  85 13            sta track        current track number
05796  f698  68               pla
05797  f699  a8               tay
05798  f69a  18               clc
05799  f69b  69 0c            adc #bamjob
05800  f69d  aa               tax
05801  f69e  98               tya
05802  f69f  09 90            ora #write
05803  f6a1  4c 9d f1         jmp doit         Do job, set up error count and exit
                                                if error returns
05804  f6a4
05805  f6a4
05806  f6a4  ===> Close directory after writing file <===
05807  f6a4
05808  f6a4  a6 15      clsdir ldx lindx        save logical index
05809  f6a6  8e 75 43          stx wlindx
05810  f6a9  a5 16             lda sa           current secondary address
05811  f6ab  48                pha
05812  f6ac  bd 6b 43          lda dirent,x     get directory sector
05813  f6af  48                pha
05814  f6b0  29 1f             and #$1f
05815  f6b2  85 14             sta sector       current sector number
05816  f6b4  68                pla
05817  f6b5  29 e0             and #$e0         get sector offset
05818  f6b7  09 02             ora #$02
05819  f6b9  8d 9a 43          sta index        current index in buffer
05820  f6bc  b5 90             lda filtyp,x     drive number in FILTYP
05821  f6be  29 01             and #$01
05822  f6c0  85 12             sta drvnum       current drive#
05823  f6c2  a9 12             lda #18
05824  f6c4  85 13             sta track        current track number
05825  f6c6  20 95 fa          jsr getact       allocate a buffer
05826  f6c9  48                pha
05827  f6ca  85 a1             sta jobnum       current job number
05828  f6cc  20 57 f0          jsr drtrd        read directory sector
05829  f6cf  a0 00             ldy #$00
05830  f6d1  bd ff f0          lda bufind,x     .X is job
05831  f6d4  85 1a             sta r0+1
05832  f6d6  ad 9a 43          lda index        copy lo byte of pointer into
                                                directory buffer
05833  f6d9  85 19             sta r0
05834  f6db  b1 19             lda (r0),y       file type
05835  f6dd  29 20             and #$20         file closed?
05836  f6df  f0 41             beq clsd5        not a replace file if zero
05837  f6e1  20 a6 ed          jsr typfil       Get current file type
05838  f6e4  f0 44             beq clsd6
05839  f6e6  b1 19             lda (r0),y       temporary result
05840  f6e8  29 8f             and #$8f         mask off replace bit
05841  f6ea  91 19             sta (r0),y
05842  f6ec  c8                iny              point to old track link
```

```
line   addr  object      source code

05843  f6ed  b1 19             lda (r0),y      copy it
05844  f6ef  85 13             sta track       current track number
05845  f6f1  84 06             sty t2
05846  f6f3  a0 1b             ldy #27         extract replacement link
05847  f6f5  b1 19             lda (r0),y      to last sector
05848  f6f7  48                pha
05849  f6f8  88                dey
05850  f6f9  b1 19             lda (r0),y      replacement track link
05851  f6fb  d0 0a             bne clsd4       if not zero — or we're in trouble,
                                                so
05852  f6fd  85 13             sta track       put replacement track link in TRACK
05853  f6ff  68                pla             replacement sector link
05854  f700  85 14             sta sector      current sector number
05855  f702  a9 67             lda #$67        track or sector error
05856  f704  20 5c d9          jsr cmder2
05857  f707  48          clsd4 pha             replacement track link
05858  f708  a9 00             lda #$00        clear t&s link to replacement file
05859  f70a  91 19             sta (r0),y
05860  f70c  c8                iny             at pos. 26 & 27
05861  f70d  91 19             sta (r0),y      temporary result
05862  f70f  68                pla
05863  f710  a4 06             ldy t2          original pointer value
05864  f712  91 19             sta (r0),y      temporary result
05865  f714  c8                iny             and insert at pos. 1 & 2
05866  f715  b1 19             lda (r0),y      move old sector link
05867  f717  85 14             sta sector
05868  f719  68                pla             replacement now becomes
05869  f71a  91 19             sta (r0),y      final sector link
05870  f71c  20 1d e3          jsr delfil      scratch replaced file
05871  f71f  4c 2a f7          jmp clsd6       finish closing
05872  f722
05873  f722  b1 19       clsd5 lda (r0),y      file type
05874  f724  29 0f             and #$0f        mask bits 0-3
05875  f726  09 80             ora #$80        set bit 7 (close bit)
05876  f728  91 19             sta (r0),y      store type
05877  f72a  ae 75 43    clsd6 ldx wlindx      active buffer number
05878  f72d  a0 1c             ldy #28         set number of blocks
05879  f72f  b5 59             lda nbkl,x      block count lo
05880  f731  91 19             sta (r0),y      at position 28 (count lo)
05881  f733  c8                iny             and hi at 29
05882  f734  b5 61             lda nbkh,x      block count hi, channel 0-7
05883  f736  91 19             sta (r0),y      temporary result
05884  f738  68                pla             buffer number
05885  f739  aa                tax
05886  f73a  a9 90             lda #write      write directory sector
05887  f73c  05 12             ora drvnum
05888  f73e  20 9d f1          jsr doit        Do job, set up error count and exit
                                                if error returns
05889  f741  68                pla             restore
05890  f742  85 16             sta sa          current secondary address
05891  f744  4c 89 ed          jmp fndwch      Find the assigned write channel
05892  f747
05893  f747
```

```
line   addr  object      source code

05894  f747  ===> Open a read channel with two buffers <===
05895  f747
05896  f747  a9 02       opnrch lda #$02          number of blocks to allocate
05897  f749  20 63 ee          jsr getrch        Open a new read channel
05898  f74c  20 b4 f7          jsr intpnt        Initialize variables for open
                                                 channel
05899  f74f  a5 c5             lda type          current file type
05900  f751  48                pha
05901  f752  0a                asl a             set file type flags
05902  f753  05 12             ora drvnum
05903  f755  95 90             sta filtyp,x
05904  f757  20 22 ed          jsr strdbl        Start double buffering (reading
                                                 ahead)
05905  f75a  a6 15              ldx lindx         logical index, channel number
05906  f75c  a5 13              lda track         current track number
05907  f75e  d0 04              bne or10
05908  f760  a5 14              lda sector        current sector number
05909  f762  95 bd              sta lstchr,x      use sector link as end pointer
05910  f764  68          or10   pla               file type
05911  f765  c9 04              cmp #reltyp
05912  f767  d0 3f              bne or30
05913  f769  a4 16              ldy sa            current secondary address
05914  f76b  b9 a2 00           lda lintab,y      set channel as
05915  f76e  09 40              ora #$40          read/write
05916  f770  99 a2 00           sta lintab,y
05917  f773  ad 4b 43           lda rec           record size
05918  f776  95 71              sta rs,x          record sizes table
05919  f778  20 03 ef           jsr getbuf        Get a free buffer
05920  f77b  10 03              bpl or20          found buffer for side sector?
05921  f77d  4c 8c ee           jmp gberr         nope
05922  f780
05923  f780  a6 15       or20   ldx lindx         logical index, channel number
05924  f782  95 79              sta ss,x          side sectors table
05925  f784  ac 4c 43           ldy trkss         copy side sector track link
05926  f787  84 13              sty track         current track number
05927  f789  ac 4d 43           ldy secss         copy side sector sector link
05928  f78c  84 14              sty sector        current sector number
05929  f78e  20 97 ec           jsr seth          set SS header
05930  f791  20 75 f9           jsr rdss          read it in
05931  f794  20 87 ec           jsr watjob        Wait until job is completed
05932  f797  a6 15       orow   ldx lindx         logical index, channel number
05933  f799  a9 02              lda #$02
05934  f79b  95 69              sta nr,x          pointer for write set for next
                                                 record
05935  f79d  a9 00              lda #$00          buffer pointer = 0
05936  f79f  20 c1 f0           jsr setpnt        Set up pointer into active data
                                                 buffer
05937  f7a2  20 39 fc           jsr rd40          set op first record
05938  f7a5  4c 3e f9           jmp gethdr        restore track & sector
05939  f7a8
05940  f7a8                                       sequential set up
05941  f7a8  20 d7 ed    or30   jsr rdbyt         Read byte from file
05942  f7ab  a6 15              ldx lindx         side sector buffer
```

```
line   addr  object     source code

05943  f7ad  95 b5            sta chndat,x     channel data byte
05944  f7af  a9 88            lda #rdytlk      set READ flag, reset EOI
05945  f7b1  95 98            sta chnrdy,x     channel status
05946  f7b3  60               rts
05947  f7b4
05948  f7b4
05949  f7b4  ===> Initialize variables for open channel <===
05950  f7b4
05951  f7b4  a6 15      intpnt ldx lindx        logical index, channel number
05952  f7b6  a5 12             lda drvnum       current drive#
05953  f7b8  b4 49             ldy buf0,x       channel buffer table 1
05954  f7ba  99 4e 43          sta lstjob,y     last job by buffer
05955  f7bd  b4 51             ldy buf1,x       channel buffer table 2
05956  f7bf  99 4e 43          sta lstjob,y     last job by buffer
05957  f7c2  99 03 10          sta jobs,y       queue
05958  f7c5  b5 49             lda buf0,x       channel buffer table 1
05959  f7c7  0a                asl a            1st buf times 2
05960  f7c8  a8                tay              = pointer in buffer pointer table
05961  f7c9  a9 02             lda #$02         pointer lo in
05962  f7cb  99 29 00          sta buftab,y     buffer 0 pointer lo
05963  f7ce  b5 51             lda buf1,x       channel buffer table 2
05964  f7d0  09 80             ora #$80         set bit 7 (buffer not active)
05965  f7d2  95 51             sta buf1,x       channel buffer table 2
05966  f7d4  0a                asl a            2nd buf times 2
05967  f7d5  a8                tay              = pointer
05968  f7d6  a9 02             lda #$02
05969  f7d8  99 29 00          sta buftab,y     buffer 0 pointer lo
05970  f7db  a9 00             lda #$00
05971  f7dd  95 59             sta nbkl,x       block count lo, channel 0-7
05972  f7df  95 61             sta nbkh,x       block count hi, channel 0-7
05973  f7e1  a9 00             lda #$00
05974  f7e3  95 bd             sta lstchr,x     channel last character pointer
05975  f7e5  60                rts
05976  f7e6
05977  f7e6
05978  f7e6  ===> Open a write channel with two buffers <===
05979  f7e6
05980  f7e6  20 47 d7   opnwch jsr intts        get first track and sector
05981  f7e9  a9 02             lda #$02         number of buffers
05982  f7eb  20 60 ee          jsr getwch       Open a new write channel
05983  f7ee  20 94 ec          jsr sethdr       Set up header for active buffer
05984  f7f1  20 b4 f7          jsr intpnt       Initialize variables for open
                                                 channel
05985  f7f4  a6 15             ldx lindx        logical index, channel number
05986  f7f6  a5 c5             lda type         current file type
05987  f7f8  48                pha
05988  f7f9  0a                asl a            bit shift left
05989  f7fa  05 12             ora drvnum       current drive#
05990  f7fc  95 90             sta filtyp,x     file type flags, channel 0-7
05991  f7fe  68                pla
05992  f7ff  c9 04             cmp #reltyp      REL?
05993  f801  f0 05             beq owl0
05994  f803  a9 01             lda #rdylst      active listener
```

```
line   addr  object     source code

05995  f805  95 98           sta chnrdy,x     channel status
05996  f807  60             rts
05997  f808
05998  f808  a4 16      ow10 ldy sa           current secondary address
05999  f80a  b9 a2 00       lda lintab,y
06000  f80d  29 3f          and #$3f         reset bits 6 & 7
06001  f80f  09 40          ora #$40         set bit 6 (R/W flag)
06002  f811  99 a2 00       sta lintab,y
06003  f814  ad 4b 43       lda rec          record size
06004  f817  95 71          sta rs,x         record sizes table
06005  f819  20 03 ef       jsr getbuf       Get a free buffer
06006  f81c  10 03          bpl ow20
06007  f81e  4c 8c ee       jmp gberr        no buffer
06008  f821
06009  f821  a6 15      ow20 ldx lindx        logical index, channel number
06010  f823  95 79          sta ss,x         side sectors table
06011  f825  20 c3 f9       jsr clrbuf       zeroize buffer
06012  f828  20 b0 d6       jsr nxtts        Next available track and sector
06013  f82b  a5 13          lda track        current track number
06014  f82d  8d 4c 43       sta trkss        side sector track number
06015  f830  a5 14          lda sector       current sector number
06016  f832  8d 4d 43       sta secss        side sector sector number
06017  f835  a6 15          ldx lindx        logical index, channel number
06018  f837  b5 79          lda ss,x         side sectors table
06019  f839  20 97 ec       jsr seth         get track & sector of first SS
06020  f83c  a9 00          lda #$00         buffer pointer
06021  f83e  20 eb f9       jsr setssp       Use SS pointer to set DIRBUF
                                            & BUFTAB
06022  f841  a9 00          lda #$00         set null link
06023  f843  20 95 f8       jsr putss        put byte into side sector
06024  f846  a9 11          lda #ssioff+1    set last character
06025  f848  20 95 f8       jsr putss        put byte into side sector
06026  f84b  a9 00          lda #$00         number of side sector
06027  f84d  20 95 f8       jsr putss        put byte into side sector
06028  f850  ad 4b 43       lda rec          record size
06029  f853  20 95 f8       jsr putss        put byte into side sector
06030  f856  a5 13          lda track        current track number
06031  f858  20 95 f8       jsr putss        put byte into side sector
06032  f85b  a5 14          lda sector       current sector number
06033  f85d  20 95 f8       jsr putss        put byte into side sector
06034  f860  a9 10          lda #ssioff      buffer pointer
06035  f862  20 eb f9       jsr setssp       Use SS pointer to set DIRBUF
                                            & BUFTAB
06036  f865  20 3e f9       jsr gethdr       get first track/sector
06037  f868  a5 13          lda track        current track number
06038  f86a  20 95 f8       jsr putss        put byte into side sector
06039  f86d  a5 14          lda sector       current sector number
06040  f86f  20 95 f8       jsr putss        put byte into side sector
06041  f872  20 6e f9       jsr wrtss        write it out
06042  f875  20 87 ec       jsr watjob       Wait until job is completed
06043  f878  a9 02          lda #$02         buffer pointer
06044  f87a  20 c1 f0       jsr setpnt       Set up pointer into active data
                                            buffer
```

```
line   addr  object     source code

06045  f87d  a6 15            ldx lindx       logical index, channel number
06046  f87f  38               sec
06047  f880  a9 00            lda #$00        length of record
06048  f882  f5 71            sbc rs,x        record sizes table
06049  f884  95 69            sta nr,x        next record pointers table
06050  f886  20 ca fd         jsr nulbuf      Set null records in active buffer
06051  f889  20 19 f9         jsr nullnk      Set track link to 0, sector link to
                                               last non-0 char in buf
06052  f88c  20 60 f9         jsr wrtout      store write job code
06053  f88f  20 87 ec         jsr watjob      Wait until job is completed
06054  f892  4c 97 f7         jmp orow        finish opening channel
06055  f895
06056  f895
06057  f895  ===> Put byte into side sector <===
06058  f895
06059  f895  48         putss  pha             data
06060  f896  a6 15            ldx lindx       active buffer number
06061  f898  b5 79            lda ss,x        side sector buffer number
06062  f89a  4c c2 ec         jmp putbl
06062  f89d
06063  f89d                   .lib tstflg
```

```
line   addr  object     source code

06065  f89d  ===> Set, clear and test flags <===
06066  f89d
06067  f89d  90 06      scflg  bcc clrflg       Clear flag
06068  f89f
06069  f89f
06070  f89f  ===> Set buffer pointers <===
06071  f89f
06072  f89f  a6 15      setflg ldx lindx        logical index, channel number
06073  f8a1  15 90            ora filtyp,x      set flag
06074  f8a3  d0 06            bne clrf10        always
06075  f8a5
06076  f8a5
06077  f8a5  ===> Clear flag <===
06078  f8a5
06079  f8a5  a6 15      clrflg ldx lindx        logical index, channel number
06080  f8a7  49 ff            eor #$ff          clear flag by flipping the bits
06081  f8a9  35 90            and filtyp,x      file type flags, channel 0-7
06082  f8ab  95 90      clrf10 sta filtyp,x
06083  f8ad  60               rts
06084  f8ae
06085  f8ae
06086  f8ae  ===> Test flag <===
06087  f8ae
06088  f8ae  a6 15      tstflg ldx lindx        logical index, channel number
06089  f8b0  35 90            and filtyp,x
06090  f8b2  60               rts
06091  f8b3
06092  f8b3
06093  f8b3  ===> Test if this is a write job <===
06094  f8b3
06095  f8b3  20 95 fa   tstwrt jsr getact       Get active buffer number
06096  f8b6  aa               tax
06097  f8b7  bd 4e 43         lda lstjob,x      last job code,
06098  f8ba  29 f0            and #$f0          mask off the drive bits to see if it
                                                 is a
06099  f8bc  c9 90            cmp #$90          write job. If so, this sets the Z
                                                 flag
06100  f8be  60               rts
06101  f8bf
06102  f8bf
06103  f8bf  ===> Test for active files <===
06104  f8bf       C=0 if file active X=ENTFND, Y=LINDEX
06105  f8bf       C=1 if file inactive X=18
06106  f8bf
06107  f8bf  a2 00      tstchn ldx #$00         start search at top
06108  f8c1  86 06      tstc20 stx t2           save to look on
06109  f8c3  b5 a2            lda lintab,x      current status
06110  f8c5  c9 ff            cmp #$ff          channel allocated?
06111  f8c7  d0 08            bne tstc40        if plus, then test it
06112  f8c9  a6 06      tstc30 ldx t2           not active
06113  f8cb  e8               inx               increment counter
06114  f8cc  e0 14            cpx #maxsa+2      and continue search while <16
06115  f8ce  90 f1            bcc tstc20        searched all
```

```
line   addr  object      source code

06116  f8d0  60                rts               yes. If none found, carry = 1
06117  f8d1
06118  f8d1  86 06       tstc40 stx t2
06119  f8d3  29 3f              and #$3f          mask channel, drive number
06120  f8d5  a8                 tay
06121  f8d6  b9 90 00           lda filtyp,y      use LINDX as index
06122  f8d9  29 01              and #$01          mask off non-drive bits
06123  f8db  85 05              sta t1
06124  f8dd  ae 45 43           ldx entfnd        entry found index
06125  f8e0  b5 8b              lda fildat,x      drive number for this entry
06126  f8e2  29 01              and #$01          mask off non-drive bits and see if
06127  f8e4  c5 05              cmp t1            the drives match
06128  f8e6  d0 e1              bne tstc30
06129  f8e8  b9 6b 43           lda dirent,y      now check if the directory sectors
06130  f8eb  d5 86              cmp filent,x      are match
06131  f8ed  d0 da              bne tstc30        if they are,
06132  f8ef  18                 clc               flag all tests passed and active
                                                   file found,
06133  f8f0  60                 rts               so return with carry = 0
06134  f8f1
06135  f8f1
06136  f8f1  ===> Write out buffer if dirty <===
06137  f8f1       A buffer is "dirty" if the copy in RAM has been modified
06138  f8f1       so it does not match the copy on disk
06139  f8f1
06140  f8f1  20 a0 fa   scrub   jsr gaflgs        Get active buffer
06141  f8f4  50 06              bvc scrl          not dirty
06142  f8f6  20 60 f9           jsr wrtout        write it out
06143  f8f9  20 87 ec           jsr watjob        Wait until job is completed
06144  f8fc  60         scrl    rts
06145  f8fd
06146  f8fd
06147  f8fd  ===> Put TRACK & SECTOR into buffer <===
06148  f8fd
06149  f8fd  20 2b f9   setlnk  jsr set00         Set up pointer to active buffer
06150  f900  a5 13              lda track         current track number
06151  f902  91 27              sta (dirbuf),y    directory buffer pointer
06152  f904  c8                 iny
06153  f905  a5 14              lda sector        current sector number
06154  f907  91 27              sta (dirbuf),y    directory buffer pointer
06155  f909  4c eb fb           jmp sdirty        Set buffer dirty flag
06156  f90c
06157  f90c
06158  f90c  ===> Set TRACK & SECTOR from link in buffer <===
06159  f90c
06160  f90c  20 2b f9   getlnk  jsr set00         Set up pointer to active buffer
06161  f90f  b1 27              lda (dirbuf),y    move the track link to
06162  f911  85 13              sta track         current track number
06163  f913  c8                 iny               and
06164  f914  b1 27              lda (dirbuf),y    the sector link to
06165  f916  85 14              sta sector        current sector number
06166  f918  60                 rts
06167  f919
```

```
line   addr  object      source code

06168  f919
06169  f919  ===> Set track link to 0, sector link to last non-0 char in buf <===
06170  f919
06171  f919  20 2b f9   nullnk jsr set00        Set up pointer to active buffer
06172  f91c  a9 00             lda #$00         store $00 as
06173  f91e  91 27             sta (dirbuf),y   track link,
06174  f920  c8                iny              then
06175  f921  a6 15             ldx lindx        get the active buffer number, load
                                                 the
06176  f923  b5 69             lda nr,x         next record pointers table
06177  f925  aa                tax              and, after
06178  f926  ca                dex              subtracting 1,
06179  f927  8a                txa              store the result as the sector link
                                                 in the
06180  f928  91 27             sta (dirbuf),y   directory buffer.
06181  f92a  60                rts
06182  f92b
06183  f92b
06184  f92b  ===> Set up pointer to active buffer <===
06185  f92b
06186  f92b  20 95 fa   set00  jsr getact       Get active buffer number
06187  f92e  0a                asl a            multiplied by 2: pointer in buffer
                                                 pointer table
06188  f92f  aa                tax
06189  f930  b5 2a             lda buftab+1,x   move the hi byte of the buffer
                                                 pointer
06190  f932  85 28             sta dirbuf+1     to the directory buffer
06191  f934  a9 00             lda #$00         and use $00 as
06192  f936  85 27             sta dirbuf       lo byte
06193  f938  a0 00             ldy #$00
06194  f93a  60                rts
06195  f93b
06196  f93b
06197  f93b  ===> Set TRACK & SECTOR from header <===
06198  f93b
06199  f93b  20 6e ed   curblk jsr fndrch       Find an unused channel
06200  f93e  20 95 fa   gethdr jsr getact       Get active buffer number
06201  f941  85 a1             sta jobnum
06202  f943  0a                asl a            multiply
06203  f944  0a                asl a            it
06204  f945  0a                asl a            by 8,
06205  f946  a8                tay              so it points to the range we want.
06206  f947  b9 23 10          lda hdrs+2,y     Now we first move the
06207  f94a  85 13             sta track        current track number
06208  f94c  b9 24 10          lda hdrs+3,y     and then the
06209  f94f  85 14             sta sector       current sector number
06210  f951  60                rts
06211  f952
06212  f952
06213  f952
```

```
line   addr  object      source code

06214  f952  ===> Do read and write jobs <===
06215  f952
06216  f952  a9 90       wrtab  lda #write
06217  f954  8d 3c 43           sta cmd          temporary job command
06218  f957  d0 28              bne sj10         always
06219  f959
06220  f959
06221  f959  ===> Store read job code $80 <===
06222  f959
06223  f959  a9 80       rdab   lda #read
06224  f95b  8d 3c 43           sta cmd          temporary job command
06225  f95e  d0 21              bne sj10         always
06226  f960
06227  f960
06228  f960  ===> Store write job code $90 <===
06229  f960
06230  f960  a9 90       wrtout lda #write
06231  f962  8d 3c 43           sta cmd          temporary job command
06232  f965  d0 26              bne sj20
06233  f967
06234  f967
06235  f967  ===> Store read job code $80 <===
06236  f967
06237  f967  a9 80       rdin   lda #read
06238  f969  8d 3c 43           sta cmd          temporary job command
06239  f96c  d0 1f              bne sj20
06240  f96e
06241  f96e
06242  f96e  ===> Write job code $90 <===
06243  f96e
06244  f96e  a9 90       wrtss  lda #write
06245  f970  8d 3c 43           sta cmd          temporary job command
06246  f973  d0 02              bne rds5
06247  f975
06248  f975
06249  f975  ===> Store read job code $80 <===
06250  f975
06251  f975  a9 80       rdss   lda #read
06252  f977  8d 3c 43    rds5   sta cmd          temporary job command
06253  f97a  a6 15               ldx lindx        logical index, channel number
06254  f97c  b5 79               lda ss,x         side sectors table
06255  f97e  aa                  tax              if the resulting buffer number is
                                                  less
06256  f97f  10 13               bpl sj30         than 127, branch, else
06257  f981  20 94 ec    sj10   jsr sethdr       Set up header for active buffer
06258  f984  20 95 fa           jsr getact       Get active buffer number
06259  f987  aa                  tax
06260  f988  a5 12               lda drvnum       current drive number
06261  f98a  9d 4e 43            sta lstjob,x     last job by buffer
06262  f98d  20 f6 fb    sj20   jsr cdirty       Clear buffer dirty flag
06263  f990  20 95 fa           jsr getact       Get active buffer number
06264  f993  aa                  tax
06265  f994  4c 0e f1    sj30   jmp setljb       Set up job using last job's drive
```

```
line   addr  object     source code

06266  f997
06267  f997
06268  f997  ===> Set TRACK & SECTOR from link in buffer <===
06269  f997
06270  f997  a9 00      rdlnk  lda #$00
06271  f999  20 c1 f0          jsr setpnt       Set up pointer into active data
                                                buffer
06272  f99c  20 b8 ed          jsr getbyt       Read one byte from the active buffer
                                                as the
06273  f99f  85 13             sta track        current track number
06274  f9a1  20 b8 ed          jsr getbyt       and the next as the
06275  f9a4  85 14             sta sector       current sector number
06276  f9a6  60                rts
06277  f9a7
06278  f9a7
06279  f9a7  ===> Move bytes between buffers <===
06280  f9a7       On entry, .A holds the number of bytes to move,
06281  f9a7                 .Y the source buffer number
06282  f9a7                 .X the destination buffer number
06283  f9a7
06284  f9a7  48         b0tob0 pha
06285  f9a8  a9 00             lda #$00
06286  f9aa  85 04             sta t0           temporary work area
06287  f9ac  85 06             sta t2
06288  f9ae  b9 ff f0          lda bufind,y     buffer (Y) hi
06289  f9b1  85 05             sta t1
06290  f9b3  bd ff f0          lda bufind,x     buffer (X) hi
06291  f9b6  85 07             sta t3
06292  f9b8  68                pla              use 'number of bytes to move'
06293  f9b9  a8                tay              as a
06294  f9ba  88                dey              count down index
06295  f9bb  b1 04      b02    lda (t0),y       temporary work area
06296  f9bd  91 06             sta (t2),y
06297  f9bf  88                dey
06298  f9c0  10 f9             bpl b02
06299  f9c2  60                rts
06300  f9c3
06301  f9c3
06302  f9c3  ===> Clear buffer <===
06303  f9c3
06304  f9c3  a8         clrbuf tay              buffer number hi address
06305  f9c4  b9 ff f0          lda bufind,y
06306  f9c7  85 05             sta t1
06307  f9c9  a9 00             lda #$00
06308  f9cb  85 04             sta t0           temporary work area
06309  f9cd  a8                tay
06310  f9ce  91 04      cb10   sta (t0),y       temporary work area
06311  f9d0  c8                iny              loop to fill buffers with zeros
06312  f9d1  d0 fb             bne cb10
06313  f9d3  60                rts
06314  f9d4
06315  f9d4
06316  f9d4
```

```
line   addr  object     source code

06317  f9d4  ===> Set side sector pointers to 0 <===
06318  f9d4
06319  f9d4  a9 00      ssset  lda #$00
06320  f9d6  20 de f9          jsr ssdir        Use SS pointer to set DIRBUF
06321  f9d9  a0 02             ldy #$02
06322  f9db  b1 27             lda (dirbuf),y   directory buffer pointer
06323  f9dd  60                 rts
06324  f9de
06325  f9de
06326  f9de  ===> Use SS pointer to set DIRBUF <===      (.A = lo byte)
06327  f9de
06328  f9de  85 27      ssdir  sta dirbuf       directory buffer pointer
06329  f9e0  a6 15             ldx lindx        the active buffer number
06330  f9e2  b5 79             lda ss,x         side sectors table
06331  f9e4  aa                tax
06332  f9e5  bd ff f0          lda bufind,x     buffer address hi
06333  f9e8  85 28             sta dirbuf+1     buffer pointer hi
06334  f9ea  60                rts
06335  f9eb
06336  f9eb
06337  f9eb  ===> Use SS pointer to set DIRBUF & BUFTAB <===    (.A is lo bite)
06338  f9eb
06339  f9eb  48         setssp pha
06340  f9ec  20 de f9          jsr ssdir        Use SS pointer to set DIRBUF
06341  f9ef  48                pha              buffer address hi
06342  f9f0  8a                txa              buffer number
06343  f9f1  0a                asl a            multiplied by 2
06344  f9f2  aa                tax              gives pointer to buffer pointer
                                                table
06345  f9f3  68                pla              hi
06346  f9f4  95 2a             sta buftab+1,x   buffer 0 pointer hi
06347  f9f6  68                pla              buffer pointer
06348  f9f7  95 29             sta buftab,x     buffer 0 pointer lo
06349  f9f9  60                rts
06350  f9fa
06351  f9fa
06352  f9fa  ===> Use SSNUM & SSIND to set SS & BUFTAB <===
06353  f9fa
06354  f9fa  20 68 fa   sspos  jsr sstest       to see if both are resident and
                                                within range
06355  f9fd  30 0e             bmi ssp10        N = set means out of range
06356  f9ff  50 13             bvc ssp20        V=clear means yes, in residence, so
                                                store
06357  fa01  a6 15             ldx lindx        get the active buffer number
06358  fa03  b5 79             lda ss,x         and the side sector buffer number
06359  fa05  20 1d fa          jsr ibrd         read in the side sector
06360  fa08  20 68 fa          jsr sstest       and do another test
06361  fa0b  10 07             bpl ssp20        if N-flag clear, it's in range, else
06362  fa0d  20 ae fc   ssp10  jsr ssend        Set SS & BUFTAB to end of last
                                                record
06363  fa10  2c e5 d2          bit er1          =127 (sets V bit)
06364  fa13  60                rts              V=0: all OK. V=1: out of range
06365  fa14
```

```
line   addr  object      source code

06366  fa14  a5 84       ssp20  lda ssind        OK, set pointer with index
06367  fa16  20 eb f9           jsr setssp       Use SS pointer to set DIRBUF
                                                 & BUFTAB
06368  fa19  2c e4 d2           bit er0          =63 (clears N & V bits)
06369  fa1c  60                 rts
06370  fa1d
06371  fa1d
06372  fa1d  ===> Indirect block read/write <===
06373  fa1d
06374  fa1d  85 a1       ibrd   sta jobnum       .A = buffer number for R/W
06375  fa1f  a9 80              lda #read
06376  fa21  d0 04              bne ibop
06377  fa23  85 a1       ibwt   sta jobnum       current job number
06378  fa25  a9 90              lda #write
06379  fa27  48          ibop   pha              push job code onto stack
06380  fa28  b5 90              lda filtyp,x     file type's drive number
06381  fa2a  29 01              and #$01         mask off non-drive bits
06382  fa2c  85 12              sta drvnum       current drive number
06383  fa2e  68                 pla              pull job code, then get the
06384  fa2f  05 12              ora drvnum       drive number
06385  fa31  8d 3c 43           sta cmd          temporary job command
06386  fa34  b1 27              lda (dirbuf),y   directory buffer pointer
06387  fa36  85 13              sta track        track
06388  fa38  c8                 iny
06389  fa39  b1 27              lda (dirbuf),y   & sector of requested side sector
                                                 block
06390  fa3b  85 14              sta sector       current sector number
06391  fa3d  a5 a1              lda jobnum       current job number
06392  fa3f  20 97 ec           jsr seth
06393  fa42  a6 a1              ldx jobnum       current job number
06394  fa44  4c a0 f1           jmp doit2        do the job
06395  fa47
06396  fa47
06397  fa47  ===> Get side sector pointers <===
06398  fa47
06399  fa47  a6 15       gsspnt ldx lindx        logical index, channel number
06400  fa49  b5 79              lda ss,x         side sectors table
06401  fa4b  4c e4 f0           jmp gp1          Set up pointers for DIRBUF
06402  fa4e
06403  fa4e
06404  fa4e  ===> Calculate side sectors <===
06405  fa4e
06406  fa4e  a9 78       scall  lda #nssp        the number of side sector pointers
                                                 in a buffer
06407  fa50  20 5e fa           jsr addt12       add number of side sectors needed *
                                                 120
06408  fa53  ca          sscalc dex              (X) = number of last side sector
06409  fa54  10 f8              bpl scall        branch if .X >= 0
06410  fa56  a5 07              lda t3
06411  fa58  4a                 lsr a            number of data block pointers in
                                                 last side sector
06412  fa59  20 5e fa           jsr addt12       added to (5,6)
06413  fa5c  a5 08              lda t4           number of side sector blocks needed
```

```
line   addr  object      source code

06414  fa5e  18          addtl2 clc              add (A) to (5,6)
06415  fa5f  65 05              adc tl
06416  fa61  85 05              sta tl
06417  fa63  90 02              bcc addrts
06418  fa65  e6 06              inc t2
06419  fa67  60          addrts rts
06420  fa68
06421  fa68
06422  fa68  ===> Test SSNUM & SSIND for range and residence <===
06423  fa68
06424  fa68  ------------------------------------------------
06425  fa68  On exit, the flags may have the following meaning:
06426  fa68
06427  fa68       N   range        V   residence          ER
06428  fa68       0   OK           0   yes                ER0
06429  fa68       0   maybe        1   no                 ER1
06430  fa68       1   bad          0   yes                ER2
06431  fa68       1   bad          0   no                 ER3
06432  fa68  ------------------------------------------------
06433  fa68
06434  fa68  20 d4 f9   sstest jsr ssset        number of side sector
06435  fa6b  c5 83             cmp ssnum        same as requested number?
06436  fa6d  d0 0e             bne st20
06437  fa6f  a4 84             ldy ssind        (end) pointer in side sector
06438  fa71  b1 27             lda (dirbuf),y   data block pointer (track number)
06439  fa73  f0 04             beq st10         if 0: no pointer available
06440  fa75  2c e4 d2          bit er0          if 63 (clears V & N) -- OK, resident
06441  fa78  60                rts              all OK
06442  fa79
06443  fa79  2c e6 d2   st10   bit er2          =191 (clears V, sets N)
06444  fa7c  60                rts              out of range
06445  fa7d
06446  fa7d  a5 83      st20   lda ssnum        side sector number
06447  fa7f  c9 06             cmp #nssl        more than 5 side sectors needed?
06448  fa81  b0 0a             bcs st30
06449  fa83  0a                asl a            times 2
06450  fa84  a8                tay              =pointer to side sector link table
06451  fa85  a9 04             lda #$04         position of first link
06452  fa87  85 27             sta dirbuf       directory buffer pointer
06453  fa89  b1 27             lda (dirbuf),y   directory buffer pointer
06454  fa8b  d0 04             bne st40         side sector already allocated?
06455  fa8d  2c e7 d2   st30   bit er3          255 - sets V & N
06456  fa90  60                rts              way out of range
06457  fa91
06458  fa91  2c e5 d2   st40   bit er1          127 sets V, clears N
06459  fa94  60                rts              not resident -- range??
06460  fa95
06461  fa95
06462  fa95  ===> Get active buffer number <===
06463  fa95
06464  fa95  a6 15      getact ldx lindx        logical index, channel number
06465  fa97  b5 49             lda buf0,x       channel buffer table 1
06466  fa99  10 02             bpl ga1
```

```
line   addr  object     source code

06467  fa9b  b5 51            lda buf1,x       channel buffer table 2
06468  fa9d  29 bf      ga1   and #$bf         clear bit 6 (strip the dirty bit)
06469  fa9f  60              rts
06470  faa0
06471  faa0
06472  faa0  ===> Set the inactive buffer's number <===
06473  faa0
06474  faa0  a6 15     gaflgs ldx lindx        logical index, channel number
06475  faa2  8e 49 43         stx lbused       last buffer used
06476  faa5  b5 49            lda buf0,x       channel buffer table 1
06477  faa7  10 09            bpl ga3          if bit 3 not set,this buffer is
                                                active, else
06478  faa9  8a               txa
06479  faaa  18               clc
06480  faab  69 08            adc #mxchns      add 8 to channel number
06481  faad  8d 49 43         sta lbused       last buffer used
06482  fab0  b5 51            lda buf1,x       channel buffer table 2
06483  fab2  85 05     ga3    sta tl
06484  fab4  29 1f            and #$1f         clear bits 5, 6 and 7
06485  fab6  24 05            bit tl           test bits 6 and 7
06486  fab8  60               rts
06487  fab9
06488  fab9
06489  fab9  ===> Set up next relative record <===
06490  fab9
06491  fab9  a9 60     nxtrec lda #getflg+ovrflo     overflow flag
06492  fabb  20 a5 f8         jsr clrflg       clear bit 5 in ($90,x)
06493  fabe  a9 80            lda #lrf         last record flag
06494  fac0  20 ae f8         jsr tstflg       bit 7 in ($90,x) set?
06495  fac3  d0 41            bne nxtr40       yes
06496  fac5  a6 15            ldx lindx        current channel number
06497  fac7  f6 59            inc recl,x       go to next record#
06498  fac9  d0 02            bne nxtr15
06499  facb  f6 61            inc rech,x       block count hi, channel 0-7
06500  facd  a6 15     nxtr15 ldx lindx        logical index, channel number
06501  facf  b5 69            lda nr,x         next record pointers table
06502  fad1  f0 2e            beq nstr45
06503  fad3  20 e1 f0         jsr getpnt       Get the active buffer pointer
06504  fad6  a6 15            ldx lindx        logical index, channel number
06505  fad8  d5 69            cmp nr,x         next record pointers table
06506  fada  90 03            bcc nxtr20
06507  fadc  20 25 fb         jsr nrbuf        Set up next record in buffer
06508  fadf  a6 15     nxtr20 ldx lindx        logical index, channel number
06509  fae1  b5 69            lda nr,x         next record pointers table
06510  fae3  20 c1 f0         jsr setpnt       Set up pointer into active data
                                                buffer
06511  fae6  a1 29            lda (buftab,x)   buffer 0 pointer lo
06512  fae8  85 18            sta data         temporary data byte
06513  faea  a5 60            lda getflg+ovrflo
06514  faec  20 a5 f8         jsr clrflg       reset bit 5 in ($90,x)
06515  faef  20 ec fd         jsr addnr        Add record size and next record
                                                pointer
06516  faf2  48        nxout  pha              store it
```

```
line   addr  object     source code

06517  faf3  90 28              bcc nxtr30       gone over end of block?
06518  faf5  a9 00              lda #$00         yes
06519  faf7  20 ef f0           jsr drdbyt       Direct read of a byte (.A =
                                                 position)
06520  fafa  d0 21              bne nxtr30       track link <> 0?
06521  fafc  68                 pla              write pointer
06522  fafd  c9 02              cmp #$02         =2?
06523  faff  f0 12              beq nxtr50
06524  fb01  a9 80       nstr45 lda #1rf         set bit 7
06525  fb03  20 9f f8           jsr setflg       Set flag
06526  fb06  20 b0 ed    nxtr40 jsr getpre       Set buffer pointers
06527  fb09  b5 29              lda buftab,x     get the pointer, use as
06528  fb0b  99 bd 00           sta lstchr,y     end pointer
06529  fb0e  a9 0d              lda #cr          throw in a carriage return
06530  fb10  85 18              sta data         temporary data byte
06531  fb12  60                 rts
06532  fb13
06533  fb13  20 1e fb    nxtr50 jsr nxtr35
06534  fb16  a6 15              ldx lindx        logical index, channel number
06535  fb18  a9 00              lda #$00
06536  fb1a  95 69              sta nr,x         next record pointers table
06537  fb1c  60                 rts
06538  fb1d
06539  fb1d  68          nxtr30 pla
06540  fb1e  a6 15       nxtr35 ldx lindx        logical index, channel number
06541  fb20  95 69              sta nr,x         next record pointers table
06542  fb22  4c 53 fc           jmp setlst       Set pointer to last character in
                                                 record
06542  fb25
06543  fb25                     .lib nrbuf
```

```
line   addr  object     source code

06545  fb25  ===> Set up next record in buffer <===
06546  fb25
06547  fb25  20 54 ee  nrbuf  jsr setdrn       Set drive number
06548  fb28  20 97 f9         jsr rdlnk        Set TRACK & SECTOR from link in
                                                buffer
06549  fb2b  20 a0 fa         jsr gaflgs       Get active buffer and set LBUSED
06550  fb2e  50 16            bvc nrbu50       V clear means buffer not dirty
                                                (changed) so no need to write
06551  fb30  20 60 f9         jsr wrtout       dirty -- write out
06552  fb33  20 d6 eb         jsr dblbuf       Toggle buffer
06553  fb36  a9 02            lda #$02         point to first data byte in the new
                                                sector
06554  fb38  20 c1 f0         jsr setpnt       Set up pointer into active data
                                                buffer
06555  fb3b  20 b3 f8         jsr tstwrt       Test if this is a write job
06556  fb3e  d0 24            bne nrbu20       not a write job
06557  fb40  20 59 f9         jsr rdab         read in needed buffer, then
06558  fb43  4c 87 ec         jmp watjob       wait around until done
06559  fb46
06560  fb46  20 d6 eb  nrbu50 jsr dblbuf       Toggle active buffer
06561  fb49  20 b3 f8         jsr tstwrt       Test if this is a write job
06562  fb4c  d0 06            bne nrbu70       not a write job
06563  fb4e  20 59 f9         jsr rdab         read in needed buffer
06564  fb51  20 87 ec         jsr watjob       Wait until job is completed
06565  fb54  20 97 f9  nrbu70 jsr rdlnk        Set TRACK & SECTOR from link in
                                                buffer
06566  fb57  a5 13            lda track        current track number
06567  fb59  f0 09            beq nrbu20       if track link =0 then it's last
                                                block
06568  fb5b  20 d6 eb         jsr dblbuf       start read job on the
06569  fb5e  20 59 f9         jsr rdab         inactive buffer
06570  fb61  20 d6 eb         jsr dblbuf       Toggle
06571  fb64  60        nrbu20 rts
06572  fb65
06573  fb65
06574  fb65  ===> Put relative record into buffer <===
06575  fb65
06576  fb65  20 eb fb  relput jsr sdirty       Set buffer dirty flag
06577  fb68  20 95 fa         jsr getact       Get active buffer number
06578  fb6b  0a               asl a            multiply buffer number by two
06579  fb6c  aa               tax              pointer in BP table
06580  fb6d  a5 18            lda data         temporary data byte
06581  fb6f  81 29            sta (buftab,x)   buffer 0 pointer lo
06582  fb71  b4 29            ldy buftab,x     increment the pointer
06583  fb73  c8               iny              if new pointer value is NOT 1,
06584  fb74  d0 09            bne relp06
06585  fb76  a4 15            ldy lindx        log. index/channel number
06586  fb78  b9 69 00         lda nr,y         next record pointers table
06587  fb7b  f0 0a            beq relp07
06588  fb7d  a0 02            ldy #$02         point to first data byte
06589  fb7f  98        relp06 tya
06590  fb80  a4 15            ldy lindx        log. index/channel number
06591  fb82  d9 69 00         cmp nr,y         next record pointers table
```

```
line   addr  object     source code

06592  fb85  d0 05            bne relp10       if NR <> pointer then NR isn't a
                                                pointer so set a new one
06593  fb87  a9 20      relp07 lda #ovrflo      set bit 5 (overflow flag) in ($90,x)
06594  fb89  4c b0 ed         jmp getpre       Set buffer pointers
06595  fb8c
06596  fb8c  f6 29      relp10 inc buftab,x     write back new pointer
06597  fb8e  d0 03            bne relp20       if 0 then next buffer not needed
06598  fb90  20 25 fb         jsr nrbuf        Set up next record in buffer
06599  fb93  60        relp20 rts
06600  fb94
06601  fb94
06602  fb94  ===> Write out relative records <===
06603  fb94
06604  fb94  a9 a0      wrtrel lda #lrf+ovrflo  check all flags
06605  fb96                                    to check for last record and
                                                overflow. If clear, some flag is
                                                set
06606  fb96  20 ae f8         jsr tstflg       Test flag
06607  fb99  d0 24            bne wr50
06608  fb9b  a5 18      wr10  lda data         ready to put data
06609  fb9d  20 65 fb         jsr relput       Put relative record into buffer
06610  fba0  a5 a0      wr20  lda eoiflg       current EOI status
06611  fba2  f0 0d            beq wr40         EOI was sent
06612  fba4  60               rts
06613  fba5
06614  fba5  a9 20      wr30  lda #ovrflo      bit 5 (overflow) set?
06615  fba7  20 ae f8         jsr tstflg       Test flag
06616  fbaa  f0 05            beq wr40         if Z set, no record overlow
06617  fbac  a9 51            lda #recovf      overflow in record error -- flag:
06618  fbae  8d 73 43         sta erword       set error for end of print
06619  fbb1  20 d9 fb  wr40   jsr clrec        Clear rest of relative record
06620  fbb4  20 39 fc         jsr rd40
06621  fbb7  ad 73 43         lda erword       error word for recovery
06622  fbba  f0 0b            beq wr51         if no errors
06623  fbbc  4c c9 db         jmp cmderr       Command level error handling
06624  fbbf
06625  fbbf  29 80      wr50  and #lrf         error flag AND last record flag not
                                                0, branch
06626  fbc1  d0 05            bne wr60         because last record flag is set, so
                                                add to file
06627  fbc3  a5 a0            lda eoiflg       EOI sent?
06628  fbc5  f0 de            beq wr30         no
06629  fbc7  60        wr51   rts
06630  fbc8
06631  fbc8  a5 18      wr60  lda data         temporary data byte
06632  fbca  48               pha
06633  fbcb  20 04 fe         jsr addrel       Add blocks to a relative file
06634  fbce  68               pla              add to the relatve file
06635  fbcf  85 18            sta data         temporary data byte
06636  fbd1  a9 80            lda #lrf         last record flag
06637  fbd3  20 a5 f8         jsr clrflg       Clear flag
06638  fbd6  4c 9b fb         jmp wr10
06639  fbd9
```

```
line   addr  object      source code

06640  fbd9
06641  fbd9  ===> Clear rest of relative record <===
06642  fbd9
06643  fbd9  a9 20       clrec  lda #ovrflo      overflow flag
06644  fbdb  20 ae f8           jsr tstflg       Test flag
06645  fbde  d0 0a              bne clr10        if Z not set, there is overflow
06646  fbe0  a9 00              lda #$00         null byte
06647  fbe2  85 18              sta data         temporary data byte
06648  fbe4  20 65 fb           jsr relput       Put relative record into buffer
06649  fbe7  4c d9 fb           jmp clrec        Clear rest of relative record
06650  fbea
06651  fbea  60          clr10  rts
06652  fbeb
06653  fbeb
06654  fbeb  ===> Set buffer dirty flag <===
06655  fbeb
06656  fbeb  20 a0 fa    sdirty jsr gaflgs       Get active buffer and set LBUSED
06657  fbee  09 40              ora #getflg      set dirty flag bit 6
06658  fbf0  ae 49 43           ldx lbused       last buffer used
06659  fbf3  95 49              sta buf0,x
06660  fbf5  60                 rts
06661  fbf6
06662  fbf6
06663  fbf6  ===> Clear buffer dirty flag <===
06664  fbf6
06665  fbf6  20 a0 fa    cdirty jsr gaflgs       Get active buffer and set LBUSED
06666  fbf9  29 bf              and #$bf         clear dirty flag bit 6
06667  fbfb  ae 49 43           ldx lbused       last buffer used
06668  fbfe  95 49              sta buf0,x
06669  fc00  60                 rts
06670  fc01
06671  fc01
06672  fc01  ===> Read relative records <===
06673  fc01
06674  fc01  a9 80       rdrel  lda #lrf         last record flag
06675  fc03  20 ae f8           jsr tstflg       Test flag
06676  fc06  d0 3c              bne rd05         no record error
06677  fc08  a9 40       rd10   lda #getflg
06678  fc0a  20 9f f8           jsr setflg
06679  fc0d  20 b0 ed           jsr getpre       Set buffer pointers
06680  fc10  b5 29              lda buftab,x     buffer 0 pointer lo
06681  fc12  d9 bd 00           cmp lstchr,y     channel last character pointer
06682  fc15  f0 22              beq rd40         if = end pointer. Since we want the
                                                 next record, set one up
06683  fc17  f6 29              inc buftab,x     buffer 0 pointer lo
06684  fc19  d0 06              bne rd20         BP = 0? no need for next buffer
06685  fc1b  20 25 fb           jsr nrbuf        Set up next record in buffer
06686  fc1e  20 b0 ed    rd15   jsr getpre       Set buffer pointers
06687  fc21  a1 29       rd20   lda (buftab,x)   buffer 0 pointer lo
06688  fc23  99 b5 00    rd25   sta chndat,y     channel data byte
06689  fc26  a9 89              lda #rndrdy      R/W random access ready, reset EOI.
                                                 Store as channel status
06690  fc28  99 98 00           sta chnrdy,y     write, read, eoi flags, channel
                                                 status
```

```
line   addr  object      source code

06691  fc2b  b5 29             lda buftab,x      buffer 0 pointer lo
06692  fc2d  d9 bd 00          cmp lstchr,y      channel last character pointer
06693  fc30  f0 01             beq rd30          if BP = end pointer
06694  fc32  60                rts
06695  fc33
06696  fc33  a9 81       rd30  lda #rndeoi       EOI flag
06697  fc35  99 98 00          sta chnrdy,y      write, read, eoi flags, channel
                                                 status
06698  fc38  60                rts
06699  fc39
06700  fc39  20 b9 fa    rd40  jsr nxtrec        Set up next relative record
06701  fc3c  20 b0 ed          jsr getpre        Set buffer pointers
06702  fc3f  a5 18             lda data          temporary data byte
06703  fc41  4c 23 fc          jmp rd25
06704  fc44
06705  fc44  a6 15       rd05  ldx lindx         no record character set up
06706  fc46  a9 0d             lda #cr           CR
06707  fc48  95 b5             sta chndat,x      channel data byte
06708  fc4a  a9 81             lda #rndeoi       set R/W, EOI
06709  fc4c  95 98             sta chnrdy,x      write, read, eoi flags, channel
                                                 status
06710  fc4e  a9 50             lda #norec        record not present error
06711  fc50  20 c9 db          jsr cmderr        Command level error handling
06712  fc53
06713  fc53
06714  fc53  ===> Set pointer to last character in record <===
06715  fc53
06716  fc53  a6 15    setlst ldx lindx         log. index/channel number
06717  fc55  b5 69             lda nr,x          next record pointers table
06718  fc57  85 1a             sta r1
06719  fc59  c6 1a             dec r1
06720  fc5b  c9 02             cmp #$02          pointer to first data byte in the
                                                 sector
06721  fc5d  d0 04             bne set101
06722  fc5f  a9 ff             lda #$ff          point to last byte in
06723  fc61  85 1a             sta r1            record
06724  fc63  b5 71    set101 lda rs,x          copy record size
06725  fc65  85 1b             sta r2
06726  fc67  20 e1 f0          jsr getpnt        Get the active buffer pointer
06727  fc6a  a6 15             ldx lindx         log. index/channel number
06728  fc6c  c5 1a             cmp r1
06729  fc6e  90 18             bcc set110        BP >= end pointer in R1? Then find
                                                 last non-0 character
06730  fc70  f0 16             beq set110
06731  fc72  20 d6 eb          jsr dblbuf        Double buffer: switch
                                                 active/inactive buffers
06732  fc75  20 95 fc          jsr fndlst        Find last non-zero character in
                                                 record
06733  fc78  90 07             bcc set105
06734  fc7a  a6 15             ldx lindx         log. index/channel number
06735  fc7c  95 bd             sta lstchr,x      channel last character pointer
06736  fc7e  4c d6 eb          jmp dblbuf        Double buffer: switch
                                                 active/inactive buffers
```

```
line   addr  object     source code

06737  fc81
06738  fc81  20 d6 eb   set105 jsr dblbuf       Double buffer: switch
                                                 active/inactive buffers
06739  fc84  a9 ff             lda #$ff
06740  fc86  85 1a             sta r1
06741  fc88  20 95 fc   set110 jsr fndlst       Find last non-zero character in
                                                 record
06742  fc8b  b0 03             bcs set140
06743  fc8d  20 e1 f0           jsr getpnt      Get the active buffer pointer
06744  fc90  a6 15       set140 ldx lindx       log. index/channel number
06745  fc92  95 bd              sta lstchr,x    channel last character pointer
06746  fc94  60                 rts
06747  fc95
06748  fc95
06749  fc95  ===> Find last non-zero character in record <===
06750  fc95
06751  fc95  20 2b f9   fndlst jsr set00        Set up pointer to active buffer
06752  fc98  a4 1a              ldy r1
06753  fc9a  b1 27       fnd110 lda (dirbuf),y  directory buffer pointer
06754  fc9c  d0 0d              bne fnd120      byte in buffer = 0?
06755  fc9e  88                 dey             pointer =
06756  fc9f  c0 02              cpy #$02        less than or equal to 2?
06757  fca1  90 04              bcc fnd130      then there isn't one, since start of
                                                 record not in here
06758  fca3  c6 1b              dec r2
06759  fca5  d0 f3              bne fnd110      if not counted down to 0 yet
06760  fca7  c6 1b       fnd130 dec r2
06761  fca9  18                 clc             to indicate record not found
06762  fcaa  60                 rts
06763  fcab
06764  fcab  98          fnd120 tya             last non-zero character found
06765  fcac  38                 sec             so set carry flag and
06766  fcad  60                 rts
06767  fcae
06768  fcae
06769  fcae  ===> Set side sector & BUFTAB to end of last record <===
06770  fcae
06771  fcae  20 d4 f9   ssend  jsr ssset        Set side sector pointers to 0
06772  fcb1  85 83              sta ssnum       side sector number
06773  fcb3  a9 04              lda #$04        points to first side sector's track
                                                 number
06774  fcb5  85 27              sta dirbuf      directory buffer pointer
06775  fcb7  a0 0a              ldy #ssioff-6   offset to last side sector track
06776  fcb9  d0 04              bne se20        always
06777  fcbb                                     test if last or previous side sector
                                                 tracks have been written (= 0)
06778  fcbb  88          se10   dey
06779  fcbc  88                 dey
06780  fcbd  30 26              bmi break       if less than 0
06781  fcbf  b1 27       se20   lda (dirbuf),y  look for last side sector #
06782  fcc1  f0 f8              beq se10        last ss# =0? Not yet found
06783  fcc3  98                 tya             offset for last side sector track
06784  fcc4  4a                 lsr a           number of last side sector equal to
                                                 current
```

```
line   addr  object     source code

06785  fcc5  c5 83            cmp ssnum         side sector number
06786  fcc7  f0 09            beq se30          yes -- last side sector here, so
                                                 store #
06787  fcc9  85 83            sta ssnum         side sector number
06788  fccb  a6 15            ldx lindx         log. index/channel number
06789  fccd  b5 79            lda ss,x          side sectors table
06790  fccf  20 1d fa         jsr ibrd          read last side sector
06791  fcd2  a0 00      se30   ldy #$00          set side sector index
06792  fcd4  84 27            sty dirbuf        directory buffer pointer
06793  fcd6  b1 27            lda (dirbuf),y    debug
06794  fcd8  d0 0b            bne break         last side sector track link must be
                                                 0
06795  fcda  c8               iny
06796  fcdb  b1 27            lda (dirbuf),y    directory buffer pointer
06797  fcdd  a8               tay               back up to track
06798  fcde  88               dey
06799  fcdf  84 84            sty ssind         (end) pointer in side sector
06800  fce1  98               tya               end pointer -1
06801  fce2  4c eb f9         jmp setssp        Use side sector pointer to set
                                                 DIRBUF & BUFTAB
06802  fce5
06803  fce5  ===> Indicate system TRACK OR SECTOR error <===
06804  fce5
06805  fce5  a9 67      break lda #$67          illegal track or sector error
06806  fce7  20 5c d9         jsr cmder2
06807  fcea
06808  fcea
06809  fcea  ===> RECORD command -- position pointer to given record <===
06810  fcea
06811  fcea  20 b6 dc   record jsr cmdset       Initialize cmd tables & pointers
06812  fced  ad 01 43         lda cmdbuf+1
06813  fcf0  85 16            sta sa            current secondary address
06814  fcf2  20 6e ed         jsr fndrch        Find the assigned read channel
06815  fcf5  90 05            bcc r20           if clear, channel found
06816  fcf7  a9 70            lda #nochnl       no channel error
06817  fcf9  20 c9 db         jsr cmderr        Command level error handling
06818  fcfc
06819  fcfc  a9 e0      r20   lda #lrf+getflg+ovrflo for bit 6 & 7
06820  fcfe  20 a5 f8         jsr clrflg        Clear flag
06821  fd01  20 a6 ed         jsr typfil        Get current file type
06822  fd04  f0 05            beq r30           relative?
06823  fd06  a9 64            lda #mistyp       file type mismatch error
06824  fd08  20 c9 db         jsr cmderr        Command level error handling
06825  fd0b
06826  fd0b  b5 90      r30   lda filtyp,x      file type flags, channel 0-7
06827  fd0d  29 01            and #$01          mask off non-drive bits, result is
06828  fd0f  85 12            sta drvnum        current drive number
06829  fd11  ad 02 43         lda cmdbuf+2      3rd command byte
06830  fd14  95 59            sta recl,x        = record number lo
06831  fd16  ad 03 43         lda cmdbuf+3      4th byte
06832  fd19  95 61            sta rech,x        = record number hi
06833  fd1b  a6 15            ldx lindx         clear CNRDY to RNRDY
06834  fd1d  a9 89            lda #rndrdy       random access ready
```

```
line   addr  object      source code

06835  fd1f  95 98               sta chnrdy,x     write, read, eoi flags, channel
                                                  status
06836  fd21  ad 04 43            lda cmdbuf+4     offset 5th byte; byte pointer into
                                                  record
06837  fd24  f0 10               beq r40          = 0?
06838  fd26  38                  sec              byte number
06839  fd27  e9 01               sbc #$01         -1 (because of CR separator)
06840  fd29  f0 0b               beq r40          if 0
06841  fd2b  d5 71               cmp rs,x         record sizes table
06842  fd2d  90 07               bcc r40          if < record size
06843  fd2f  a9 51               lda #recovf      OVERFLOW IN RECORD error
06844  fd31  8d 73 43            sta erword       error word for recovery
06845  fd34  a9 00               lda #$00         position in record
06846  fd36  85 82        r40    sta recptr       set offset
06847  fd38  20 b3 ea            jsr fndrel       calculate side sector stuff
06848  fd3b  20 fa f9            jsr sspos        set side sector pointers
06849  fd3e  50 08               bvc r50          data block present?
06850  fd40  a9 80               lda #lrf         last record flag, bit 7
06851  fd42  20 9f f8            jsr setflg       Set last record flag
06852  fd45  4c 44 fc            jmp rd05
06853  fd48
06854  fd48  20 58 fd     r50    jsr positn       Position to record
06855  fd4b  a9 80               lda #lrf         last record flag
06856  fd4d  20 ae f8            jsr tstflg       Test flag
06857  fd50  f0 03               beq r60          not set
06858  fd52  4c 44 fc            jmp rd05
06859  fd55
06860  fd55  4c 9f db     r60    jmp endcmd       That's all
06861  fd58
06862  fd58
06863  fd58  ===> Position to record <===
06864  fd58
06865  fd58  20 7a fd     positn jsr posbuf       Position proper data blocks into
                                                  buffers
06866  fd5b  a5 85               lda relptr       relative file pointer to track
06867  fd5d  20 c1 f0            jsr setpnt       Set up pointer into active data
                                                  buffer
06868  fd60  a6 15               ldx lindx        log. index/channel number
06869  fd62  b5 71               lda rs,x         record sizes table
06870  fd64  38                  sec              calculate the offset
06871  fd65  e5 82               sbc recptr       - position in record
06872  fd67  b0 03               bcs p2           if offset > 0 -- else we're in
                                                  trouble:
06873  fd69  4c e5 fc            jmp break        should not be needed
06874  fd6c
06875  fd6c  18           p2     clc
06876  fd6d  65 85               adc relptr       add REL data block pointer
06877  fd6f  90 03               bcc p30          < 256? y>
06878  fd71  69 01               adc #$01         + 2 (carry = 1!)
06879  fd73  38                  sec
06880  fd74  20 f2 fa     p30    jsr nxout
06881  fd77  4c 1e fc            jmp rd15
06882  fd7a
```

```
line   addr  object     source code

06883  fd7a
06884  fd7a  ===> Position proper data blocks into buffers <===
06885  fd7a
06886  fd7a  a5 27      posbuf lda dirbuf       directory buffer pointer
06887  fd7c  85 1c             sta r3
06888  fd7e  a5 28             lda dirbuf+1     current buffer pointer hi
06889  fd80  85 1d             sta r4
06890  fd82  20 b8 fd          jsr bhere        Check if desired block is in buffer
06891  fd85  f0 22             beq p20          link bytes = T/S from header buffer?
06892  fd87  20 f1 f8          jsr scrub        Clean buffer
06893  fd8a  20 0c f9          jsr getlnk       Set TRACK & SECTOR from link in
                                                buffer
06894  fd8d  a5 13             lda track        current track number
06895  fd8f  f0 19             beq p80          track link = 0 (final block)?
06896  fd91  20 d6 eb          jsr dblbuf       try inactive buffer
06897  fd94  20 b8 fd          jsr bhere        is this the one?
06898  fd97  d0 11             bne p80          no
06899  fd99  20 0c f9          jsr getlnk       Set TRACK & SECTOR from link in
                                                buffer
06900  fd9c  a5 13             lda track        current track number
06901  fd9e  f0 09             beq p20          final block?
06902  fda0  20 d6 eb          jsr dblbuf       Double buffer: switch
                                                active/inactive buffers
06903  fda3  20 59 f9          jsr rdab         read in next buffer
06904  fda6  20 d6 eb          jsr dblbuf       Toggle buffer
06905  fda9  60         p20     rts
06906  fdaa
06907  fdaa  a0 00      p80     ldy #$00         ger proper block
06908  fdac  b1 1c             lda (r3),y       track link
06909  fdae  85 13             sta track        current track number
06910  fdb0  c8                iny
06911  fdb1  b1 1c             lda (r3),y       sector link
06912  fdb3  85 14             sta sector       current sector number
06913  fdb5  4c 22 ed          jmp strdbl       get next block as well, then RTS
06914  fdb8
06915  fdb8
06916  fdb8  ===> Check if desired block is in buffer <===
06917  fdb8
06918  fdb8  20 3e f9   bhere  jsr gethdr       get the header
06919  fdbb  a0 00             ldy #$00
06920  fdbd  b1 1c             lda (r3),y       track link
06921  fdbf  c5 13             cmp track        current track number
06922  fdc1  f0 01             beq bh10         test sector as well
06923  fdc3  60                rts              Z=0
06924  fdc4  c8         bh10   iny
06925  fdc5  b1 1c             lda (r3),y
06926  fdc7  c5 14             cmp sector       sector link
06927  fdc9  60                rts              Z=1 if they are equal
06927  fdca
06928  fdca                    .lib nulbuf
```

```
line   addr  object      source code

06930  fdca  ===> Set null records in active buffer <===
06931  fdca
06932  fdca  20 2b f9   nulbuf jsr set00        Set indirect pointer
06933  fdcd  a0 02             ldy #$02
06934  fdcf  a9 00             lda #$00         zeroize buffer from 3rd byte onwards
06935  fdd1  91 27       nb20  sta (dirbuf),y   directory buffer pointer
06936  fdd3  c8                iny
06937  fdd4  d0 fb             bne nb20
06938  fdd6  20 ec fd          jsr addnr        (AC) points to next record
06939  fdd9  95 69       nb25  sta nr,x         next record pointers table
06940  fddb  a8                tay
06941  fddc  a9 ff             lda #$ff         for 1st byte in next record
06942  fdde  91 27             sta (dirbuf),y   init record with CR
06943  fde0  20 ec fd          jsr addnr        Add record size and next record
                                                pointer
06944  fde3  90 f4             bcc nb25         not done
06945  fde5  d0 04             bne nb30         last byte is end of record? n>
06946  fde7  a9 00             lda #$00         flag last record
06947  fde9  95 69             sta nr,x         next record pointers table
06948  fdeb  60          nb30  rts
06949  fdec
06950  fdec
06951  fdec  ===> Add record size and next record pointer <===
06952  fdec        on exit: C=1 if buffer boundary crossed
06953  fdec
06954  fdec  a6 15       addnr ldx lindx        log. index/channel number
06955  fdee  b5 69             lda nr,x         next record pointers table
06956  fdf0  38                sec              after loading next record pointer
06957  fdf1  f0 0d             beq an05
06958  fdf3  18                clc              then add record size; test if < or =
                                                256
06959  fdf4  75 71             adc rs,x         record sizes table
06960  fdf6  90 0b             bcc an10
06961  fdf8  d0 06             bne an05
06962  fdfa  a9 02             lda #$02         bypass link, set flags
06963  fdfc  2c e3 d2          bit er00         Error flag variables for use by BIT
06964  fdff  60                rts
06965  fe00
06966  fe00  69 01       an05  adc #$01         adjust for link & set carry flag
06967  fe02  38                sec
06968  fe03  60          an10  rts
06969  fe04
06970  fe04
06971  fe04  ===> Add blocks to a relative file <===
06972  fe04
06973  fe04  20 54 ee  addrel jsr setdrn        Set drive number
06974  fe07  20 ae fc          jsr ssend        Set side sector & BUFTAB to end of
                                                last record
06975  fe0a  20 7a fd          jsr posbuf       Position proper data blocks into
                                                buffers
06976  fe0d  a5 84             lda ssind        save side sector index
06977  fe0f  85 1a             sta r1
06978  fe11  a5 83             lda ssnum        save side sector number
```

```
line   addr  object      source code

06979  fe13  85 19             sta r0          temporary result
06980  fe15  a9 00             lda #$00        clear flag for one block
06981  fe17  85 1b             sta r2
06982  fe19  a9 00             lda #$00        clear RECPTR for calculations
06983  fe1b  85 82             sta recptr      pointer to start of record
06984  fe1d  20 b3 ea          jsr fndrel      calculate side sector pointers
06985  fe20  20 34 db          jsr numfre      Get number of blocks free in DRVNUM
06986  fe23  a4 15             ldy lindx       log. index/channel number
06987  fe25  b6 71             ldx rs,y        record sizes table
06988  fe27  ca                dex             reduce size
06989  fe28  8a                txa
06990  fe29  18                clc             add record pointer to record size:
06991  fe2a  65 85             adc relptr      relative file pointer to track
06992  fe2c  90 0c             bcc ar10        no carry: no span to next block
06993  fe2e  e6 84             inc ssind       (end) pointer in side sector
06994  fe30  e6 84             inc ssind       (end) pointer in side sector
06995  fe32  d0 06             bne ar10
06996  fe34  e6 83             inc ssnum       side sector number
06997  fe36  a9 10             lda #ssioff     side sector offset (will start new
                                                block)
06998  fe38  85 84             sta ssind       (end) pointer in side sector
06999  fe3a
07000  fe3a  a5 1a      ar10   lda r1
07001  fe3c  18                clc
07002  fe3d  69 02             adc #$02
07003  fe3f  20 eb f9          jsr setssp      Use side sector pointer to set
                                                DIRBUF & BUFTAB
07004  fe42  a5 83             lda ssnum       side sector number
07005  fe44  c9 06             cmp #nssl       number of side sector links
07006  fe46  90 05             bcc ar25        if valid (less than or equal to 6)
07007  fe48  a9 52      ar20   lda #bigfil     file too large error
07008  fe4a  20 c9 db          jsr cmderr      Command level error handling
07009  fe4d  a5 84      ar25   lda ssind       calculate number of blocks needed
                                                and check
07010  fe4f  38                sec             against available
07011  fe50  e5 1a             sbc r1
07012  fe52  b0 03             bcs ar30        if result is positive.
07013  fe54  e9 0f             sbc #$0f        side sector index offset -1
07014  fe56  18                clc             and store number of s-s indices into
07015  fe57  85 07      ar30   sta t3          # of side sector indices
07016  fe59  a5 83             lda ssnum       side sector number
07017  fe5b  e5 19             sbc r0          to find number of s-s needed, store
                                                into
07018  fe5d  85 08             sta t4
07019  fe5f  a2 00             ldx #$00        set up results accumulator area
07020  fe61  86 05             stx t1
07021  fe63  86 06             stx t2
07022  fe65  aa                tax             .X= side sector number
07023  fe66  20 53 fa          jsr sscalc      calculate number of blocks needed
07024  fe69  a5 06             lda t2          hi byte of number we need. If not 0,
07025  fe6b  d0 07             bne ar35
07026  fe6d  a6 05             ldx t1          if lo byte
07027  fe6f  ca                dex             minus 1
```

```
line   addr  object     source code

07028  fe70  d0 02           bne ar35        is 0, branch, else
07029  fe72  e6 1b           inc r2          check if enough blocks left: compare
                                             with
07030  fe74  cd 78 43  ar35   cmp nbtemp+1    hi byte of NBTEMP+1 then branch if
07031  fe77  90 09           bcc ar40        more than enough
07032  fe79  d0 cd           bne ar20        not enough. Else if just enough,
                                             better check lo
07033  fe7b  ad 77 43        lda nbtemp      number of blocks free (lo byte 0/1)
07034  fe7e  c5 05           cmp t1
07035  fe80  90 c6           bcc ar20        if not enough, abort
07036  fe82  a9 01     ar40  lda #$01        sector link
07037  fe84  20 ef f0        jsr drdbyt      look at sector link   (.A =
                                             position)
07038  fe87  18              clc
07039  fe88  69 01           adc #$01        gives NR
07040  fe8a  a6 15           ldx lindx       log. index/channel number
07041  fe8c  95 69           sta nr,x        next record pointers table
07042  fe8e  20 b0 d6        jsr nxtts       Next available track and sector
07043  fe91  20 fd f8        jsr setlnk      Put TRACK & SECTOR into header
07044  fe94  a5 1b           lda r2
07045  fe96  d0 15           bne ar50        if add-1-block flag is set
07046  fe98  20 60 f9        jsr wrtout      write current last record
07047  fe9b  20 d6 eb  ar45  jsr dblbuf      Toggle buffers
07048  fe9e  20 94 ec        jsr sethdr      Set up header for active buffer
07049  fea1  20 b0 d6        jsr nxtts       Get another
07050  fea4  20 fd f8        jsr setlnk      Set up link
07051  fea7  20 ca fd        jsr nulbuf      Clean it out
07052  feaa  4c b9 fe        jmp ar55
07053  fead
07054  fead  20 d6 eb  ar50  jsr dblbuf      Switch buffers
07055  feb0  20 94 ec        jsr sethdr      Set up header from T/S
07056  feb3  20 ca fd        jsr nulbuf      Clean buffer
07057  feb6  20 19 f9        jsr nullnk      Set track link to 0, sector link to
                                             last non-0 char in buf
07058  feb9  20 60 f9  ar55  jsr wrtout      write buffer
07059  febc  20 0c f9        jsr getlnk      get track & Sector from link
07060  febf  a5 13           lda track
07061  fec1  48              pha             save them
07062  fec2  a5 14           lda sector
07063  fec4  48              pha
07064  fec5  20 3e f9        jsr gethdr      now get track and sector from header
07065  fec8  a5 14           lda sector      current sector number
07066  feca  48              pha
07067  fecb  a5 13           lda track       current track number
07068  fecd  48              pha
07069  fece  20 47 fa        jsr gsspnt      Get side sector pointers
07070  fed1  aa              tax
07071  fed2  d0 0a           bne ar60
07072  fed4  20 33 ff        jsr newss       Create a new side sector, change the
                                             old to reflect it
07073  fed7  a9 10           lda #ssioff
07074  fed9  20 eb f9        jsr setssp      Use side sector pointer to set
                                             DIRBUF & BUFTAB
```

```
line   addr  object      source code

07075  fedc  e6 19              inc r0          advance side sector count
07076  fede  68          ar60   pla             track
07077  fedf  20 95 f8           jsr putss       put byte into side sector
07078  fee2  68                 pla             sector
07079  fee3  20 95 f8           jsr putss       put byte into side sector
07080  fee6  68                 pla             sector link
07081  fee7  85 14              sta sector      current sector number
07082  fee9  68                 pla             track link
07083  feea  85 13              sta track       current track number
07084  feec  f0 0f              beq ar65        if 0, there are no more blocks in
                                                 this file
07085  feee  a5 19              lda r0          count
07086  fef0  c5 83              cmp ssnum       side sector number
07087  fef2  d0 a7              bne ar45        if unequal, not enough blocks done
                                                 yet
07088  fef4  20 47 fa           jsr gsspnt      Get side sector pointers
07089  fef7  c5 84              cmp ssind       (end) pointer in side sector
07090  fef9  90 a0              bcc ar45        if SSIND>.A, we're almost done
07091  fefb  f0 b0              beq ar50        if equal, one more block left
                                                    else the job is done
07092  fefd  20 47 fa   ar65    jsr gsspnt      Get side sector pointers
07093  ff00  48                 pha
07094  ff01  a9 00              lda #$00
07095  ff03  20 de f9           jsr ssdir       Use side sector pointer to set
                                                 DIRBUF
07096  ff06  a9 00              lda #$00        zero .A and .Y: zero track link of
                                                 s-s sector
07097  ff08  a8                 tay
07098  ff09  91 27              sta (dirbuf),y  directory buffer pointer
07099  ff0b  c8                 iny
07100  ff0c  68                 pla             pull this sector's pointer,
07101  ff0d  38                 sec             subtract 1, and store the result as
                                                 the
07102  ff0e  e9 01              sbc #$01        sector link of the s-s sector in
07103  ff10  91 27              sta (dirbuf),y  directory buffer pointer
07104  ff12  20 6e f9           jsr wrtss       and write the side sector
07105  ff15  20 87 ec           jsr watjob      Wait until job is completed
07106  ff18  20 55 f6           jsr mapout      Write out BAM to drive specified in
                                                 LSTJOB
07107  ff1b  20 b3 ea           jsr fndrel      Find relative file
07108  ff1e  20 d6 eb           jsr dblbuf      Get back toleading buffer
07109  ff21  20 fa f9           jsr sspos       Use SSNUM & SSIND to set SS & BUFTAB
07110  ff24  70 03              bvs ar70        if V set, record is still beyond end
                                                 of relative file
07111  ff26  4c 58 fd           jmp positn      Position to record
07112  ff29
07113  ff29  a9 80      ar70    lda #lrf        beyond end, so signal last record
                                                 flag
07114  ff2b  20 b0 ed           jsr getpre      Set buffer pointers
07115  ff2e  a9 50              lda #norec      record not present error
07116  ff30  20 c9 db           jsr cmderr      Command level error handling
07117  ff33
07118  ff33
```

```
line   addr  object     source code

07119  ff33  ===> Create a new side sector, change the old to reflect it <===
07120  ff33
07121  ff33  20 b0 d6   newss  jsr nxtts        Next track and sector based on
                                                header
07122  ff36  20 d6 eb          jsr dblbuf       use inactive bufefr
07123  ff39  20 f1 f8          jsr scrub        Write out buffer if dirty
07124  ff3c  20 95 fa          jsr getact       Get active buffer number
07125  ff3f  48                 pha              active buffer number
07126  ff40  20 c3 f9          jsr clrbuf       Clear buffer
07127  ff43  a6 15              ldx lindx        log. index/channel number
07128  ff45  b5 79              lda ss,x         set registers for transfer
07129  ff47  a8                 tay              buffer number with side sectors
07130  ff48  68                 pla
07131  ff49  aa                 tax
07132  ff4a  a9 10              lda #ssioff      number of characters
07133  ff4c  20 a7 f9           jsr b0tob0       Move bytes between buffers
07134  ff4f  a9 00              lda #$00         point to start of the old side
                                                sector buffer
07135  ff51  20 de f9           jsr ssdir        Use side sector pointer to set
                                                DIRBUF
07136  ff54  a0 02              ldy #$02
07137  ff56  b1 27              lda (dirbuf),y   get side sector number
07138  ff58  48                 pha
07139  ff59  a9 00              lda #$00
07140  ff5b  20 c1 f0           jsr setpnt       Set up pointer into active data
                                                buffer
07141  ff5e  68                 pla              side sector number
07142  ff5f  18                 clc
07143  ff60  69 01              adc #$01         add 1, then store in the new
07144  ff62  91 27              sta (dirbuf),y   side sector
07145  ff64  0a                 asl a            multiply by 2
07146  ff65  69 04              adc #$04
07147  ff67  85 1c              sta r3           save position
07148  ff69  a8                 tay
07149  ff6a  38                 sec
07150  ff6b  e9 02              sbc #$02         subtract 2
07151  ff6d  85 1d              sta r4
07152  ff6f  a5 13              lda track        current track number
07153  ff71  85 1a              sta r1           save for side sector update
07154  ff73  91 27              sta (dirbuf),y   put track in side sector
07155  ff75  c8                 iny
07156  ff76  a5 14              lda sector       current sector number
07157  ff78  85 1b              sta r2           save for side sector update
07158  ff7a  91 27              sta (dirbuf),y   put sector in side sector
07159  ff7c  a0 00              ldy #$00         to set track link at start of new
                                                side sector at 0
07160  ff7e  98                 tya
07161  ff7f  91 27              sta (dirbuf),y   null link
07162  ff81  c8                 iny
07163  ff82  a9 11              lda #ssioff+1    indicating that the last non-zero
                                                character
07164  ff84                                      in the buffer is the one following
                                                the side sector offset
```

```
line   addr  object     source code

07165  ff84  91 27            sta (dirbuf),y   directory buffer pointer
07166  ff86  a9 10            lda #$10         side sector offset
07167  ff88  20 c1 f0         jsr setpnt       Set up pointer into active data
                                                buffer
07168  ff8b  20 52 f9         jsr wrtab        Do read and write jobs
07169  ff8e  20 87 ec         jsr watjob       Wait until job is completed
07170  ff91
07171  ff91  Finished creating new block. Now revise old one to reflect the new.
07172  ff91
07173  ff91  a6 15       ns20  ldx lindx       log. index/channel number
07174  ff93  b5 79             lda ss,x         get side sector buffer number
07175  ff95  48                pha
07176  ff96  20 a0 fa         jsr gaflgs       Get active buffer and set LBUSED
07177  ff99  a6 15             ldx lindx        log. index/channel number
07178  ff9b  95 79             sta ss,x         swap active buffer and side sector
07179  ff9d  68                pla
07180  ff9e  ae 49 43          ldx lbused       last buffer used
07181  ffa1  95 49             sta buf0,x
07182  ffa3  a9 00             lda #$00         for start of buffer
07183  ffa5  20 c1 f0          jsr setpnt       Set link to new side sector
07184  ffa8  a0 00             ldy #$00         to set track link, point to new side
                                                sector block
07185  ffaa  a5 13             lda track        current track number
07186  ffac  91 27             sta (dirbuf),y   directory buffer pointer
07187  ffae  c8                iny
07188  ffaf  a5 14             lda sector       current sector number
07189  ffb1  91 27             sta (dirbuf),y   directory buffer pointer
07190  ffb3  4c c3 ff          jmp ns50
07191  ffb6
07192  ffb6  20 95 fa    ns40  jsr getact       Get active buffer number
07193  ffb9  a6 15             ldx lindx        log. index/channel number
07194  ffbb  20 1d fa          jsr ibrd         read next side sictor
07195  ffbe  a9 00             lda #$00
07196  ffc0  20 c1 f0          jsr setpnt       pointer = 0
07197  ffc3  c6 1d       ns50  dec r4
07198  ffc5  c6 1d             dec r4
07199  ffc7  a4 1c             ldy r3           get new side sector link pointer
07200  ffc9  a5 1a             lda r1
07201  ffcb  91 27             sta (dirbuf),y   put track in
07202  ffcd  c8                iny
07203  ffce  a5 1b             lda r2
07204  ffd0  91 27             sta (dirbuf),y   put sector in
07205  ffd2  20 60 f9          jsr wrtout       write it back
07206  ffd5  20 87 ec          jsr watjob       Wait until job is completed
07207  ffd8  a4 1d             ldy r4
07208  ffda  c0 03             cpy #3           if Y >3 there are more side sectors
                                                to update
07209  ffdc  b0 d8             bcs ns40
07210  ffde  4c d6 eb          jmp dblbuf       Double buffer: switch
                                                active/inactive buffers
07211  ffe1
07212  ffe1  6c f0 10    nmi   jmp (vnmi)       $10f0
07213  ffe4
```

```
line   addr  object     source code

07215  ffe4  ===> copyright notice <===
07216  ffe4
07217  ffe4  43 42 4d   cbmdos .byte 'cbm80',$ae
07218  ffe7  38 30 ae



07220  ffea                    * = $ffe9



07222  ffe9  00         fchksm .byte 0  f-rom checksum









07224  ffea  05 ea      ublock .word ublkrd     U1
07225  ffec  3d ea             .word ublkwt     U2
07226  ffee  00 13             .word $1300      U3
07227  fff0  03 13             .word $1303      U4
07228  fff2  06 13             .word $1306      U5
07229  fff4  09 13             .word $1309      U6
07230  fff6  0c 13             .word $130c      U7
07231  fff8  0f 13             .word $130f      U8
07232  fffa
07233  fffa                    *=$fffa



07235  fffa  ===> NNMI ROM routine ($ffe1) <===
07236  fffa
07237  fffa  e1 ff             .word nmi        NMI
07238  fffc  2b d3             .word dskint     start/reset
07239  fffe  0b d5             .word atnirq     IRQ - UJ, U:
07239  0000
07240  0000                    .end

errors in pass 1 = 00000
errors in pass 2 = 00000

assembly completed
```

```
label    address  line numbers

acc200 = $eb8b:  3992,  4036
accum  =   $22:   212,  3908,  3915,  3927,  3930,  3932,  3934,  3936,  3939
                 3941,  3969,  3972,  3979,  3980,  3985,  3988,  3990,  3998
                 4000,  4001,  4003,  4005,  4006,  4007,  4009,  4018,  4036
                 4037,  4038,  4047
accx2  = $eb8a:  3947,  3994,  4030,  4035
accx4  = $eb87:  3997,  4030
actjob = $10a0:   172
ad10   = $dd23:  1928,  1937
add100 = $eb95:  4046,  4050
addfil = $f1a9:  3181,  5036,  5413
addnr  = $fdec:  6401,  6824,  6829,  6840
addrel = $fe04:  6519,  6859
addres = $eb92:  3946,  3993,  4044
addrts = $fa67:  6303,  6305
addt12 = $fa5e:  6293,  6298,  6300
adrsed =   $10:   191,   790,   798,   807
af08   = $f1d9:  5055,  5061
af10   = $f1ea:  5062,  5069
af15   = $f1f4:  5059,  5073
af20   = $f201:  5060,  5064,  5067,  5072,  5078
af25   = $f210:  5083,  5085
af30   = $f232:  5100,  5103
af50   = $f252:  5106,  5116
again  = $ec7f:  4161,  4192
ah10   = $e946:  3670,  3685
ah20   = $e965:  3672,  3674,  3686
ah30   = $e96b:  3689,  3694
ah35   = $e972:  3693,  3696,  3699
ah40   = $e97f:  3691,  3700
alldrs = $dd1e:  1926,  2748,  2993,  3258,  5514
an05   = $fe00:  6843,  6847,  6852
an10   = $fe03:  6846,  6854
ap30   = $f4f7:  5456,  5459
apmode =   $02:    63,  5351,  5368
append = $f4df:  3165,  5370,  5448,  5451
ar10   = $fe3a:  6878,  6881,  6886
ar20   = $fe48:  6893,  6918,  6921
ar25   = $fe4d:  6892,  6895
ar30   = $fe57:  6898,  6901
ar35   = $fe74:  6911,  6914,  6916
ar40   = $fe82:  6917,  6922
ar45   = $fe9b:  6933,  6973,  6976
ar50   = $fead:  6931,  6940,  6977
ar55   = $feb9:  6938,  6944
ar60   = $fede:  6957,  6962
ar65   = $fefd:  6970,  6978
ar70   = $ff29:  6996,  6999
aschex = $e93c:  3655,  3665
atn10  = $d51e:   756,   822
atn20  = $d526:   759,   761
atn30  = $d52f:   760,   763
atn40  = $d5a2:   783,   791,   799,   808,   817,   820,   821
```

```
label    address  line numbers

atn50  = $d5aa:   762,   824
atn60  = $d5bd:   825,   833
atn70  = $d5cd:   837,   840
atna   =   $01:   140,   756,   826,   833
atni   =   $80:   147
atnirq = $d50b:   748,  7124
atnnd  = $0284:   156
atnne  = $0286:   158
atnpd  = $0285:   157
atnpe  = $0287:   159,   684,   750
autofg = $10f3:   176,  2117
autoit = $ddc8:  2045,  2061,  2119,  2126
av1    = $d7a0:  1093,  1097
av2    = $d7b1:  1020,  1065,  1102,  3727
av3    = $d7b3:  1101,  1103,  1107
av4    = $d7bc:  1100,  1108
avail  = $d795:  1019,  1064,  1086,  3726
avck   = $d7bd:  1098,  1113,  5677
avck3  = $d7c3:  1116,  1118
avck4  = $d7c4:  1117,  1119,  1122
avck5  = $d7c9:  1115,  1120
avck6  = $d7da:  1124,  1128
b02    = $f9bb:  6182,  6185
b0tob0 = $f9a7:  6171,  7019
ba10   = $e9a3:  3726,  3743
ba20   = $e9a6:  3727,  3734
ba30   = $e9ca:  3741,  3745
ba35   = $e9cc:  3746,  3750
ba40   = $e9d1:  3728,  3749
badbch =   $05:    99
badblk =   $0a:   103
badbyt =   $10:   105
badcmd =   $31:  1196,  1658,  3485,  3592
badfn  =   $33:  1198,  2662,  5289
badhch =   $09:   102
badid  =   $0b:   104
badsyn =   $30:  1195,  1808,  3007,  3018,  3595,  5202
badts  =   $66:  1209,  4975
bam0   = $4100:    55,  4955
baml   = $4200:    56,  4957
bamjob =   $0c:    54,    57,    57,  2701,  3538,  3866,  5687
bcd2   = $d9ca:  1414,  1416
bcddec = $d9c1:  1409,  1435
bcjmp  = $e8fe:  3614,  3616,  3627
bctab  = $e8f8:  3602,  3623,  3625
be05   = $ea4e:  3835
be10   = $ea5d:  3839,  3842
bfcnt  =   $0c:    53,    54,   274,   276,   595,   619,   621,   720
bh10   = $fdc4:  6808,  6810
bhere  = $fdb8:  6776,  6783,  6804
bigfil =   $52:  1202,  6893
bkotst = $ea95:  3757,  3804,  3826,  3881
blk10  = $e8c1:  3592,  3606
```

```
label    address  line numbers

blk30  = $e8c6:  3595,  3599,  3660
blk40  = $e8cb:  3591,  3598
blk50  = $e8d3:  3602,  3605
blk60  = $e8dd:  3603,  3607
blkalc = $e999:  3627,  3722
blkexc = $ea49:  3631,  3833
blkfre = $e990:  3628,  3716
blknb  = $ela9:  2531,  2598,  2608,  2644
blknbl = $elad:  2600,  2602
blkpar = $e90a:  3610,  3637,  3793,  3825
blkptr = $ea60:  3632,  3847
blkrd  = $e9fc:  3629,  3782
blkrd2 = $e9dc:  3757,  3770,  3833
blkrd3 = $e9e8:  3770,  3782,  3794
blktst = $ea98:  3716,  3722,  3882
blkwt  = $eal9:  3630,  3804
block  = $e8b6:   350,   354,  3587
bmask  = $ebce:  1472,  4059,  4085,  4091
bmpnt  =   $02:   184,   959,   961,   966,  1043,  1051,  1079,  1081,  1090
                 1094,  1120,  1471,  1473,  1475,  1477,  2689,  2696,  2710
                 2717,  2843,  2860,  3395,  3398,  3410,  3421,  3725,  4058
                 4060,  4062,  4064,  4084
bp05   = $e917:  3536,  3641,  3643,  3652
bp10   = $e926:  3645,  3647,  3650,  3659
bp20   = $e92d:  3649,  3655
break  = $fce5:  6666,  6680,  6691,  6759
bt05   = $eaab:  3742,  3891
bt15   = $ea83:  3868,  3873
bt20   = $ea88:  3867,  3871
buf0   =   $49:   220,   614,   620,   622,  3564,  3574,  4097,  4099,  4521
                 4531,  4572,  4577,  5841,  5846,  6351,  6362,  6545,  6554,  7067
buf1   =   $51:   221,   615,  4100,  4102,  4522,  4536,  4581,  4586,  5843
                 5851,  5853,  6353,  6368
bufind = $f0ff:   591,  3837,  4908,  4917,  5718,  6175,  6177,  6192,  6218
bufs   = $1100:   177,   513,  2952,  2954,  2956
buftab =   $29:   218,   219,   589,   592,   598,   601,   604,   607,  1532
                 1533,  1553,  1569,  1570,  1578,  2430,  2432,  2706,  3579
                 3764,  3852,  4242,  4243,  4322,  4403,  4405,  4409,  4411
                 4413,  4414,  4484,  4733,  4737,  4738,  4739,  4741,  4781
                 4794,  4795,  4797,  4873,  4876,  4896,  4898,  5527,  5850
                 5857,  6077,  6232,  6234,  6397,  6413,  6467,  6468,  6482
                 6566,  6569,  6573,  6577
buftst = $ea75:  3847,  3860,  3881
bufuse = $433e:   258,   634,   636,  3549,  3552,  3555,  3556,  3558,  3559
                 4606,  4607,  4623,  4624,  4636,  4637
bump   =   $c0:    88,   680,   682,  4170
bw10   = $ea27:  3809,  3811
bw20   = $ea32:  3817
catid1 = $de0d:  2073,  2076,  2081
catid2 = $ddd8:  2052,  2055
catid3 = $dddb:  2049,  2054
catid4 = $ddee:  2051,  2063
cb     =   $45:   219,  1277,  1286,  1294,  1352,  1418,  1432,  1434,  1437
```

```
label    address  line numbers

                  1445,  1450,  1459,  1650,  1681,  1831,  1868,  1894,  1982
                  2017,  3477,  3479,  4772
cb10   = $f9ce:   6197,  6199
cbmdos = $ffe4:   7103
cbmv2  =   $73:    670,  1213,  5009
cbptr  =   $1c:     57,   218,   219,  5527
cc10   = $dfc4:   2228,  2316,  2323
cc15   = $dfca:   2317,  2320
cc20   = $dfd4:   2322,  2324
cdelay = $d393:    519,   520,   524
cdirty = $fbf6:   6149,  6551
char   = $437b:    285,  1651,  1654,  1828,  1833
chkin  = $e6c3:   3149,  3293,  3311
chkio  = $e6e0:   3176,  3267,  3311
chndat =   $b5:    245,   924,  1440,  3575,  3796,  4699,  4740,  4750,  4785
                  5521,  5831,  6574,  6593
chnrdy =   $98:    241,   630,   632,   850,   918,   927,  1461,  1557,  1606
                  3245,  3573,  3776,  4427,  4713,  4720,  4724,  4745,  4783
                  5462,  5833,  5883,  6576,  6583,  6595,  6721
cjumph = $d2b7:    353,   357,  1668
cjumpl = $d2ac:    349,  1666
ck10   = $e6cd:   3298,  3303
ck20   = $e6df:   3300,  3306
ck25   = $e6e3:   3312,  3319
ck30   = $e6ef:   3314,  3318
ckm1   = $f4c7:   5432,  5435
ckm2   = $f4d2:   5433,  5437
ckt1   = $f4d4:   5438,  5441
ckt2   = $f4de:   5439,  5443
cktm   = $f4bf:   5223,  5229,  5429
cld2   = $ef74:   4663,  4671,  4673
cldchn = $ef54:   2861,  4274,  4658
close  = $f58d:    818,  5533
clr10  = $fbea:   6531,  6537
clrb2  = $dbc2:   1693,  1695
clrbuf = $f9c3:   5899,  6191,  7012
clrc1  = $ef4c:   4650,  4652
clrcb  = $dbbe:   1344,  1686,  1691
clrchn = $ef48:   4648
clrec  = $fbd9:   6505,  6529,  6535
clrf10 = $f8ab:   5962,  5970
clrflg = $f8a5:   5955,  5967,  6378,  6400,  6523,  6706
cls05  = $f599:   5538,  5545
cls10  = $f59c:   5534,  5540
cls20  = $f5b0:   5553,  5555
clsall = $f5ac:   5541,  5551
clsc28 = $f5c3:   5564,  5567
clsc30 = $f5dd:   5571,  5578
clsc31 = $f5e0:   5575,  5579
clschn = $f5ba:   3203,  5542,  5553,  5561
clsd   = $ef58:   4660,  4674
clsd4  = $f707:   5739,  5745
clsd5  = $f722:   5724,  5761
```

label address line numbers

```
label    address  line numbers

cpytrk = $e3b3:  2873,  2884,  2888
cr     =   $0d:    51,  1873,  1877,  4767,  5617,  6415,  6592
cr20   = $d3a0:   522,   530
cr30   = $d3a2:   531,   569
cs07   = $dccd:  1871,  1880
cs08   = $dcce:  1869,  1874,  1878,  1881
cs10   = $dcff:  1905,   191
ctbsiz =   $07:   110,   170,   404,   404,   664
curblk = $f93b:  2358,  2399,  4840,  5046,  5321,  5362,  6087
cy     = $e5da:  3101,  3170,  3176
cy10   = $e5ec:  3167,  3183,  3200
cy15   = $e5f9:  3189,  3193
cy20   = $e5fc:  3187,  3190
cy30   = $e60e:  3195,  3197
daco   =   $02:   141,   756,   771,   833,   863,   874,   900
data   =   $18:   199,   770,   775,   784,   801,   809,   898,  1558,  1601
                 1607,  3243,  3453,  4128,  4134,  4144,  4423,  4428,  4445
                 4451,  4768,  4776,  4784,  6398,  6416,  6466,  6494,  6517
                 6521,  6533,  6588
davi   =   $40:   146
davo   =   $10:   144,   463,   751,   928,   934,   944,  1373
dblbuf = $ebd6:  4096,  4304,  4307,  4431,  4440,  4444,  4471,  5585,  5622
                 5639,  6438,  6446,  6454,  6456,  6617,  6622,  6624,  6782
                 6788,  6790,  6933,  6940,  6994,  7008,  7096
dcde   = $d54a:   774
dcde20 = $d55e:   780,   784
dcde30 = $d56a:   788,   790,   803
dcde40 = $d56f:   786,   793
dcde50 = $d573:   795,   806
dcde60 = $d57d:   778,   800
dcde70 = $d58b:   782,   807
dcde80 = $d5a2:   819
dchksm = $d2a0:   338
dectab = $e98d:  3695,  3711
del1   = $e33c:  2814,  2818
del2   = $e323:  2807,  2820
deldir = $e345:  2769,  2825,  3366,  5272
delfil = $e31d:  2777,  2792,  2805,  5758
delind = $4398:   309,  2147,  2188,  2362,  2363,  2368,  3068,  3078,  3330
                 5057,  5070,  5077,  5078,  5125
delsec = $4397:   308,  2333,  2356,  2360,  3127,  5061,  5075,  5122
diagok = $d3dc:   574,   638,   640
dirl   = $da9f:  1536,  1548,  1610
dirl0  = $dac2:  1549,  1582
dir3   = $dad8:  1540,  1561
dirbuf =   $27:   213,  2246,  2250,  2263,  2279,  2296,  2300,  2305,  2354
                 2361,  2397,  2418,  2431,  2433,  2436,  2516,  2521,  2532
                 2566,  2616,  2618,  2620,  2625,  2730,  2733,  2737,  2740
                 2765,  2771,  2775,  2827,  3109,  3343,  3346,  3349,  3353
                 3364,  4485,  4764,  4874,  4877,  4897,  4899,  5100,  5109
                 5112,  5115,  5309,  5311,  5314,  5317,  5355,  5357,  5377
                 5380,  5383,  6039,  6042,  6049,  6052,  6061,  6068,  6078
                 6080,  6208,  6214,  6219,  6272,  6275,  6324,  6338,  6339
```

```
label    address  line numbers

                  6639,  6660,  6667,  6678,  6679,  6682,  6772,  6774,  6821
                  6828,  6984,  6989,  7023,  7030,  7040,  7044,  7047,  7051
                  7072,  7075,  7087,  7090
dirent = $436b:    277,  5129,  5320,  5403,  5700,  6017
direrr =   $71:   1024,  1069,  1125,  1211
dirlen =   $18:    108,  2514,  2519,  2523
dirlst = $4346:    262,  1556,  4753,  5536
dirsec = $4396:    307,  2377,  2401,  3056,  3090
dirtyp =   $07:     68,  3580,  3580,  4717,  5570
div100 = $eb1c:   3969,  3976
div120 = $eb16:   3910,  3966
div150 = $eb2c:   3978,  4008
div200 = $eb2e:   3979,  3983
div254 = $eb13:   3907,  3964
div300 = $eb48:   3987,  3993
div400 = $eb55:   3996,  3998
div500 = $eb64:   4002,  4004,  4006
div600 = $eb7b:   4014,  4016,  4018
div700 = $eb7d:   4012,  4019
doit   = $f19d:   4272,  4280,  5028,  5691,  5776
doit2  = $f1a0:   4817,  5029,  6280
dosver =   $02:     77,  2736
drdbyt = $f0ef:   2349,  4905,  6405,  6923
drt    = $f05d:   4806,  4812
drtrd  = $f057:   3758,  4805,  5066,  5716
drtwrt = $f05b:   2713,  2741,  3817,  3827,  4811,  5116,  5322,  5363
drvcnt = $4392:    303,  2054,  2057,  2110,  2120,  2150,  2194,  2237,  3120
drvflg = $4393:    304,  2087,  2155,  2181,  2491,  2497,  2499,  2501,  2509
                  3070,  3076
drvnum =   $12:    193,   957,  1077,  1486,  1515,  1615,  1684,  1994,  2000
                  2003,  2058,  2060,  2068,  2116,  2122,  2125,  2128,  2230
                  2293,  2611,  2666,  2669,  2699,  2726,  2836,  2891,  2930
                  2961,  3125,  3179,  3211,  3270,  3338,  3723,  3885,  4220
                  4260,  4270,  4278,  4281,  4495,  4618,  4670,  4812,  5051
                  5133,  5155,  5182,  5193,  5411,  5420,  5494,  5522,  5710
                  5775,  5790,  5840,  5877,  6147,  6268,  6270,  6714
dskcpy = $e454:    351,   355,  2977
dskful =   $72:    982,  1055,  1212
dskid  = $4340:    259,  2072,  2075,  2676,  2678,  2729,  2732,  2853,  2854
                  2855,  2856,  4223,  4225,  4285,  4287
dskinl = $d41f:    610,   612
dskin2 = $d426:    614,   618
dskint = $d32b:    456,  7123
dupl   = $e371:   2845,  2848
duplct = $e350:    349,   353,  2834
dx0000 = $e476:   2978,  2992
dx0010 = $e499:   3007,  3012
dx0020 = $e49e:   2996,  3000,  3010
echksm = $e7ae:   3430
endcmd = $db9f:   1675,  2742,  2879,  3074,  3168,  3171,  3288,  3340,  3508
                  3582,  3718,  3752,  3784,  3799,  3821,  3828,  3840,  3855
                  4258,  5165,  5371,  5424,  5546,  5556,  6746
endit  = $e520:   3054,  3074
```

```
label    address  line numbers

endrd  = $f2ad:  5163,  5184
entfnd = $4345:   261,  2167,  2204,  2221,  2267,  2335,  2387,  2489,  2782,  6012
eoiflg =   $a0:   242,   767,   895,  3186,  3247,  4146,  6496,  6513
eoii   =   $20:   145,   766,   894
eoio   =   $08:   143,   463,   751,   928,   934,   944
eoiout =   $80:   120,   125,  1605,  4426,  4773
eoisnd =   $08:   121,  3185,  3246,  4714
er0    = $d2e4:   393,  6254,  6326
er00   = $d2e3:   392,  6849
er1    = $d2e5:   394,  6249,  6344
er2    = $d2e6:   395,  6329
er3    = $d2e7:   396,  6341
erblks = $434a:   266
ermsg2 = $d9fa:  1444
err1   = $d93e:  1324,  1326
err10  = $d9ae:  1359,  1364,  1388,  1390
err2   = $d95b:  1334,  1342
errbuf = $43dc:   323,   603,   606,  1431,  1433,  1439,  1457,  4762,  4765
                 4778,  4780
errchn =   $07:    58,   622,   623,   632,  1440,  1458,  1461,  3475,  4783,  4785
errcnt = $435d:   276,  4160,  5018
errend =   $f9:  1259
errled =   $20:   152,   424,   467,   577,  1348,  1497
errmsg = $d9dd:  1338,  1343,  1430,  1682
erroff = $da4b:  1425,  1496,  1683,  4769
error  = $d925:  1311,  2052,  2970,  4185
errsa  =   $10:    59,   624,  4342
errtab = $d7eb:  1222,  1247,  1259,  1264,  1272,  1280
errtok =   $c4:  1247
errts0 = $d9d7:   671,  1427,  4771
erword = $4373:   278,  1346,  1675,  1807,  1903,  3732,  3749,  6504,  6507,  6730
exec   =   $e0:    90,  2962
exlp0  = $e4f8:  3055,  3102
flcnt  = $437d:   287,  1773,  1785,  1795,  1900,  1918,  2010,  2987,  3002
                 3094,  3098,  3182,  3299,  3656,  3701,  5221,  5225,  5483,  5496
flptr  =   $81:   229,  1898,  1928,  1931,  3860,  3861,  3882
f2cnt  = $437e:   288,  1774,  1790,  1901,  1917,  1920,  1936,  2011,  2013
                 2014,  2089,  2314,  2989,  3004,  3095,  3100,  3136,  3199
                 3297,  5212,  5484,  5497
f2ptr  = $437f:   289,  1786,  1899,  1921,  2266,  2315,  2316,  2320,  3151
                 3183,  3197,  3208,  3218,  3223,  3657,  3686,  5349,  5393,  5401
fchksm = $ffe9:  7107
ff10   = $ded6:  2178,  2192,  2202
ff15   = $debe:  2180,  2195
ff25   = $dee6:  2179,  2190,  2203
ff30   = $def5:  2205,  2209
ff40   = $df03:  2193,  2208,  2210,  2215
ffre   = $deb7:  2177,  2794,  3072
ffst   = $dec9:  2187,  2752,  3052,  5215,  5517
fil03  = $d4fc:   732,   736
fil04  = $d504:   737,   741
fil5   = $d4e1:   721,   727
fil6   = $d4eb:   722,   726
```

```
label    address  line numbers

filcnt = $439b:   312,  2347,  2352,  2388,  3062,  3084
fildat =   $8b:   239,  1907,  1934,  2097,  2114,  2211,  2231,  2292,  2295
                 2660,  2784,  2835,  3024,  3030,  3032,  3123,  3139,  3140
                 3177,  3209,  3224,  3259,  3261,  3262,  3265,  3268,  3293
                 5049,  5134,  5157,  5162,  5241,  5270,  5293,  5335,  5340
                 5409,  5482,  5493,  6013
file01 = $d4c3:   706,   719
file02 = $d4db:   709,   718
filent =   $86:   238,  1906,  2277,  3142,  3143,  3214,  3219,  3271,  5124
                 5127,  5128,  5318,  5402,  6018
filnop =   $61:  1204,  4236
filopn =   $60:  1203,  5337
filsec = $438b:   296,  1909,  2301,  2790,  3537,  3563,  3705,  3851,  3862
                 3883,  3886,  3888,  5090,  5397
filtbl = $4380:   294,  1710,  1716,  1762,  1784,  1843,  1855,  1905,  1929
                 1932,  1981,  2015,  2239,  2672,  2722,  2982,  2997,  3038
                 3049,  3058,  3088,  3108,  3128,  3129,  3283,  5094,  5208
                 5256,  5429,  5498,  5511
filtrk = $4386:   295,  1848,  1860,  1908,  2268,  2271,  2297,  2298,  2321
                 2787,  3121,  3122,  3301,  3312,  3703,  5087,  5245,  5259
                 5265,  5275,  5325,  5394,  5480,  5500
filtyp =   $90:   240,  3581,  4381,  5164,  5525,  5708,  5791,  5878,  5961
                 5969,  5970,  5977,  6009,  6266,  6712
fixit  = $e523:  3053,  3073,  3076
fl05   = $e0a8:  2451,  2462
fl10   = $e0c3:  2453,  2455,  2460,  2463
flexst =   $63:  1206,  3315,  5286
flntfd =   $62:  1205,  3304,  5328
fm2030 =   $42:    79,   401
fm2040 =   $41:    78,   400
fmt102 = $e422:  2951,  2958
fmt105 = $e443:  2964,  2965
fmt110 = $e453:  2967,  2972
fnd1   = $d778:  1065,  1068
fnd10  = $ef7d:  4679,  4683
fnd3   = $d786:  1066,  1072
fnd30  = $ef8b:  4680,  4687
fndc20 = $ed76:  4338,  4340
fndc25 = $ed7c:  4341,  4343
fndc30 = $ed88:  4346,  4351
fndfil = $deel:  2201,  2207,  2214,  2591
fnd110 = $fc9a:  6639,  6645
fnd120 = $fcab:  6640,  6650
fnd130 = $fca7:  6643,  6646
fndlmt = $e0al:  2241,  2409,  2447
fndlnx = $ef79:  4511,  4677
fndlst = $fc95:  6618,  6627,  6637
fndn0  = $d727:  1009,  1014,  1016,  1019
fndn1  = $d72a:  1020,  1030
fndn2  = $d744:  1021,  1032
fndn3  = $d723:  1017,  1031
fndn5  = $d739:  1023,  1027
fndnxt = $d6fe:   967,  1000
```

```
label    address  line numbers

fndrch = $ed6e:   915,  1371,  2487,  3482,  3872,  4336,  4695,  5305,  6087,  6700
fndrel = $eab3:  3906,  6733,  6870,  6993
fndsec = $d76c:  1044,  1052,  1058
fndwl0 = $ed9a:  4366,  4373
fndwl3 = $ed91:  4358,  4360
fndwl5 = $eda2:  4364,  4373
fndw20 = $eda4:  4365,  4374
fndwch = $ed89:   847,  1382,  4112,  4356,  5574,  5779
format = $e420:  2681,  2864,  2950
found  = $4395:   306,  2163,  2170,  2191,  2192,  2206,  2313,  2325,  3066,  3080
fre25  = $eece:  4552,  4566
frebl  = $ef3a:  4636,  4642
freb2  = $ef44:  4639,  4641
frebuf = $ef34:  4579,  4588,  4597,  4632
frechn = $eea4:  1389,  2816,  3383,  4510,  4542,  4650,  4672,  4885,  4888
                 5537,  5579,  5605
freco  = $eeab:  4544,  4549
freich = $f0d3:  1350,  1687,  4883,  5538
fremsg = $e20b:  2646,  2652,  2654
frerts = $da34:  1470,  1478
frets  = $dalc:  1467,  2805,  2818,  3717
freuse = $ebb4:  1468,  4056,  4070
fsl0   = $ddad:  2019,  2022
fsl5   = $ddb5:  2012,  2020,  2023
fslset = $dd95:  2009,  2747,  5513
ga1    = $fa9d:  6352,  6354
ga3    = $fab2:  6363,  6369
gaflgs = $faa0:  6028,  6360,  6435,  6542,  6551,  7062
gberr  = $ee8c:  4527,  4535,  5809,  5895
gbyte  = $ef95:  3242,  4695
gcbyte = $e662:  3242,  5448
gdl    = $db31:  1604,  1610
gel0   = $f02b:  4763,  4766,  4775
gel5   = $f032:  3483,  4778
ge20   = $f03b:  3480,  4777,  4782
ge30   = $f03d:  4774,  4783
get    = $efa3:   939,  4697,  4705
get0   = $efc7:  4718,  4723
get00  = $efad:  4707,  4710
getl   = $efcc:  4715,  4727
get3   = $effc:  4749,  4756
get6   = $f001:  4728,  4753
getact = $fa95:   712,  1513,  1529,  1549,  1566,  1575,  3281,  3874,  4103
                 4211,  4233,  4314,  4390,  4491,  4666,  4790,  4870,  4892
                 4906,  5092,  5644,  5713,  5983,  6074,  6088,  6145,  6150
                 6350,  6463,  7010,  7078
getbl  = $edd2:  4402,  4413
getb2  = $edce:  4407,  4410
getb5  = $ef27:  4615,  4620
getbul = $ef09:  4606,  4613
getbu2 = $ef16:  4608,  4612
getbu3 = $ef29:  4611,  4623,  4626
getbu4 = $ef19:  4614,  4627
```

```
label    address  line numbers

getbuf = $ef03:  4525,  4534,  4603,  5807,  5893
getbyt = $edb8:  1597,  3377,  3379,  4296,  4298,  4400,  4421,  4434,  4438
                 4448,  4775,  5519,  6159,  6161
getd3  = $db20:  1598,  1601
getdir = $dbla:  1597,  4755
geterc = $f00c:  4712,  4761
getflg =   $40:    41,  6377,  6399,  6543,  6563,  6705
gethdr = $f93e:   954,  5417,  5826,  5924,  6088,  6804,  6950
getlnk = $f90c:  6048,  6779,  6785,  6945
getnam = $e0c9:  1539,  2471
getpnt = $f0el:  2351,  2488,  3805,  4141,  4481,  4761,  4892,  5614,  5619
                 6389,  6612,  6629
getpre = $edb0:  3763,  3853,  4390,  4400,  4732,  6412,  6480,  6565,  6572
                 6587,  7000
getr2  = $ee64:  4502,  4508
getr4  = $eea3:  4524,  4533,  4537
getr5  = $ee94:  4526,  4531
getr55 = $ee76:  4515,  4517
getrch = $ee63:  1507,  3529,  3561,  4507,  5785
getsim = $e9e2:  3763,  3773
getwch = $ee60:  4501,  5870
gib20  = $e67b:  3248,  3250,  3253
gibyte = $e65e:  3190,  3240
gn05   = $e0f4:  2490,  2497
gn050  = $e0f9:  2492,  2499
gn051  = $e10b:  2500,  2507
gn10   = $e118:  2498,  2514
gn12   = $e125:  2518,  2520
gn14   = $e139:  2524,  2526,  2529,  2531
gn15   = $e147:  2535,  2546
gn20   = $e168:  2559,  2562
gn22   = $e170:  2566,  2571
gn30   = $e180:  2574,  2581
gn35   = $e190:  2576,  2579,  2582
gn37   = $e195:  2584,  2590
gn40   = $ela4:  2586,  2591
gn45   = $ela8:  2495,  2593
gnsub  = $e0d9:  2475,  2485
goodj  =   $01:    95
gpl    = $f0e4:  4894,  6287
gsspnt = $fa47:  6285,  6955,  6974,  6978
hdrs   = $1021:   168,  1318,  1320,  2071,  2074,  4217,  4219,  4224,  4226
                 4277,  4284,  4286,  4939,  4941,  4986,  4988,  6094,  6096
hed2ts = $f15d:  4974,  4981,  5008
hex0   = $d9b5:  1397,  1402
hex5   = $d9c0:  1398,  1404
hexdec = $d9bl:  1394,  1448,  1453
ibop   = $fa27:  6262,  6265
ibrd   = $fald:  6245,  6260,  6676,  7080
ibwt   = $fa23:  6263
id     = $1000:    48,   649
id20   = $ecel:  4253,  4258
id2030 =   $64:    76,   654
```

label address line numbers

```
                    5811,  5820,  5830,  5839,  5873,  5897,  5905,  5933,  5948
                    5960,  5967,  5976,  6063,  6140,  6215,  6243,  6285,  6350
                    6360,  6382,  6386,  6390,  6394,  6420,  6426,  6471,  6476
                    6591,  6602,  6613,  6620,  6630,  6674,  6719,  6754,  6840
                    6872,  6926,  7013,  7059,  7063,  7079
lintab =   $a2:      244,   610,   624,   626,   707,  3161,  3162,  3164,  3566
                    3568,  4345,  4361,  4517,  4550,  4556,  4661,  5466,  5562
                    5802,  5804,  5887,  5890,  5997
linuse = $4348:      264,   628,  4564,  4565,  4679,  4688,  4689
lisner =   $01:      119
listen = $d5d0:      830,   842,   910
lk05   = $de7d:     2146,  2158,  2159
lk10   = $de87:     2150,  2171
lk15   = $de8d:     2151,  2154
lk20   = $de9c:     2160,  2168
lk25   = $deal:     2149,  2162,  2169
lk26   = $deaa:     2164,  2167
lk30   = $debl:     2161,  2170
loadir = $f509:     5171,  5472
longln =   $32:     1197,  1886
lookup = $de7a:     2145,  3096,  3135,  3266
loop   = $d74e:     1039,  1054
lrf    =   $80:       40,  3191,  3251,  5449,  6379,  6410,  6490,  6511,  6522
                    6560,  6705,  6736,  6741,  6999
lsn10  = $d5d8:      845,   846
lsn15  = $d5e7:      849,   853
lsn20  = $d5ef:      858
lsn21  = $d5f5:      861,   862
lsn25  = $d603:      860,   871,   887
lsn26  = $d613:      877,   878
lsn28  = $d628:      885,   886
lsn30  = $d630:      852,   857,   891
lsn40  = $d64c:      903,   904
lsnact =   $0e:      189,   789,   793,   805,   824,  1361
lsnadr =   $0c:      187,   584,   785
lsnerr = $d999:     1362,  1382
lstbuf = $4399:      310,  2340,  2342,  2350,  3060,  3086
lstchr =   $bd:      246,  1458,  1512,  1580,  1581,  1603,  3475,  3571,  3774
                    3795,  3798,  4401,  4406,  4424,  4450,  4734,  4742,  5797
                    5862,  6414,  6567,  6578,  6621,  6631
lstdrv = $4394:      305,  1685,  1926,  1992,  5194,  5481,  5523
lstjob = $434e:      274,   714,  1516,  4168,  4177,  4192,  4493,  4619,  4668
                    4925,  5022,  5053,  5646,  5842,  5844,  5985,  6148
lstsec = $433d:      257,  1006,  1012,  1029
lxint  =   $3f:       61,   627
m30    = $e7f8:     3467,  3470,  3482
mapchk = $f671:     5662,  5681
mapout = $f655:     2815,  5578,  5644,  6992
maxl   = $d7dd:     1133,  1135
maxsa  =   $12:       49,   244,   609,  4337,  4357,  6002
maxsec = $d7db:     1005,  1132,  2870,  3409,  4970,  4999
maxtrk =   $24:       52,   974,  1045,  2876,  4943,  4997,  5680
mdmode =   $03:       64,  5332
```

```
label    address  line numbers

mem    = $e7af:   350,   354,  3435
memerr = $e7fe:  3437,  3449,  3485
memrd  = $e7d4:  3447,  3452
memwrt = $e803:  3445,  3488,  3492
mer5   = $d8ef:  1265,  1271,  1273
mer6   = $d8f7:  1275,  1281
mer65  = $d8fe:  1279,  1302
mer7   = $d911:  1285,  1292
mer70  = $d910:  1276,  1291
mistyp =   $64:  1207,  3157,  5345,  6709
mol0   = $f65c:  3339,  5647
mode   = $439d:   314,  5218,  5234,  5250,  5263,  5331,  5350,  5367,  5436
modlst = $d2cb:   384,   385,  5434
movbl  = $db0e:  1588,  1592
movbuf = $db0c:  1528,  1545,  1565,  1587
moverr = $d8e4:  1264,  1268,  1299,  1443
movlpl = $e4dc:  3043,  3048
movlp2 = $e4eb:  2990,  3050
mrkl   = $e76a:  3382,  3385
mrk2   = $e754:  3375,  3387
mrkbam = $e74b:  3337,  3355,  3360,  3372
msgl   = $elff:  2646,  2649
msgfre = $elfa:  2493,  2503,  2644
msglen =   $0c:  2645,  2654
mul100 = $eaf5:  3944,  3949
mul200 = $eafc:  3945,  3947
mul25  = $eae6:  3933,  3936
mul400 = $eb12:  3954,  3956,  3958
mul50  = $eaf1:  3935,  3940,  3942
mulply = $ead1:  3906,  3926
mxchns =   $08:    47,    58,    60,   220,   221,   223,   225,   226,   227
                  228,   240,   241,   245,   246,   277,   613,  6366
mxfils =   $05:   106,   238,   239,   294,   295,   296,  1851,  1904
n101   = $e223:  2661,  2665
n108   = $e24f:  2674,  2687
n110   = $e25e:  2682,  2691,  2694
n111   = $e261:  2696,  2698
nambuf = $41b4:   321,  1588,  2537,  2550,  2553,  2556,  2561,  2563,  2567
                 2573,  2577,  2583,  2588,  2589,  2600,  2629,  2633,  2635
                 2636,  2638,  2647
nb20   = $fdd1:  6821,  6823
nb25   = $fdd9:  6825,  6830
nb30   = $fdeb:  6831,  6834
nbcmds =   $06:  3600,  3625
nbkh   =   $61:   224,  4328,  5602,  5612,  5626,  5770,  5860
nbkl   =   $59:   222,  4326,  5600,  5611,  5624,  5627,  5767,  5859
nbsiz  =   $1b:   109,  1591,  2598
nbtemp = $4377:   282,  1524,  1541,  1543,  1561,  1563,  1634,  1635,  2508
                 2517,  2522,  2612,  2614,  6916,  6919
ncmds  =   $0b:   347,   357,  1652
nd15   = $eld9:  2625,  2631
nd20   = $elel:  2622,  2627,  2629
ndaci  =   $40:   153
```

```
label    address  line numbers

new    = $e217:   351,   355,  2659
newdir = $e1b4:  2510,  2608,  5516
newmap = $e776:  2714,  3393
newmpv = $e773:  3328,  3392
newss  = $ff33:  6958,  7007
nm10   = $e783:  3402,  3427
nm20   = $e795:  3413,  3418
nm30   = $e79f:  3420,  3425
nmi    = $ffe1:  7098,  7122
nmiflg = $10f2:   175,   694
nmodes =   $04:   385,  5431
noblk  =   $65:  1208,  3746
nochnl =   $70:  1210,  3532,  3868,  4528,  4684,  6702
nodblk =   $04:    98
nofile =   $34:  1199,  1757
nohdr  =   $02:    96
norec  =   $50:  1200,  5390,  6596,  7001
nosync =   $03:    97
notlk  = $d66b:   916,   920
notrdy =   $00:   117,  4723
notyet = $ec85:  4157,  4194
nr     =   $69:   226,  5822,  5937,  6064,  6387,  6391,  6395,  6422,  6427
                 6472,  6477,  6603,  6825,  6833,  6841,  6927
nrbu20 = $fb64:  6442,  6453,  6457
nrbu50 = $fb46:  6436,  6446
nrbu70 = $fb54:  6448,  6451
nrbuf  = $fb25:  6393,  6433,  6484,  6571
nrfdi  =   $80:   154
ns20   = $ff91:  7059
ns40   = $ffb6:  7078,  7095
ns50   = $ffc3:  7076,  7083
nssl   =   $06:    44,    45,    45,  6333,  6891
nssp   =   $78:    46,  6292
nstr45 = $fb01:  6388,  6410
ntypes =   $05:   388,  2018,  5437
nulbuf = $fdca:  5938,  6818,  6937,  6942
nullnk = $f919:  5939,  6059,  6943
numfl  = $db42:  1622,  1633
numf2  = $db48:  1624,  1626,  1631
numfre = $db34:  1615,  2650,  6871
numsec = $1099:   170,   661,  1136
nxdbl  = $f0b4:  4860,  4861
nxdrbk = $f083:  4840,  5073
nxout  = $faf2:  6402,  6766
nxtl   = $d6c2:   962,   975,   981,   986,   992
nxt2   = $d6ec:   971,   985
nxtbl  = $f056:  4796,  4800
nxtbuf = $f044:  2394,  2819,  3386,  4790
nxtds  = $d6b7:   957,  4847
nxterr = $d6e7:   970,   982,   993
nxtr15 = $facd:  6384,  6386
nxtr20 = $fadf:  6392,  6394
nxtr30 = $fb1d:  6403,  6406,  6425
```

label address line numbers

```
nxtr35 = $fb1e:  6419,  6426
nxtr40 = $fb06:  6381,  6412
nxtr50 = $fb13:  6409,  6419
nxtrec = $fab9:  6377,  6586
nxtts  = $d6b0:   954,  4463,  5900,  6928,  6935,  7007
ob05   = $e845:  3532,  3539,  3550,  3553
ob10   = $e84a:  3527,  3535
ob15   = $e85d:  3544,  3547
ob30   = $e88e:  3530,  3565
ok     = $ec7d:  4159,  4163,  4189
okerr  = $d9d2:  1425,  1641
onedrv = $dd10:  1917,  2659,  5213
op021  = $f2b5:  5146,  5148,  5167
op04   = $f2c1:  5170,  5176
op041  = $f2d6:  5168,  5186
op0415 = $f2dd:  5150,  5190
op042  = $f2ec:  5187,  5196
op049  = $f2f5:  5197,  5200
op10   = $f2fd:  5201,  5205
op100  = $f405:  5327,  5331
op110  = $f417:  5333,  5336,  5340
op115  = $f420:  5297,  5299,  5345,  5354
op120  = $f425:  5344,  5348
op125  = $f44b:  5352,  5366
op130  = $f47e:  5387,  5389,  5393
op20   = $f301:  5199,  5206,  5208
op40   = $f336:  5222,  5226,  5230,  5261,  5264
op45   = $f34a:  5233,  5239
op50   = $f360:  5236,  5240,  5247,  5250
op60   = $f36a:  5228,  5256
op75   = $f381:  5252,  5265
op80   = $f395:  5271,  5275
op81   = $f39f:  5268,  5277,  5281
op815  = $f3ae:  5285,  5289
op82   = $f3b3:  5283,  5293
op90   = $f3f9:  5253,  5325
open   = $f279:  1647,  5140
opf1   = $f4bc:  5416,  5424
opfin  = $f4a7:  5323,  5369,  5414
opir10 = $e65a:  3233,  3235
opirf1 = $e61e:  3152,  3184,  3208
opnblk = $e837:  3525,  5188
opnird = $f06c:  2341,  2379,  2806,  3217,  3374,  4822
opnirw = $f07c:  3180,  4833
opnrch = $f747:  4826,  5161,  5181,  5399,  5784
opntyp = $f072:  4825
opnwch = $f7e6:  4835,  5300,  5412,  5868
opread = $f45b:  3230,  5366,  5376
optsch = $de10:  2085,  2145,  2749,  3050,  5214,  5515
opwrt  = $f49b:  5273,  5278,  5409
or10   = $f764:  5795,  5798
or20   = $f780:  5808,  5811
or30   = $f7a8:  5800,  5829
```

```
label    address  line numbers

pr30   = $dca0:  1830,  1853
pr35   = $dca2:  1834,  1854
pr40   = $dcb4:  1858,  1861
prgsec = $4374:   279,  5159,  5423
prgtrk =   $11:   192,  5149,  5421
prgtyp =   $02:    70,  3155,  5156,  5156,  5190,  5190,  5237
prscln = $dbe6:  1722,  1755,  2977,  5196,  5505
prseq  = $e4a8:  2834,  2979,  3015
ps05   = $db6b:  1643,  1646,  1649
ps10   = $db75:  1653,  1657
ps20   = $db85:  1655,  1661
ps30   = $db8f:  1663,  1665
pu10   = $d348:   472,   475
pu20   = $d34e:   476,   488
pu30   = $d351:   479,   481
pupsl  = $e571:  3051,  3071,  3119
put    = $ebfd:   908,  4122
putbl  = $ecc2:  4234,  4239,  5950
putbyt = $ecb6:  1518,  1520,  1522,  1523,  1525,  1527,  1535,  1537,  1538
                 1542,  1544,  1547,  1562,  1564,  1572,  1573,  1574,  1589
                 3814,  4135,  4145,  4232,  4458,  4467,  4469,  4853,  4855
                 4860,  4862,  4864,  5085,  5088,  5091,  5618,  5634,  5636
putss  = $f895:  5911,  5913,  5915,  5917,  5919,  5921,  5926,  5928,  5947
                 6963,  6965
r0     =   $19:   206,  2751,  2793,  2796,  3733,  3892,  5719,  5721,  5722
                 5727,  5729,  5731,  5735,  5738,  5747,  5749,  5752,  5754
                 5757,  5761,  5764,  5768,  5771,  6865,  6903,  6961,  6971
r1     =   $1a:   207,  6604,  6605,  6609,  6614,  6626,  6638,  6863,  6886
                 6897,  7039,  7086
r2     =   $1b:   208,  6611,  6644,  6646,  6867,  6915,  6930,  7043,  7089
r20    = $fcfc:  6701,  6705
r3     =   $1c:   209,  6773,  6794,  6797,  6806,  6811,  7033,  7085
r30    = $fd0b:  6708,  6712
r4     =   $1d:   210,  3971,  3974,  6775,  7037,  7083,  7084,  7093
r40    = $fd36:  6723,  6726,  6728,  6732
r50    = $fd48:  6735,  6740
r60    = $fd55:  6743,  6746
ra10   = $d3a8:   539,   544,   547
ra30   = $d3b8:   549,   564
ra40   = $d3ba:   550,   562
ramtst = $d3a6:   538
rconl  = $d383:   511,   515
rd05   = $fc44:  6562,  6591,  6738,  6744
rd1    = $edeb:  4425,  4431
rd10   = $fc08:  6563
rd15   = $fc1e:  6572,  6767
rd20   = $fc21:  6570,  6573
rd25   = $fc23:  6574,  6589
rd3    = $ee12:  4422,  4446
rd30   = $fc33:  6579,  6582
rd4    = $ee13:  4436,  4448
rd40   = $fc39:  5825,  6506,  6568,  6586
rdab   = $f959:  3275,  6110,  6443,  6449,  6455,  6789
```

```
label    address  line numbers

rdbuf  = $ed46:  4294,  4306,  4309,  4443
rdbyt  = $edd7:  2809,  2811,  4421,  4748,  4798,  5829
rdin   = $f967:  6124
rdlnk  = $f997:  5452,  6157,  6434,  6451
rdmode =   $00:    65
rdrel  = $fc01:  4708,  6560
rds5   = $f977:  6133,  6139
rdss   = $f975:  5818,  6138
rdylst =   $01:   123,   124,   125,   629,  5882
rdytlk =   $88:   122,   124,   631,  1460,  1554,  4782,  5832
read   =   $80:    84,  2909,  4309,  4805,  6110,  6124,  6138,  6261
read15 = $e3ff:  2921,  2924
reads  = $e3dc:  2885,  2908
reads1 = $e3e6:  2912,  2917
reads3 = $e3f5:  2919,  2923
reads8 = $e3f3:  2915,  2918
rec    = $434b:   267,  1895,  2302,  2306,  3229,  5114,  5217,  5258,  5384
                 5385,  5388,  5805,  5891,  5916
rec1   = $ec5b:  4172,  4173
rec2   = $ec66:  4177,  4184
rec3   = $ec6c:  4179,  4180
rec4   = $ec78:  4165,  4185
rec5   = $ec7b:  4182,  4187
rech   =   $61:   225,  3931,  6385,  6718
recl   =   $59:   223,  3929,  6383,  6716
record = $fcea:   350,   354,  6697
recov  = $ec4a:  4164
recovf =   $51:  1201,  6503,  6729
recptr =   $82:   234,  3950,  6732,  6757,  6869
rel1   = $eee0:  4574,  4580
rel10  = $eec8:  4561,  4564
rel15  = $eec2:  4560,  4563
rel2   = $eef1:  4583,  4589
rel3   = $ef02:  4592,  4598
relbuf = $eecf:  4527,  4557,  4571
relp06 = $fb7f:  6470,  6475
relp07 = $fb87:  6473,  6479
relp10 = $fb8c:  6478,  6482
relp20 = $fb93:  6483,  6485
relptr =   $85:   237,  3909,  3911,  3912,  6752,  6762,  6877
relput = $fb65:  6462,  6495,  6534
reltyp =   $04:    67,  4384,  5082,  5105,  5227,  5298,  5353,  5572,  5799,  5880
rename = $e67c:   351,   355,  3258
result =   $1e:   211,  3913,  3952,  3953,  3955,  3957,  4013,  4015,  4017
                 4025,  4026,  4027,  4046,  4048
revcnt = $435c:   275,   644,  4164,  4174,  5016
rfdo   =   $04:   142,   463,   756,   763,   826,   842,   871,   882,   891,  1383
rlindx = $4376:   281
rm10   = $d362:   491,   509
rn10   = $e68b:  3263,  3265
rndeoi =   $81:   125,  4744,  6582,  6594
rndget = $efd7:  4721,  4732
rndrdy =   $89:   124,  3572,  3775,  4719,  6575,  6720
```

```
label    address  line numbers

rnget1 = $efe5:  3783,  4735,  4738
rnget2 = $efe7:  3820,  3854,  4137,  4739
rnget3 = $eff8:  4743,  4746
rom    = $d000:    38,   326
roml   = $d2f8:   405
rs     =   $71:   227,  3942,  5806,  5892,  5936,  6610,  6727,  6755,  6845,  6873
rt10   = $d36e:   498,   503
rt20   = $d370:   499,   501
sa     =   $16:   197,   796,   813,   858,  1357,  1505,  2471,  2479,  2486
                 3160,  3202,  3241,  3565,  3871,  4111,  4114,  4336,  4356
                 4513,  4542,  4549,  4649,  4651,  4659,  4660,  4673,  4710
                 4727,  4823,  4834,  4884,  4887,  5036,  5045,  5121,  5140
                 5231,  5304,  5358,  5361,  5365,  5414,  5465,  5533,  5543
                 5552,  5554,  5561,  5698,  5778,  5801,  5886,  6699
sa05   = $dd6c:  1982,  1990
sa10   = $dd7f:  1987,  1991
sa20   = $dd85:  1984,  1993
sc15   = $e2d3:  2758,  2795
sc17   = $e2f3:  2772,  2782
sc20   = $e30b:  2785,  2793
sc25   = $e30d:  2759,  2767,  2794
sc30   = $e312:  2753,  2796
scall  = $fa4e:  6292,  6295
scflg  = $f89d:  5955
schtbl = $de6b:  2107,  2137
scrl   = $f8fc:  6029,  6032
scrend = $dba9:  1680,  2800
scrtch = $e2c1:   351,   355,  2747
scrub  = $f8f1:  5584,  6028,  6778,  7009
sd20   = $dd48:  1950,  1967
sd22   = $dd49:  1951,  1964,  1966
sd24   = $dd4b:  1952,  1975
sd40   = $dd4e:  1959
sd50   = $dd5d:  1968
sdirty = $fbeb:  6043,  6462,  6542
se10   = $fcbb:  6664,  6668
se20   = $fcbf:  6662,  6667
se30   = $fcd2:  6672,  6677
search = $e043:  2160,  2201,  2357,  2364,  2370,  2386,  5071
secinc = $4344:   260,   643,  1002,  4843,  4846,  4849
secss  = $434d:   269,  5111,  5381,  5815,  5904
sector =   $14:   195,   979,   990,  1000,  1003,  1008,  1011,  1013,  1015
                 1018,  1027,  1028,  1063,  1067,  1099,  1321,  1429,  1452
                 1680,  1703,  2275,  2339,  2359,  2378,  2400,  2708,  2719
                 2776,  2791,  2812,  2871,  2872,  2887,  2898,  3216,  3274
                 3334,  3354,  3357,  3380,  3729,  3731,  3736,  3887,  4074
                 4081,  4218,  4267,  4299,  4439,  4468,  4854,  4989,  5000
                 5040,  5063,  5065,  5074,  5160,  5180,  5316,  5398,  5422
                 5453,  5651,  5703,  5742,  5755,  5796,  5816,  5903,  5920
                 5927,  6041,  6053,  6097,  6162,  6276,  6798,  6812,  6948
                 6951,  6967,  7042,  7074
sectrk = $d2ea:   400,   404,   660
seek   =   $b0:    87,  4271
```

```
label    address  line numbers

seqget = $eff9:  4731,  4748,  4754
seqtyp =   $01:    69,  5183,  5183,  5248
set00  = $f92b:  6037,  6048,  6059,  6074,  6637,  6818
setany = $dd64:  1717,  1979
setbmp = $d789:  1038,  1077,  1467,  2668,  2841,  2857,  3372,  3392
setdir = $f0e4:  4893
setdrn = $ee54:  4441,  4462,  4491,  6433,  6859
setdrv = $dd3a:  1930,  1943
seterr = $d48d:   670
setfle = $dd5d:  1948,  1972
setflg = $f89f:  3252,  5960,  6411,  6564,  6737
seth   = $ec97:  2078,  2897,  2960,  4212,  4268,  4815,  5817,  5907,  6278
sethdr = $ec94:  4211,  4293,  4305,  4442,  4472,  5871,  6144,  6934,  6941
setjob = $f116:  2913,  2933,  4932,  5030
setl01 = $fc63:  6607,  6610
setl05 = $fc81:  6619,  6624
setl10 = $fc88:  6615,  6616,  6627
setl40 = $fc90:  6628,  6630
setlds = $da35:  1483,  1995,  2129,  2157,  2183,  2667,  3893,  4113,  4255
                 4696,  5158
setljb = $f10e:  4316,  4925,  6152
setlnk = $f8fd:  6037,  6929,  6936
setlst = $fc53:  6428,  6602
setnme = $dd4e:  1946,  1960
setpnt = $f0c1:  1509,  2381,  2808,  3222,  3236,  3280,  3376,  3772,  3812
                 4433,  4465,  4474,  4779,  4828,  4851,  4859,  4869,  5079
                 5307,  5459,  5633,  5824,  5932,  6158,  6396,  6440,  6753
                 7026,  7053,  7069,  7082
setr3  = $e3cc:  2896,  2901
setr6  = $e3db:  2899,  2903
setrh  = $e3c1:  2884,  2891
sets10 = $d474:   653
sets20 = $d47a:   656
sets30 = $d47f:   652,   655,   659
sets40 = $d481:   660,   665
setsec = $d46b:   649
setssp = $f9eb:  5909,  5923,  6225,  6253,  6687,  6889,  6960
simprs = $dbd2:  1709,  3326,  4250,  5176
sj10   = $f981:  6105,  6112,  6144
sj20   = $f98d:  6119,  6126,  6149
sj30   = $f994:  6143,  6152
sjb1   = $f188:  4950,  4973,  5015
sjb2   = $f140:  4954,  4957
sjb3   = $f143:  4956,  4958
sjb4   = $f14a:  4958,  4969
sp10   = $dbe3:  1713,  1717
sr10   = $dff1:  2342,  2395
sr15   = $dff7:  2343,  2346
sr20   = $e004:  2351,  2392
sr30   = $e028:  2355,  2367
sr40   = $e055:  2389,  2394
sr50   = $e05b:  2369,  2397
srchst = $dfda:  2148,  2189,  2332,  3331,  5058
```

```
label    address  line numbers

srre   = $e031:  2177,  2375,  3361
ss     =   $79:   228,   616,  4590,  4595,  5812,  5898,  5906,  5949,  6141
                 6216,  6244,  6286,  6675,  7014,  7060,  7064
sscalc = $fa53:  5597,  6294,  6909
ssdir  = $f9de:  6206,  6214,  6226,  6981,  7021
ssend  = $fcae:  5586,  6248,  6657,  6860
ssind  =   $84:   236,  3919,  5593,  6252,  6323,  6685,  6862,  6879,  6880
                 6884,  6895,  6975
ssioff =   $10:    45,  5595,  5912,  5922,  6661,  6883,  6959,  7018,  7049
ssnum  =   $83:   235,  3914,  5587,  6321,  6332,  6658,  6671,  6673,  6864
                 6882,  6890,  6902,  6972
ssp10  = $fa0d:  6241,  6248
ssp20  = $fa14:  6242,  6247,  6252
sspos  = $f9fa:  6240,  6734,  6995
ssset  = $f9d4:  6205,  6320,  6657
sstest = $fa68:  6240,  6246,  6320
st10   = $fa79:  6325,  6329
st20   = $fa7d:  6322,  6332
st30   = $fa8d:  6334,  6341
st40   = $fa91:  6340,  6344
stdir  = $da54:  1504,  5518
strl   = $ed3a:  4301,  4304
strdbl = $ed22:  4293,  5792,  6799
strsiz = $433a:   254,  2413,  2448,  2456,  2459
strtit = $ed4c:  4310,  4313
struct = $d2bb:   366,   370,  1803
t0     =   $04:   201,  3403,  3414,  5670,  6173,  6182,  6195,  6197
t0v1   = $ddc5:  2031,  2033,  2035
t1     =   $05:   202,   705,   736,  1104,  1617,  1669,  3404,  3415,  3441
                 3478,  3542,  3545,  3551,  3557,  3615,  3667,  3677,  3680
                 3838,  4604,  4609,  4614,  5591,  5599,  5673,  6011,  6015
                 6176,  6193,  6301,  6302,  6369,  6371,  6906,  6912,  6920
t2     =   $06:   203,  1105,  2895,  2896,  2900,  2902,  2911,  2918,  2932
                 2938,  3405,  3416,  3678,  4905,  4911,  5592,  5601,  5676
                 5733,  5751,  5996,  6000,  6006,  6174,  6183,  6304,  6907,  6910
t3     =   $07:   204,   702,   706,   718,  1091,  1123,  1297,  1300,  3668
                 3698,  3702,  4909,  5596,  5667,  6178,  6296,  6901
t4     =   $08:   205,  5588,  5589,  5655,  5666,  5669,  5672,  5675,  6299,  6904
tabjmp = $d2f8:   411,   512
tagcmd = $dbef:  1664,  1755
talk   = $d660:   839,   915
talkl  = $d665:   917,   949
talker =   $80:   118
tc25   = $dbf4:  1757,  1764
tc30   = $dbf9:  1756,  1760,  2992
tc35   = $dc01:  1765,  5512
tc40   = $dc0b:  1768,  1770
tc50   = $dc29:  1776,  1782
tc60   = $dc46:  1793,  1795
tc70   = $dc4d:  1796,  1798
tc75   = $dc55:  1783,  1801
tc80   = $dc61:  1804,  1807
temp   =   $04:   185,   201,   202,   203,   204,   205,   421,   491,   518
```

label address line numbers

```
label    address  line numbers

wrprot =   $08:   101
wrt0   = $ee24:  4459,  4462,  5457
wrtab  = $f952:  6103,  7054
wrtbuf = $ed4a:  4312,  4470,  4856,  5637
wrtbyt = $ee1e:  4129,  4458
wrtcl  = $ed6b:  4325,  4327,  4329
wrtout = $f960:  2828,  3286,  5940,  6030,  6117,  6437,  6932,  6944,  7091
wrtrel = $fb94:  4132,  6490
wrtss  = $f96e:  5929,  6131,  6990
wtmode =   $01:    66,  5251,  5262
wverer =   $07:   100
wverfy =   $a0:    86
x0015  = $e4af:  3018,  3023,  3029,  3031
x0020  = $e4b4:  3017,  3021
zerres = $eb7e:  3926,  3977,  4024
```

## CONTENTS

### Disc Controller Section

```
line   addr  object     source code

00004  0000
00005  0000
00006  0000  ===> Equates for Disk Controller <===
00007  0000
00008  0000            goodj  = 1              job completed successfully
00009  0000            nohdr  = 2              header block not found
00010  0000            nosync = 3              no sync character
00011  0000            nodblk = 4              data block not found
00012  0000            badbch = 5              data block checksum error
00013  0000            wverer = 7              verify error after write
00014  0000            wrprot = 8              write protect on
00015  0000            badhch = 9              header block checksum error
00016  0000            badid  = 11             ID mismatch error
00017  0000            badfmt = 12             format error
00018  0000
00019  0000            bid    = 7              data block ID
00020  0000            hbid   = 8              header ID byte
00021  0000            tries  = 10             maximum allowable errors
00022  0000            offbyt = 15             OFF byte
00023  0000            bufpnt = $21            pointer into header buffer
00024  0000            maxtrk = 36             highest track number plus 1
00025  0000            stpcnt = $8c            steps to move to track
00026  0000            bmpfl  = $c1            flag to indicate head is stepping
00027  0000
00028  0000            readj  = $fc            read command for PCR
00029  0000            writej = $10            write command for PCR
00030  0000            verfyj = $20            do wverfy
00031  0000            seekj  = $30            do seek
00032  0000            bumpj  = $40            do bump
00033  0000            jumpj  = $50            do jumpc
00034  0000
00035  0000            readc  = $80            read a sector
00036  0000            jumpc  = $d0            jump to machine code in buffer
00037  0000            exec   = $e0            execute code in buffer when speed &
                                                head ready
00038  0000
00039  0000
00040  0000  ===> external labels < ===
00041  0000
00042  0000                   *=0
00043  0000            irqcnt *=*+1            irq counter lo
00044  0001            mtrtm  *=*+2            motor timing drive 0/1
00045  0003            drvst  *=*+2            drive status (0/1)
00046  0005            steps  *=*+2            steps to move to desired track
                                                (0-127
00047  0007            cow    *=*+1            buffer for irq/motor timing bits
00048  0008            work   *=*+1            temporary workspace
00049  0009            ctrack *=*+1            current track
00050  000a            nxtrk  *=*+1            next track to move head to
00051  000b            csectr *=*+1            highest sector to read
00052  000c            csect  *=*+1            last sector read
00053  000d            stab   *=*+5            current block header (ID1, ID2,
                                                track, sector, parity)
```

```
line   addr  object      source code

00054  0012              drive  *=*+1      current drive
00055  0013              track  *=*+1      current track
00056  0014              nexts  *=*+1      next (optimal) sector to service
00057  0015              sectr  *=*+1      highest sector number in current
                                           track
00058  0016              bufpt  *=*+2      buffer pointer lo address
00059  0018              hdrpt  *=*+2      pointer to active values in header
                                           table
00060  001a              ftnum  *=*+4      track currently being formatted
00061  001e              job    *=*+1      temporary storage of job code
00062  001f              errcnt *=*        error counter (format routine)
00063  001f              jobnum *=*+1      storage of current job number (DC)
00064  0020
00065  0020
00066  0020                     *=$40
00067  0040
00068  0040              via    *=*
00069  0040              prb    = 0        port b data register
00070  0040              pra    = 1        port a data register
00071  0040              ddrb   = 2        port b data direction register
00072  0040              t1lc1  = 4        timer 1 write latch/read counter
00073  0040              t1ch   = 5        timer 1 trigger t1c1/t1c1 transfer
00074  0040              acr    = 11       auxiliary control register
00075  0040              pcr    = 12       peripheral control register
00076  0040              ifr    = 13       interrupt flag register
00077  0040              ier    = 14       interrupt enable register
00078  0040
00079  0040
00080  0040                     *=$80
00081  0080
00082  0080              rriot  *=*        port a data register
00083  0080              orega  = 0        port a data register
00084  0080              drega  = 1        port a data direction register
00085  0080              oregb  = 2        port b data register
00086  0080              dregb  = 3        port b data direction register
00087  0080              timer  = 15       timer offset
00088  0080
00089  0080
```

```
line   addr  object     source code

00091  0080  ===> Common area defines <===
00092  0080
00093  0080                    *=$0400          seen as $1000 by Bus Controller
00094  0400
00095  0400            tick    *=*+1            ($1000) interrupt interval counter
00096  0401            delay   *=*+1            ($1001) delay constant drive motors
00097  0402            cutmt   *=*+1            ($1002) cutout constant drive motors
00098  0403            jobs    *=*+6            ($1003) job queue for buffer
00099  0409
00100  0409                    *=$0499
00101  0499            numsec  *=*+4            ($1099) number of sectors zone 1-4
00102  049d            gap1    *=*+1            ($109d) size of gap after sector
                                                header
00103  049e            gap2    *=*+1            ($109e) minimum bytes between
                                                sectors
00104  049f
00105  049f                    *=$04a0
00106  04a0            actjob  *=*+1            ($10a0) current job number
00107  04a1
00108  04a1
00109  04a1   ===> Internal labels <===
00110  04a1
00111  04a1            formt   = $0500
00112  04a1            starti  = $fc00          vector to main idling loop for disk
                                                controller
00113  04a1            donei   = $fc02          vector to disk controller job
                                                completed
00114  04a1            reset   = $fc04          Soft reset routine
00114  04a1            .end
00115  04a1
```

```
line   addr  object      source code

00117  04a1
00118  04a1
00119  04a1              *       = $fc00
00120  fc00
00121  fc00
00122  fc00  ===> vector to main idling loop for disk controller <===
00123  fc00
00124  fc00  54 fc       starti .word start       DC scanning job queue
00125  fc02
00126  fc02
00127  fc02  ===> vector to disk controller job completed <===
00128  fc02
00129  fc02  08 ff       donei  .word error       Terminate disk controller routines
00130  fc04
00131  fc04
00132  fc04  ===> Soft reset routine <===
00133  fc04
00134  fc04  ad 00 04    reset  lda tick          interrupt interval counter
00135  fc07  d0 fb               bne reset        Soft reset routine
00136  fc09
00137  fc09
00138  fc09
00139  fc09
00140  fc09
00141  fc09  ===> Power-on reset <===
00142  fc09
00143  fc09  a2 3f       resel  ldx #$3f          reset the stack
00144  fc0b  9a                 txs
00145  fc0c  d8                 cld
00146  fc0d  8e 00 04           stx tick         interrupt interval counter
00147  fc10  a9 00              lda #$00
00148  fc12  95 00       rese22 sta irqcnt,x     irq counter lo
00149  fc14  d5 00              cmp irqcnt,x     irq counter lo
00150  fc16  d0 fe       rese2  bne rese2
00151  fc18  9d 03 04           sta jobs,x       job queue for buffer 0
00152  fc1b  ca                 dex
00153  fc1c  10 f4              bpl rese22
00154  fc1e  86 40              stx via+prb      port b data register
00155  fc20  86 42              stx via+ddrb     port b data direction register
00156  fc22  8e 02 04           stx cutmt        cutout constant drive motors
00157  fc25  86 81              stx rriot+drega  port a data direction register
00158  fc27  86 1a              stx ftnum        track currently being formatted
00159  fc29  a9 07              lda #$07
00160  fc2b  85 83              sta rriot+dregb  port b data direction register
00161  fc2d  a9 fc              lda #readj
00162  fc2f  85 4c              sta via+pcr      peripheral control register
00163  fc31  a9 92              lda #$92
00164  fc33  85 4e              sta via+ier      interrupt enable register
00165  fc35  a9 01              lda #$01
00166  fc37  85 4b              sta via+acr      auxiliary control register
00167  fc39  4a                 lsr a
00168  fc3a  85 44              sta via+t1lcl    timer 1 write latch/read counter
00169  fc3c  85 16              sta bufpt        buffer pointer lo address
```

```
line   addr  object     source code

00170  fc3e  a9 0f            lda #$0f
00171  fc40  8d 00 04         sta tick          interrupt interval counter
00172  fc43  85 8f            sta rriot+timer   timer offset
00173  fc45  a9 80            lda #readc
00174  fc47  85 03            sta drvst         drive status (0/1)
00175  fc49  85 04            sta drvst+1
00176  fc4b  a9 32            lda #$32
00177  fc4d  8d 01 04         sta delay         delay constant drive motors
00178  fc50  a9 04            lda #$04
00179  fc52  85 19            sta hdrpt+1
00180  fc54
00181  fc54
00182  fc54  ===> DC scanning job queue, waiting for something to do <===
00183  fc54
00184  fc54  a0 0e       start  ldy #$0e          initialize buffer pointer to 14
00185  fc56  58          start1 cli
00186  fc57  b9 03 04           lda jobs,y        is there a job for buffer in .Y?
00187  fc5a  10 31              bpl start5        Decrement job queue pointer
00188  fc5c  c9 d0              cmp #jumpc
00189  fc5e  d0 05              bne start2        Mask out drive number, test if motor
                                                  running
00190  fc60  84 1f              sty jobnum        storage of current job number
00191  fc62  4c 20 fd           jmp ex1           Execute job in data buffer
00192  fc65
00193  fc65
00194  fc65  ===> Mask out drive number, test if motor running <===
00195  fc65
00196  fc65  29 01       start2 and #$01          mask out drive number
00197  fc67  aa                 tax
00198  fc68  85 12              sta drive         current drive
00199  fc6a  78                 sei
00200  fc6b  a5 40              lda via+prb       port b data register
00201  fc6d  3d ea ff           and andc,x        acceleration/deceleration bits drive
                                                  0/1
00202  fc70  f0 0e              beq start3        Wait for motor turned on and head
                                                  positioned (irq)
00203  fc72  a5 40              lda via+prb       port b data register
00204  fc74  5d ea ff           eor andc,x        acceleration/deceleration bits drive
                                                  0/1
00205  fc77  85 40              sta via+prb       port b data register
00206  fc79  a5 00              lda irqcnt        irq counter lo
00207  fc7b  6d 01 04           adc delay         delay constant drive motors (32)
00208  fc7e  95 01              sta mtrtm,x       motor timing drive 0/1 (about 1.5s)
00209  fc80
00210  fc80
00211  fc80  ===> Wait for motor turned on and head positioned (irq) <===
00212  fc80
00213  fc80  b5 03       start3 lda drvst,x       set startup flag
00214  fc82  30 06              bmi start4
00215  fc84  a5 00              lda irqcnt        irq counter lo
00216  fc86  95 01              sta mtrtm,x       motor timing drive 0/1
00217  fc88  b5 03              lda drvst,x       drive status (0/1)
00218  fc8a  0a          start4 asl a
```

```
line   addr  object      source code

00219  fc8b  10 05              bpl que          Loop to check job queue for step
                                                 command
00220  fc8d
00221  fc8d
00222  fc8d  ===> Decrement job queue pointer <===
00223  fc8d
00224  fc8d  88          start5 dey
00225  fc8e  10 c6              bpl start1
00226  fc90  d0 c2       start6 bne start        DC scanning job queue
00227  fc92
00228  fc92
00229  fc92  ===> Loop to check job queue <===
00230  fc92
00231  fc92  58          que    cli
00232  fc93  a9 40              lda #bumpj
00233  fc95  85 08              sta work         temporary workspace
00234  fc97  a0 0e              ldy #$0e
00235  fc99  84 1f              sty jobnum       storage of current job number
00236  fc9b
00237  fc9b
00238  fc9b  ===> Check if job for buffer (bit 7 = 1) <===
00239  fc9b
00240  fc9b  20 a5 fd    que1   jsr setjob       Set pointers in header buffer
                                                 according to job code
00241  fc9e  10 2f              bpl que2         Check other jobs, bump head
00242  fca0  29 01              and #$01         mask number to see if
00243  fca2  c5 12              cmp drive        current drive
00244  fca4  d0 29              bne que2         Check other jobs, bump head
00245  fca6  a5 1e              lda job          temporary storage of job code
00246  fca8  c9 40              cmp #bumpj
00247  fcaa  d0 03              bne que4         Calculate distance to next track
00248  fcac  4c 2a fd           jmp bump         Position head (bump to track 1)
00249  fcaf
00250  fcaf
00251  fcaf  ===> Calculate distance to next track <===
00252  fcaf
00253  fcaf  b5 03       que4   lda drvst,x      drive status (0/1)
00254  fcb1  29 3f              and #$3f
00255  fcb3  85 13              sta track        current track
00256  fcb5  f0 3d              beq gotu         Check position of track
00257  fcb7  38                 sec              calculate distance
00258  fcb8  f1 18              sbc (hdrpt),y    are we already on this track?
00259  fcba  f0 38              beq gotu         Check position of track
00260  fcbc  85 0a              sta nxtrk        number of steps to the track we want
00261  fcbe  10 05              bpl que5         check if another job is closer
00262  fcc0  18                 clc
00263  fcc1  49 ff              eor #$ff
00264  fcc3  69 01              adc #$01
00265  fcc5  c5 08       que5   cmp work         temporary workspace
00266  fcc7  b0 06              bcs que2         Check other jobs, bump head
00267  fcc9  85 08              sta work         temporary workspace
00268  fccb  a5 0a              lda nxtrk        next track to move head to
00269  fccd  85 09              sta ctrack       current track
```

```
line   addr  object     source code

00270  fccf
00271  fccf
00272  fccf  ===> Check other jobs, bump head <===
00273  fccf
00274  fccf  c6 1f      que2   dec jobnum       storage of current job number
00275  fcd1  10 c8             bpl que1         Check if job for buffer (bit 7 = 1)
00276  fcd3  a5 08             lda work         temporary workspace
00277  fcd5  24 09             bit ctrack       current track
00278  fcd7  30 05             bmi que6
00279  fcd9  18                clc
00280  fcda  49 ff             eor #$ff
00281  fcdc  69 01             adc #$01
00282  fcde  85 08      que6   sta work         temporary workspace
00283  fce0  0a                asl a
00284  fce1  78                sei
00285  fce2  95 05             sta steps,x      steps to move to desired track
                                                (0-127 in, >127 out)
00286  fce4  a9 40             lda #bumpj
00287  fce6  15 03             ora drvst,x      drive status (0/1)
00288  fce8  18                clc
00289  fce9  65 08             adc work         temporary workspace
00290  fceb  95 03             sta drvst,x      drive status (0/1)
00291  fced  d0 a1      que7   bne start6
00292  fcef
00293  fcef
00294  fcef  f3 fc 1f   tabl   .byte $f3, $fc, $1f, $19, $12
00295  fcf2  19 12
00296  fcf4
00297  fcf4
00298  fcf4  ===> Check position of track <===
00299  fcf4
00300  fcf4  b5 03      gotu   lda drvst,x      drive status (0/1)
00301  fcf6  30 f5             bmi que7
00302  fcf8  a2 04             ldx #$04
00303  fcfa  b1 18             lda (hdrpt),y    pointer to active values in header
                                                table
00304  fcfc
00305  fcfc
00306  fcfc  ===> Determine zone <===
00307  fcfc
00308  fcfc  dd ef fc   gotu1  cmp tabl,x
00309  fcff  ca                dex
00310  fd00  b0 fa             bcs gotu1        Determine zone
00311  fd02  bd 99 04          lda numsec,x     number of sectors zone 1-4
00312  fd05  85 15             sta sectr        highest sector number in current
                                                track
00313  fd07  8a                txa
00314  fd08  0a                asl a
00315  fd09  85 08             sta work         temporary workspace
00316  fd0b  a5 82             lda rriot+oregb  port b data register
00317  fd0d  29 f8             and #$f8
00318  fd0f  05 08             ora work         temporary workspace
00319  fd11  05 12             ora drive        current drive
```

```
line   addr  object      source code

00320  fd13  85 82              sta rriot+oregb  port b data register
00321  fd15  a6 12              ldx drive        current drive
00322  fd17  a5 1e              lda job          temporary storage of job code
00323  fd19  c9 e0              cmp #exec        execute code in buffer when up to
                                                 speed and head ready
00324  fd1b  f0 03              beq ex1          Execute job in data buffer
00325  fd1d  4c 82 fe           jmp seek         Read sector header and compare ID
00326  fd20
00327  fd20
00328  fd20  ===> Execute job in data buffer <===
00329  fd20
00330  fd20  a5 1f       ex1    lda jobnum       storage of current job number
00331  fd22  18                 clc
00332  fd23  69 05              adc #$05
00333  fd25  85 17              sta bufpt+1      buffer pointer hi address
00334  fd27  6c 16 00           jmp (bufpt)      buffer pointer lo address
00335  fd2a
00336  fd2a
00337  fd2a  ===> Position head (bump to track 1) <===
00338  fd2a
00339  fd2a  78          bump   sei
00340  fd2b  a9 c1              lda #bmpf1       indicate head is stepping
00341  fd2d  95 03              sta drvst,x      drive status (0/1)
00342  fd2f  a9 0f              lda #$0f
00343  fd31  3d ec ff           and andd,x       motor timing bits drive 0/1
00344  fd34  05 40              ora via+prb      port b data register
00345  fd36  85 40              sta via+prb      port b data register
00346  fd38  a9 8c              lda #stpcnt
00347  fd3a  95 05              sta steps,x      steps to move to desired track
                                                 (0-127 in, >127 out)
00348  fd3c  4c c6 fe           jmp done         DC message 1, OK (BC: 00)
00349  fd3f
00350  fd3f
00351  fd3f  ===> Optimize job (1) <===
00352  fd3f
00353  fd3f  a9 7f       fsnum1 lda #$7f
00354  fd41  85 0c              sta csect        last sector read
00355  fd43  a5 10              lda stab+3       sector number in block header
00356  fd45  18                 clc
00357  fd46  69 02              adc #$02
00358  fd48  c5 15              cmp sectr        highest sector number in current
                                                 track
00359  fd4a  90 02              bcc fsnum2       Check queue for best job to do next
00360  fd4c  a9 00              lda #$00
00361  fd4e
00362  fd4e
00363  fd4e  ===> Check queue for best job to do next <===
00364  fd4e
00365  fd4e  85 14       fsnum2 sta nexts        next (optimal) sector to service
00366  fd50  a2 0e              ldx #$0e
00367  fd52  86 1f              stx jobnum       storage of current job number
00368  fd54  a2 ff              ldx #$ff
00369  fd56
```

```
line   addr  object      source code

00370  fd56
00371  fd56  20 a5 fd   fsnum3 jsr setjob       Set pointers in header buffer
                                                according to job code
00372  fd59  10 33             bpl fsnum5
00373  fd5b  85 08             sta work          temporary workspace
00374  fd5d  b1 18             lda (hdrpt),y     pointer to active values in header
                                                table
00375  fd5f  c5 13             cmp track         current track
00376  fd61  d0 2b             bne fsnum5
00377  fd63  a5 08             lda work          temporary workspace
00378  fd65  29 01             and #$01          mask out drive number
00379  fd67  c5 12             cmp drive         current drive
00380  fd69  d0 23             bne fsnum5
00381  fd6b  98                tya
00382  fd6c  c9 e0             cmp #exec         execute code in buffer when up to
                                                speed and head ready
00383  fd6e  f0 1e             beq fsnum5
00384  fd70  c8                iny
00385  fd71  38                sec
00386  fd72  b1 18             lda (hdrpt),y     pointer to active values in header
                                                table
00387  fd74  e5 14             sbc nexts         next (optimal) sector to service
00388  fd76  10 03             bpl fsnum4
00389  fd78  18                clc
00390  fd79  65 15             adc sectr         highest sector number in current
                                                track
00391  fd7b  85 0b      fsnum4 sta csectr        highest sector to read
00392  fd7d  38                sec
00393  fd7e  e5 0c             sbc csect         last sector read
00394  fd80  10 0c             bpl fsnum5
00395  fd82  a6 1f             ldx jobnum        storage of current job number
00396  fd84  a5 0b             lda csectr        highest sector to read
00397  fd86  85 0c             sta csect         last sector read
00398  fd88  8a                txa
00399  fd89  18                clc
00400  fd8a  69 05             adc #$05
00401  fd8c  85 17             sta bufpt+1       buffer pointer hi address
00402  fd8e  c6 1f      fsnum5 dec jobnum        storage of current job number
00403  fd90  10 c4             bpl fsnum3
00404  fd92  8a                txa
00405  fd93  10 03             bpl fsnum6
00406  fd95  4c 54 fc          jmp start         DC scanning job queue
00407  fd98
00408  fd98
00409  fd98  8e a0 04   fsnum6 stx actjob        current job number
00410  fd9b  86 1f             stx jobnum        storage of current job number
00411  fd9d  20 a5 fd          jsr setjob        Set pointers in header buffer
                                                according to job code
00412  fda0  a5 1e             lda job           temporary storage of job code
00413  fda2  4c c4 fd          jmp reed          Read sector into buffer (code 00)
00414  fda5
00415  fda5
00416  fda5
```

```
line   addr  object      source code

00417  fda5  ===> Set pointers in header buffer according to job code <===
00418  fda5
00419  fda5  a4 1f       setjob ldy jobnum       storage of current job number
00420  fda7  b9 03 04           lda jobs,y       job queue for buffer 0
00421  fdaa  48                 pha
00422  fdab  29 70              and #$70
00423  fdad  85 1e              sta job          temporary storage of job code
00424  fdaf  98                 tya
00425  fdb0  0a                 asl a
00426  fdb1  0a                 asl a
00427  fdb2  0a                 asl a
00428  fdb3  69 21              adc #bufpnt
00429  fdb5  85 18              sta hdrpt        pointer to active values in header
                                                 table
00430  fdb7  a0 02              ldy #$02
00431  fdb9  68                 pla
00432  fdba  60                 rts
00433  fdbb
00434  fdbb
00435  fdbb  ===> Optimize job (2) <===
00436  fdbb
00437  fdbb  a0 03       fsnum  ldy #$03          sector offset
00438  fdbd  b1 18              lda (hdrpt),y    pointer to active values in header
                                                 table
00439  fdbf  85 10              sta stab+3       sector number in block header
00440  fdc1  4c 3f fd           jmp fsnum1       Optimize job (1)
00441  fdc4
00442  fdc4
00443  fdc4  ===> Read sector into buffer (code 00) <===
00444  fdc4
00445  fdc4  c9 00       reed   cmp #$00
00446  fdc6  d0 28              bne wprot        verify write protect
00447  fdc8  20 dd fd           jsr dstrt        Find a data block ID (07)
00448  fdcb
00449  fdcb
00450  fdcb  ===> Wait for data byte to be ready <===
00451  fdcb
00452  fdcb  24 4d       1100   bit via+ifr      interrupt flag register
00453  fdcd  10 fc              bpl 1100         Wait for data byte to be ready
00454  fdcf  a5 41              lda via+pra      port a data register
00455  fdd1  91 16              sta (bufpt),y    buffer pointer lo address
00456  fdd3  45 08              eor work         temporary workspace
00457  fdd5  85 08              sta work         temporary workspace
00458  fdd7  c8                 iny
00459  fdd8  d0 f1              bne 1100         Wait for data byte to be ready
00460  fdda  4c 6b fe           jmp 1211
00461  fddd
00462  fddd
00463  fddd  ===> Find a data block ID (07) <===
00464  fddd
00465  fddd  a0 00       dstrt  ldy #$00
00466  fddf  84 08              sty work         temporary workspace
00467  fde1  20 d3 fe           jsr srch         Find specified data block header
```

```
line   addr  object      source code

00468  fde4  20 3e ff           jsr syncl        Wait for a data block sync mark
00469  fde7  c9 07              cmp #bid
00470  fde9  f0 04              beq dstrtx       found a good block
00471  fdeb  a9 04              lda #nodblk
00472  fded  d0 0b              bne errx         error exit
00473  fdef
00474  fdef
00475  fdef  ===> Found a good block <===
00476  fdef
00477  fdef  60          dstrtx rts
00478  fdf0
00479  fdf0
00480  fdf0  ===> Verify write protect <===
00481  fdf0
00482  fdf0  c9 20       wprot  cmp #verfyj
00483  fdf2  10 63              bpl verfy        Compare sector with buffer
00484  fdf4  a5 82              lda rriot+oregb  port b data register
00485  fdf6  29 08              and #$08
00486  fdf8  f0 03              beq rite         write buffer to disk
00487  fdfa
00488  fdfa
00489  fdfa  ===> error exit <===
00490  fdfa
00491  fdfa  4c 08 ff    errx   jmp error        Terminate disk controller routines
00492  fdfd
00493  fdfd
00494  fdfd  ===> write buffer to disk <===
00495  fdfd
00496  fdfd  a9 10       rite   lda #writej
00497  fdff  85 4e              sta via+ier      interrupt enable register
00498  fe01  20 d3 fe           jsr srch         Find specified data block header
00499  fe04  ae 9d 04           ldx gapl         size of gap after sector header
00500  fe07  ca                 dex
00501  fe08
00502  fe08
00503  fe08  ===> Check OFF bytes <===
00504  fe08
00505  fe08  24 4d       1200   bit via+ifr      interrupt flag register
00506  fe0a  10 fc              bpl 1200         Check OFF bytes
00507  fe0c  24 41              bit via+pra      port a data register
00508  fe0e  ca                 dex
00509  fe0f  d0 f7              bne 1200         Check OFF bytes
00510  fe11  a9 de              lda #$de
00511  fe13  85 4c              sta via+pcr      peripheral control register
00512  fe15  a9 dc              lda #$dc
00513  fe17  a2 ff              ldx #$ff
00514  fe19  20 79 ff           jsr out          Write character in .X to disk
00515  fe1c  20 79 ff           jsr out          Write character in .X to disk
00516  fe1f  20 79 ff           jsr out          Write character in .X to disk
00517  fe22  24 4d       1202   bit via+ifr      interrupt flag register
00518  fe24  10 fc              bpl 1202
00519  fe26  24 41              bit via+pra      port a data register
00520  fe28  85 4c              sta via+pcr      peripheral control register
```

```
line   addr  object      source code

00521  fe2a  a9 07             lda #bid
00522  fe2c  85 80             sta rriot+orega  port a data register
00523  fe2e  a0 00             ldy #$00
00524  fe30  84 08             sty work        temporary workspace
00525  fe32
00526  fe32
00527  fe32  24 4d       1203  bit via+ifr     interrupt flag register
00528  fe34  10 fc             bpl 1203
00529  fe36  24 41             bit via+pra     port a data register
00530  fe38  b1 16             lda (bufpt),y   buffer pointer lo address
00531  fe3a  85 80             sta rriot+orega port a data register
00532  fe3c  45 08             eor work        temporary workspace
00533  fe3e  85 08             sta work        temporary workspace
00534  fe40  c8                iny
00535  fe41  d0 ef             bne 1203
00536  fe43  aa                tax
00537  fe44  20 79 ff          jsr out         Write character in .X to disk
00538  fe47  20 58 ff          jsr 1204        write byte to disk
00539  fe4a  a4 1f             ldy jobnum      storage of current job number
00540  fe4c  b9 03 04          lda jobs,y      job queue for buffer 0
00541  fe4f  49 30             eor #$30
00542  fe51  99 03 04          sta jobs,y      job queue for buffer 0
00543  fe54  4c bb fd          jmp fsnum       Optimize job (2)
00544  fe57
00545  fe57
00546  fe57  ===> Compare sector with buffer <===
00547  fe57
00548  fe57  20 dd fd    verfy jsr dstrt       Find a data block ID (07)
00549  fe5a  24 4d       1210  bit via+ifr     interrupt flag register
00550  fe5c  10 fc             bpl 1210
00551  fe5e  a5 41             lda via+pra     port a data register
00552  fe60  d1 16             cmp (bufpt),y   buffer pointer lo address
00553  fe62  d0 1a             bne 1212        DC error 7, verify error (BC: 25)
00554  fe64  45 08             eor work        temporary workspace
00555  fe66  85 08             sta work        temporary workspace
00556  fe68  c8                iny
00557  fe69  d0 ef             bne 1210
00558  fe6b  20 51 ff    1211  jsr byte        Read data byte following sync mark
                                               to accumulator
00559  fe6e  c5 08             cmp work        temporary workspace
00560  fe70  f0 04             beq 12111
00561  fe72  a9 05             lda #badbch
00562  fe74  d0 0a             bne 1213        always
00563  fe76  a9 10       12111 lda #$10
00564  fe78  24 4d             bit via+ifr     interrupt flag register
00565  fe7a  f0 4a             beq done        DC message 1, OK (BC: 00)
00566  fe7c  d0 02             bne 1213
00567  fe7e
00568  fe7e
00569  fe7e
```

```
line   addr  object      source code

00570  fe7e  ===> DC error 7, verify error (BC: 25) <===
00571  fe7e
00572  fe7e  a9 07       1212   lda #wverer
00573  fe80  d0 46       1213   bne err2
00574  fe82
00575  fe82
00576  fe82  ===> Read sector header and compare ID <===
00577  fe82
00578  fe82  a9 00       seek   lda #$00
00579  fe84  85 08              sta work          temporary workspace
00580  fe86  a2 06              ldx #$06
00581  fe88  20 02 ff           jsr head          Find next sector header
00582  fe8b  a0 04              ldy #$04
00583  fe8d
00584  fe8d
00585  fe8d  24 4d       1250   bit via+ifr       interrupt flag register
00586  fe8f  10 fc              bpl 1250
00587  fe91  a5 41              lda via+pra       port a data register
00588  fe93  99 0d 00           sta stab,y        current block header
00589  fe96  45 08              eor work          temporary workspace
00590  fe98  85 08              sta work          temporary workspace
00591  fe9a  88                 dey
00592  fe9b  10 f0              bpl 1250
00593  fe9d  c9 00              cmp #$00
00594  fe9f  d0 2a              bne cserr         DC error 9, header checksum error
                                                  (BC: 27)
00595  fea1  a5 0f              lda stab+2        track number in block header
00596  fea3  a6 12              ldx drive         current drive
00597  fea5  95 03              sta drvst,x       drive status (0/1)
00598  fea7  a5 1e              lda job           temporary storage of job code
00599  fea9  c9 30              cmp #seekj
00600  feab  f0 0f              beq eseek         Transfer sector header to header
                                                  buffer
00601  fead  a0 01              ldy #$01
00602  feaf
00603  feaf
00604  feaf  ===> Compare block header ID with current (disk) ID <===
00605  feaf
00606  feaf  b1 18       1252   lda (hdrpt),y     pointer to active values in header
                                                  table
00607  feb1  d9 0d 00           cmp stab,y        current block header
00608  feb4  d0 19              bne iderr         DC error 11, wrong disk ID (BC: 29)
00609  feb6  88                 dey
00610  feb7  10 f6              bpl 1252          Compare block header ID with current
                                                  (disk) ID
00611  feb9  4c 3f fd           jmp fsnum1        Optimize job (1)
00612  febc
00613  febc
00614  febc  ===> Transfer sector header to header buffer <===
00615  febc
00616  febc       Note: these will be recorded on disk in reverse order!
00617  febc
00618  febc  a0 04       eseek  ldy #$04          ID1, ID2, track, sector, parity
```

```
line   addr  object      source code

00619  febe  b9 0d 00   1251   lda stab,y        current block header
00620  fec1  91 18             sta (hdrpt),y     pointer to active values in header
                                                 table
00621  fec3  88                dey
00622  fec4  10 f8             bpl 1251
00623  fec6
00624  fec6
00625  fec6  ===> DC message 1, OK (BC: 00) <===
00626  fec6
00627  fec6  a9 01       done   lda #goodj
00628  fec8  4c 08 ff    err2   jmp error         Terminate disk controller routines
00629  fecb
00630  fecb
00631  fecb  ===> DC error 9, header checksum error (BC: 27) <===
00632  fecb
00633  fecb  a9 09       cserr  lda #badhch       header block checksum error
00634  fecd  d0 f9              bne err2
00635  fecf
00636  fecf
00637  fecf  ===> DC error 11, wrong disk ID (BC: 29) <===
00638  fecf
00639  fecf  a9 0b       iderr  lda #badid
00640  fed1  d0 f5              bne err2
00641  fed3
00642  fed3
00643  fed3  ===> Find specified data block header <===
00644  fed3
00645  fed3  a0 03       srch   ldy #$03          sector offset
00646  fed5  a9 00              lda #$00
00647  fed7
00648  fed7
00649  fed7  ===> Calculate block header checksum and put in header buffer <===
00650  fed7
00651  fed7  51 18       1412   eor (hdrpt),y     pointer to active values in header
                                                 table
00652  fed9  88                 dey
00653  feda  10 fb              bpl 1412          Calculate block header checksum and
                                                 put in header buffer
00654  fedc  a0 04              ldy #$04          parity offset
00655  fede  91 18              sta (hdrpt),y     pointer to active values in header
                                                 table
00656  fee0  a4 1f              ldy jobnum        storage of current job number
00657  fee2  a2 5a              ldx #$5a
00658  fee4
00659  fee4
00660  fee4  ===> Compare next sector header with header buffer <===
00661  fee4
00662  fee4  20 02 ff   1410   jsr head          Find next sector header
00663  fee7  a0 04              ldy #$04
00664  fee9  24 4d       1411   bit via+ifr       interrupt flag register
00665  feeb  10 fc              bpl 1411
00666  feed  a5 41              lda via+pra       port a data register
00667  feef  d1 18              cmp (hdrpt),y     pointer to active values in header
                                                 table
```

```
line   addr  object      source code

00668  fef1  d0 f1              bne 1410          Compare next sector header with
                                                  header buffer
00669  fef3  88                 dey
00670  fef4  10 f3              bpl 1411
00671  fef6  c8                 iny
00672  fef7  a5 40              lda via+prb       port b data register
00673  fef9  09 80              ora #$80
00674  fefb  85 40              sta via+prb       port b data register
00675  fefd  29 7f              and #$7f
00676  feff  85 40              sta via+prb       port b data register
00677  ff01  60                 rts
00678  ff02
00679  ff02
00680  ff02  ===> Find next sector header <===
00681  ff02
00682  ff02  58          head   cli
00683  ff03  ca                 dex
00684  ff04  10 1e              bpl 1420          Find start of sector header (08)
00685  ff06  a9 02              lda #nohdr        just in case!
00686  ff08
00687  ff08
00688  ff08  ===> Terminate disk controller routines <===
00689  ff08
00690  ff08  a4 1f       error  ldy jobnum        storage of current job number
00691  ff0a  99 03 04           sta jobs,y        job queue for buffer 0
00692  ff0d  48                 pha
00693  ff0e  a6 12              ldx drive         current drive
00694  ff10  a5 00              lda irqcnt        irq counter lo
00695  ff12  6d 02 04           adc cutmt         cutout constant drive motors
00696  ff15  95 01              sta mtrtm,x       motor timing drive 0/1
00697  ff17  68                 pla
00698  ff18  4a                 lsr a
00699  ff19  d0 03              bne 1421          Abort after error
00700  ff1b  4c bb fd           jmp fsnum         Optimize job (2)
00701  ff1e
00702  ff1e
00703  ff1e  ===> Abort after error <===
00704  ff1e
00705  ff1e  a2 3f       1421   ldx #$3f          reset the stack
00706  ff20  9a                 txs
00707  ff21  4c 54 fc           jmp start         DC scanning job queue
00708  ff24
00709  ff24
00710  ff24  ===> Find start of sector header (08) <===
00711  ff24
00712  ff24  20 3e ff    1420   jsr syncl         Wait for a data block sync mark
00713  ff27  c9 08              cmp #hbid
00714  ff29  d0 d7              bne head          Find next sector header
00715  ff2b  60                 rts
00716  ff2c
00717  ff2c
00718  ff2c  24 82       1423   bit rriot+oregb   port b data register
00719  ff2e  50 0d              bvc 1424
```

```
line   addr  object      source code

00720  ff30  24 4d              bit via+ifr       interrupt flag register
00721  ff32  10 f8              bpl 1423
00722  ff34  24 41              bit via+pra       port a data register
00723  ff36  24 40              bit via+prb       port b data register
00724  ff38  ca                 dex
00725  ff39  d0 f1              bne 1423
00726  ff3b  24 82              bit rriot+oregb   port b data register
00727  ff3d  60          1424   rts
00728  ff3e
00729  ff3e
00730  ff3e  ===> Wait for a data block sync mark <===
00731  ff3e
00732  ff3e       A sync mark is ten or more consecutive 1's written onto the disk.
00733  ff3e       They are used to identify the start of a block of information
00734  ff3e       recorded on the disk. The first character following a sync mark
00735  ff3e       is used to determine whether this is a header block ($08) or a
00736  ff3e       data block ($07).
00737  ff3e
00738  ff3e  78          syncl  sei
00739  ff3f  a9 d0              lda #$d0          allow about 20 ms before timing out
00740  ff41  85 45              sta via+t1ch      timer 1 trigger t1c1/t1c1 transfer
00741  ff43  a9 03              lda #nosync
00742  ff45
00743  ff45
00744  ff45  ===> Read timer, test sync <===
00745  ff45
00746  ff45  24 45       1430   bit via+t1ch      test bit 7 for timeout
00747  ff47  10 bf              bpl error         Terminate disk controller routines
00748  ff49  24 82              bit rriot+oregb   port b data register
00749  ff4b  70 f8              bvs 1430          Read timer, test sync
00750  ff4d  24 40              bit via+prb       port b data register
00751  ff4f  24 41              bit via+pra       port a data register
00752  ff51
00753  ff51
00754  ff51  ===> Read data byte following sync mark to accumulator <===
00755  ff51
00756  ff51  24 4d       byte   bit via+ifr       interrupt flag register
00757  ff53  10 fc              bpl byte          Read data byte following sync mark
                                                  to accumulator
00758  ff55  a5 41              lda via+pra       port a data register
00759  ff57  60                 rts
00760  ff58
00761  ff58
00762  ff58  ===> write byte to disk <===
00763  ff58
00764  ff58  20 79 ff    1204  jsr out           Write character in .X to disk
00765  ff5b  a2 00              ldx #$00
00766  ff5d  20 79 ff           jsr out           Write character in .X to disk
00767  ff60  a9 fc              lda #readj
00768  ff62
00769  ff62
00770  ff62
```

```
line   addr  object      source code

00771  ff62  ===> Read <===
00772  ff62
00773  ff62  24 4d       1442   bit via+ifr       interrupt flag register
00774  ff64  10 fc              bpl 1442          Read
00775  ff66  85 4c              sta via+pcr       peripheral control register
00776  ff68  a9 92              lda #$92
00777  ff6a  85 4e              sta via+ier       interrupt enable register
00778  ff6c  20 51 ff           jsr byte          Read data byte following sync mark
                                                  to accumulator
00779  ff6f  24 40              bit via+prb       port b data register
00780  ff71  20 51 ff           jsr byte          Read data byte following sync mark
                                                  to accumulator
00781  ff74  24 40              bit via+prb       port b data register
00782  ff76  4c 51 ff           jmp byte          Read data byte following sync mark
                                                  to accumulator
00783  ff79
00784  ff79
00785  ff79  ===> Write character in .X to disk <===
00786  ff79
00787  ff79  24 4d       out    bit via+ifr       interrupt flag register
00788  ff7b  10 fc              bpl out           Write character in .X to disk
00789  ff7d  86 80              stx rriot+orega   port a data register
00790  ff7f  24 41              bit via+pra       port a data register
00791  ff81  60                 rts
00792  ff82
00793  ff82
00794  ff82  ===> Interrupt routine (IRQ) <===
00795  ff82
00796  ff82  48          irq    pha
00797  ff83  8a                 txa
00798  ff84  48                 pha
00799  ff85  ad 00 04           lda tick          interrupt interval counter
00800  ff88  85 8f              sta rriot+timer   timer offset
00801  ff8a  e6 00              inc irqcnt        irq counter lo
00802  ff8c  a2 01              ldx #$01
00803  ff8e
00804  ff8e
00805  ff8e  ===> Select drive 1 <===
00806  ff8e
00807  ff8e  a5 00       irq01  lda irqcnt        irq counter lo
00808  ff90  d5 01              cmp mtrtm,x       motor timing drive 0/1
00809  ff92  d0 16              bne irq05         Activate head stepping
00810  ff94  bd ea ff           lda andc,x        acceleration/deceleration bits drive
                                                  0/1
00811  ff97  16 03              asl drvst,x       drive status (0/1)
00812  ff99  24 40              bit via+prb       port b data register
00813  ff9b  b0 08              bcs irq04
00814  ff9d  38                 sec
00815  ff9e  bd ea ff           lda andc,x        acceleration/deceleration bits drive
                                                  0/1
00816  ffa1  45 40              eor via+prb       port b data register
00817  ffa3  85 40              sta via+prb       port b data register
00818  ffa5  d0 01       irq04  bne irq40
```

```
line   addr  object      source code

00819  ffa7  18                clc
00820  ffa8
00821  ffa8
00822  ffa8  76 03       irq40  ror drvst,x       drive status (0/1)
00823  ffaa
00824  ffaa
00825  ffaa  ===> Activate head stepping <===
00826  ffaa
00827  ffaa  b5 05       irq05  lda steps,x       steps to move to desired track
                                                  (0-127 in, >127 out)
00828  ffac  d0 09              bne irq07         Step head to next track
00829  ffae  b5 03              lda drvst,x       drive status (0/1)
00830  ffb0  29 bf              and #$bf
00831  ffb2  95 03              sta drvst,x       drive status (0/1)
00832  ffb4  4c e1 ff           jmp irq12
00833  ffb7
00834  ffb7
00835  ffb7  ===> Step head to next track <===
00836  ffb7
00837  ffb7  0a          irq07  asl a
00838  ffb8  a5 40              lda via+prb       port b data register
00839  ffba  3d ef fc           and tab1,x
00840  ffbd  85 07              sta cow           buffer for irq/motor timing bits
00841  ffbf  a5 40              lda via+prb       port b data register
00842  ffc1  3d ec ff           and andd,x        motor timing bits drive 0/1
00843  ffc4  b0 07              bcs irq08
00844  ffc6  d6 05              dec steps,x       steps to move to desired track
                                                  (0-127 in, >127 out)
00845  ffc8  7d e8 ff           adc andb,x        step bits drive 0/1
00846  ffcb  d0 05              bne irq10
00847  ffcd  f6 05       irq08  inc steps,x       steps to move to desired track
                                                  (0-127 in, >127 out)
00848  ffcf  fd e8 ff           sbc andb,x        step bits drive 0/1
00849  ffd2  3d ec ff    irq10  and andd,x        motor timing bits drive 0/1
00850  ffd5  05 07              ora cow           buffer for irq/motor timing bits
00851  ffd7  85 40              sta via+prb       port b data register
00852  ffd9  b5 03              lda drvst,x       drive status (0/1)
00853  ffdb  30 04              bmi irq12
00854  ffdd  a5 00              lda irqcnt        irq counter lo
00855  ffdf  95 01              sta mtrtm,x       motor timing drive 0/1
00856  ffe1  ca          irq12  dex
00857  ffe2  10 aa              bpl irq01         Select drive 1
00858  ffe4  68                 pla
00859  ffe5  aa                 tax
00860  ffe6  68                 pla
00861  ffe7  40                 rti
00862  ffe8
00863  ffe8
00864  ffe8  04 01       andb   .byte $04, $01    step bits drive 0/1
00865  ffea  20 10       andc   .byte $20, $10    acceleration/deceleration bits drive
                                                  0/1
00866  ffec  0c 03       andd   .byte $0c, $03    motor timing bits drive 0/1
00867  ffee
```

```
line   addr  object     source code

00868  ffee  18                .byte $18
00869  ffef
00870  ffef                    *=$fffc
00871  fffc  09 fc             .word resel      Power-on reset
00872  fffe  82 ff             .word irq
00873  0000
00874  0000
00875  0000
00876  0000             .end
00877  0000
```

```
line   addr  object     source code

00879  0000
00880  0000
00881  0000
00882  0000  The format routine is transferred from ROM at $D000 to RAM to
00883  0000  reside at $1100 in buffer #0. It is listed here with the
00884  0000  addresses as seen by the Disk Controller
00885  0000
00886  0000
00887  0000
00888  0000                    * = $0500
00889  0500
00890  0500
00891  0500  a5 1a      formt  lda ftnum        track currently being formatted --
                                               #$FF on reset
00892  0502  10 2f             bpl fmttrk       positive: formatting in progress
00893  0504  78               sei
00894  0505  a9 c1             lda #bmpf1
00895  0507  95 03             sta drvst,x      drive status (0/1): head is stepping
00896  0509  a9 0f             lda #$0f
00897  050b  3d 9a 07          and stepbt,x     step bits drive 0/1
00898  050e  05 40             ora via+prb      port b data register
00899  0510  85 40             sta via+prb      port b data register
00900  0512  a9 8c             lda #stpcnt      initialize step counter to move head
                                               in drive .X
00901  0514  95 05             sta steps,x      steps to move to desired track
                                               (0-127 in, >127 out)
00902  0516  58               cli
00903  0517
00904  0517
00905  0517  ===> Wait until head positioned and set pointers <===
00906  0517
00907  0517  b5 05      form0  lda steps,x      steps to move to desired track
                                               (0-127 in, >127 out)
00908  0519  d0 fc             bne form0        Wait until head positioned and set
                                               pointers
00909  051b  98               tya              buffer number
00910  051c  0a               asl a
00911  051d  0a               asl a
00912  051e  0a               asl a            multiply by 8
00913  051f  18               clc
00914  0520  69 21             adc #bufpnt      pointer lo into header table
00915  0522  85 18             sta hdrpt        pointer to active values in header
                                               table
00916  0524  a0 00             ldy #$00
00917  0526  84 1f             sty errcnt       error counter
00918  0528  c8               iny
00919  0529  84 1a             sty ftnum        track 1 to be formatted
00920  052b  20 65 07   form01 jsr wrtshd      Create header image for first sector
00921  052e  a4 1f             ldy jobnum       storage of current job code number
00922  0530  6c 00 fc          jmp (starti)     vector to main idling loop for disk
                                               controller
00923  0533
00924  0533
```

```
line   addr  object     source code

00925  0533  ===> Format a track <===
00926  0533
00927  0533  a0 02      fmttrk ldy #$02          offset in header buffer to track
00928  0535  51 18             eor (hdrpt),y     pointer to active values in header
                                                 table
00929  0537  d0 f2             bne form01
00930  0539  a9 00             lda #$00          reset
00931  053b  85 1f             sta errcnt        error counter
00932  053d
00933  053d
00934  053d  ===> Check write protect status <===
00935  053d
00936  053d  78        fmtt01 sei
00937  053e  20 65 07         jsr wrtshd        Create header image for first sector
00938  0541  a9 08             lda #wrprot
00939  0543  25 82             and rriot+oregb   bit 3 is 1 if write protect on
00940  0545  f0 03             beq chkspd        Check speed
00941  0547  4c 84 06          jmp fmter1        Permit ten tries before aborting
00942  054a
00943  054a
00944  054a  ===> Check speed <===
00945  054a
00946  054a  20 80 07  chkspd jsr wrtnul        Erase track by writing 32*256 nulls
00947  054d  a2 ff             ldx #$ff
00948  054f  a9 da             lda #$da
00949  0551  20 2e 07          jsr write         Send value in .X to output register
00950  0554  85 4c             sta via+pcr       peripheral control register
00951  0556  20 2e 07          jsr write         Send value in .X to output register
00952  0559  20 2e 07          jsr write         Send value in .X to output register
00953  055c  a9 dc             lda #$dc
00954  055e  20 2e 07          jsr write         Send value in .X to output register
00955  0561  a2 0f             ldx #offbyt
00956  0563  20 2e 07          jsr write         Send value in .X to output register
00957  0566  85 4c             sta via+pcr       peripheral control register
00958  0568  20 0d 07          jsr readb         Read four header bytes
00959  056b  20 3e 07          jsr loop          Loop to time non-sync area
00960  056e  20 37 07          jsr read          Read a data byte
00961  0571  c9 0f             cmp #offbyt
00962  0573  f0 03             beq gaps          Calculate track capacity
00963  0575  4c 82 06          jmp fmterr        Format error
00964  0578
00965  0578
00966  0578  ===> Calculate track capacity <===
00967  0578
00968  0578  a9 11      gaps   lda #$11          number of header bytes without OFF
                                                 bytes
00969  057a  18                clc
00970  057b  6d 9d 04          adc gap1          size of gap after sector header
00971  057e  85 0a             sta nxtrk         17+9
00972  0580  a6 15             ldx sectr         highest sector number in current
                                                 track
00973  0582  a0 00             ldy #$00          calculate OFF bytes for gap
00974  0584  a9 00             lda #$00
```

```
line   addr  object      source code

00975  0586  18          gaps1  clc
00976  0587  65 0a              adc nxtrk       .A is now 26
00977  0589  90 01              bcc gaps2
00978  058b  c8                 iny
00979  058c  c8          gaps2  iny             .Y is number of sector plus
                                                remainder from .A
00980  058d  ca                 dex             leaving in (.Y,.A) the total of
                                                bytes
00981  058e  d0 f6              bne gaps1       including the header, plus one extra
00982  0590  49 ff              eor #$ff        byte in each
00983  0592  38                 sec
00984  0593  69 00              adc #$00        this is then subtracted from the
00985  0595  18                 clc             track capacity in (count1),
                                                resulting in a
00986  0596  6d a0 07           adc count1      total of the gap bytes
00987  0599  b0 03              bcs gaps3
00988  059b  ce 9f 07           dec counth
00989  059e  aa          gaps3  tax
00990  059f  98                 tya
00991  05a0  49 ff              eor #$ff
00992  05a2  38                 sec
00993  05a3  69 00              adc #$00
00994  05a5  18                 clc
00995  05a6  6d 9f 07           adc counth
00996  05a9  10 03              bpl gaps4       Calculate gap size
00997  05ab  4c 82 06           jmp fmterr      Format error
00998  05ae
00999  05ae
01000  05ae  ===> Calculate gap size <===
01001  05ae
01002  05ae  a8          gaps4  tay             total of gap bytes divided by the
01003  05af  8a                 txa             number of sectors
01004  05b0  a2 00              ldx #$00
01005  05b2  38          gaps5  sec
01006  05b3  e5 15              sbc sectr       highest sector number in current
                                                track
01007  05b5  b0 03              bcs gaps6
01008  05b7  88                 dey
01009  05b8  30 03              bmi gaps7
01010  05ba  e8          gaps6  inx
01011  05bb  d0 f5              bne gaps5
01012  05bd  86 0a       gaps7  stx nxtrk
01013  05bf  ec 9e 04           cpx gap2        number of bytes between sectors
01014  05c2  b0 03              bcs gaps8       Calculate control value, write 8192
                                                zero's
01015  05c4  4c 82 06           jmp fmterr      Format error
01016  05c7
01017  05c7
01018  05c7  ===> Calculate control value, write 8192 zero's <===
01019  05c7
01020  05c7  18          gaps8  clc
01021  05c8  65 15              adc sectr       highest sector number in current
                                                track
```

```
line   addr  object     source code

01022  05ca  8d 9e 07          sta chksum       control for header checksum
01023  05cd  20 80 07          jsr wrtnul       Record 32*256 null bytes to current
                                                track
01024  05d0
01025  05d0
01026  05d0  ===> Record current sector header and data to disk <===
01027  05d0
01028  05d0  a9 de       wrtsec lda #$de
01029  05d2  a2 ff              ldx #$ff         sync
01030  05d4  20 2e 07           jsr write        Send value in .X to output register
01031  05d7  85 4c              sta via+pcr      peripheral control register
01032  05d9  20 2e 07           jsr write        Send value in .X to output register
01033  05dc  20 2e 07           jsr write        Send value in .X to output register
01034  05df  a9 dc              lda #$dc
01035  05e1  20 2e 07           jsr write        Send value in .X to output register
01036  05e4  a2 08              ldx #hbid        constant for block header
01037  05e6  20 2e 07           jsr write        Send value in .X to output register
01038  05e9  85 4c              sta via+pcr      peripheral control register
01039  05eb  a2 ff              ldx #$ff         sync
01040  05ed  ad 9d 07           lda bkprty       header checksum
01041  05f0  24 4d       wrtse1 bit via+ifr      wait for bit 7 to set
01042  05f2  10 fc              bpl wrtse1
01043  05f4  85 80              sta rriot+orega  port a output register
01044  05f6  24 41              bit via+pra      port a data register
01045  05f8  4d 9c 07           eor sector       calculate checksum
01046  05fb  ac 9c 07           ldy sector       number of current sector
01047  05fe  ee 9c 07           inc sector
01048  0601  24 4d       wrtse2 bit via+ifr      wait for bit 7 to set
01049  0603  10 fc              bpl wrtse2
01050  0605  84 80              sty rriot+orega  port a output register
01051  0607  24 41              bit via+pra      port a data register
01052  0609  4d 9c 07           eor sector       next checksum
01053  060c  8d 9d 07           sta bkprty       header checksum
01054  060f  a0 02              ldy #$02         offset in header buffer
01055  0611  b1 18       wrtse3 lda (hdrpt),y    track, ID2, ID1
01056  0613  24 4d       wrtse4 bit via+ifr      wait for bit 7 to set
01057  0615  10 fc              bpl wrtse4
01058  0617  85 80              sta rriot+orega  port a output register
01059  0619  24 41              bit via+pra      port a data register
01060  061b  88                 dey
01061  061c  10 f3              bpl wrtse3
01062  061e  a9 00              lda #$00         null code
01063  0620  ac 9d 04           ldy gap1         size of gap after sector header
01064  0623  24 4d       wrtse5 bit via+ifr      wait for bit 7 to set
01065  0625  10 fc              bpl wrtse5
01066  0627  85 80              sta rriot+orega  port a output register
01067  0629  24 41              bit via+pra      port a data register
01068  062b  88                 dey
01069  062c  d0 f5              bne wrtse5       write .Y times
01070  062e  a9 de              lda #$de
01071  0630  20 2e 07           jsr write        Send value in .X to output register
01072  0633  85 4c              sta via+pcr      peripheral control register
01073  0635  20 2e 07           jsr write        Send value in .X to output register
```

```
line   addr  object      source code

01074  0638  20 2e 07           jsr write        Send value in .X to output register
01075  063b  a9 dc              lda #$dc
01076  063d  20 2e 07           jsr write        Send value in .X to output register
01077  0640  a2 07              ldx #bid         data block ID
01078  0642  20 2e 07           jsr write        Send value in .X to output register
01079  0645  85 4c              sta via+pcr      peripheral control register
01080  0647  a0 00              ldy #$00         block byte counter
01081  0649  a2 00              ldx #$00         null code
01082  064b  24 4d       wrtse6 bit via+ifr      wait for bit 7 to set
01083  064d  10 fc              bpl wrtse6
01084  064f  86 80              stx rriot+orega  port a output register
01085  0651  24 41              bit via+pra      port a data register
01086  0653  88                 dey
01087  0654  d0 f5              bne wrtse6       write 256 times
01088  0656  a4 0a              ldy nxtrk
01089  0658  20 2e 07           jsr write        Send value in .X to output register
01090  065b  24 4d       wrtse7 bit via+ifr      wait for bit 7 to set
01091  065d  10 fc              bpl wrtse7
01092  065f  86 80              stx rriot+orega  port a output register
01093  0661  24 41              bit via+pra      port a data register
01094  0663  88                 dey
01095  0664  10 f5              bpl wrtse7
01096  0666  ad 9c 07           lda sector       number of current sector
01097  0669  c5 15              cmp sectr        highest sector number in current
                                                track
01098  066b  f0 03              beq verfbk       Verify current sector
01099  066d  4c d0 05           jmp wrtsec       Record current sector header and
                                                data to disk
01100  0670
01101  0670
01102  0670  ===> Verify current sector <===
01103  0670
01104  0670  20 0d 07    verfbk jsr readb       Read four header bytes
01105  0673  a9 00              lda #$00
01106  0675  8d 9c 07           sta sector       number of current sector
01107  0678
01108  0678
01109  0678  ===> Check header ID <===
01110  0678
01111  0678  20 3e 07    verfbl jsr loop        Loop to time non-sync area
01112  067b  20 37 07           jsr read         Read a data byte
01113  067e  c9 08              cmp #hbid
01114  0680  f0 0e              beq 1310         Read and store header checksum
01115  0682
01116  0682
01117  0682  ===> Format error <===
01118  0682
01119  0682  a9 0c       fmterr lda #badfmt
01120  0684
01121  0684
01122  0684
```

```
line   addr  object      source code

01123  0684  ===> Permit ten tries before aborting <===
01124  0684
01125  0684  58          fmterl cli
01126  0685  e6 1f              inc errcnt       error counter
01127  0687  a0 0a              ldy #tries
01128  0689  c4 1f              cpy errcnt       error counter
01129  068b  f0 79              beq errr         Exit with format done flag, error
                                                 number in .A
01130  068d  4c 3d 05           jmp fmtt01       Check write protect status
01131  0690
01132  0690
01133  0690  ===> Read and store header checksum <===
01134  0690
01135  0690  20 37 07   1310    jsr read         Read a data byte
01136  0693  8d 9d 07           sta bkprty       header checksum
01137  0696  20 37 07           jsr read         Read a data byte
01138  0699  cd 9c 07           cmp sector       number of current sector
01139  069c  d0 e4              bne fmterr       Format error
01140  069e  4d 9d 07           eor bkprty       header checksum
01141  06a1  a0 02              ldy #$02
01142  06a3  24 4d       1330   bit via+ifr      wait for bit 7 to set
01143  06a5  10 fc              bpl 1330
01144  06a7  45 41              eor via+pra      port a data register
01145  06a9  88                 dey
01146  06aa  10 f7              bpl 1330
01147  06ac  a8                 tay
01148  06ad  d0 d3              bne fmterr       Format error
01149  06af  ee 9c 07           inc sector       number of current sector
01150  06b2  20 3e 07           jsr loop         Loop to time non-sync area
01151  06b5  20 37 07           jsr read         Read a data byte
01152  06b8  c9 07              cmp #bid
01153  06ba  d0 c6              bne fmterr       Format error
01154  06bc  a0 00              ldy #$00
01155  06be
01156  06be
01157  06be  ===> Calculate checksum for data sector <===
01158  06be
01159  06be  24 4d       verfb2 bit via+ifr      wait for bit 7 to set
01160  06c0  10 fc              bpl verfb2       Calculate checksum for data sector
01161  06c2  a5 41              lda via+pra      port a data register
01162  06c4  d0 bc              bne fmterr       Format error
01163  06c6  88                 dey
01164  06c7  d0 f5              bne verfb2       Calculate checksum for data sector
01165  06c9  20 37 07           jsr read         Read a data byte
01166  06cc  d0 b4              bne fmterr       Format error
01167  06ce  ad 9c 07           lda sector       number of current sector
01168  06d1  c5 15              cmp sectr        highest sector number in current
                                                 track
01169  06d3  d0 a3              bne verfb1       Check header ID
01170  06d5  20 3e 07           jsr loop         Loop to time non-sync area
01171  06d8  ad 9f 07           lda counth       timer hi
01172  06db  f0 03              beq verfb3
01173  06dd  4c 82 06           jmp fmterr       Format error
```

```
line   addr  object      source code

01174  06e0
01175  06e0
01176  06e0  ad 9e 07    verfb3 lda chksum       control for header checksum
01177  06e3  38                 sec
01178  06e4  65 0a              adc nxtrk        size of gap
01179  06e6  38                 sec
01180  06e7  ed a0 07           sbc count1       minus the total of bytes read
01181  06ea  10 05              bpl verfb4
01182  06ec  49 ff              eor #$ff
01183  06ee  38                 sec
01184  06ef  69 00              adc #$00
01185  06f1  c9 1c       verfb4 cmp #$1c         between -28 and +28?
01186  06f3  90 03              bcc finis        Find out if this is last track to be
                                                 formatted
01187  06f5  4c 82 06           jmp fmterr       Format error
01188  06f8
01189  06f8
01190  06f8  ===> Find out if this is last track to be formatted <===
01191  06f8
01192  06f8  e6 1a       finis  inc ftnum        track currently being formatted
01193  06fa  58                 cli
01194  06fb  a9 24              lda #maxtrk
01195  06fd  c5 1a              cmp ftnum        track currently being formatted
01196  06ff  f0 03              beq fmtend       Exit with OK flag
01197  0701  4c 2b 05    finis1 jmp form01
01198  0704
01199  0704
01200  0704  ===> Exit with OK flag <===
01201  0704
01202  0704  a9 01       fmtend lda #goodj
01203  0706
01204  0706
01205  0706  ===> Exit with format done flag, error number in .A <===
01206  0706
01207  0706  a0 ff       errr   ldy #$ff
01208  0708  84 1a              sty ftnum        track currently being formatted
01209  070a  6c 02 fc           jmp (donei)      vector to disk controller job
                                                 completed
01210  070d
01211  070d
01212  070d  ===> Read four header bytes <===
01213  070d
01214  070d  20 2e 07    readb  jsr write        Send value in .X to output register
01215  0710  24 4d       readb1 bit via+ifr      wait for bit 7 to set
01216  0712  10 fc              bpl readb1
01217  0714  a9 fc              lda #readj
01218  0716  85 4c              sta via+pcr      peripheral control register
01219  0718  a9 92              lda #$92
01220  071a  85 4e              sta via+ier      interrupt enable register
01221  071c  a2 03              ldx #$03
01222  071e  20 37 07    readb2 jsr read         Read a data byte
01223  0721  24 40              bit via+prb      port b data register
01224  0723  ca                 dex
```

```
line   addr  object     source code

01225  0724  d0 f8             bne readb2
01226  0726  60                rts
01227  0727
01228  0727
01229  0727  ===> Switch to write mode <===
01230  0727
01231  0727  a0 10       pwrite ldy #writej
01232  0729  84 4e             sty via+ier       interrupt enable register
01233  072b  85 4c             sta via+pcr       peripheral control register
01234  072d  60                rts
01235  072e
01236  072e
01237  072e  ===> Send value in .X to output register <===
01238  072e
01239  072e  24 4d       write  bit via+ifr       wait for bit 7 to set
01240  0730  10 fc             bpl write         Send value in .X to output register
01241  0732  86 80             stx rriot+orega   port a output register
01242  0734  24 41             bit via+pra       port a data register
01243  0736  60                rts
01244  0737
01245  0737
01246  0737  ===> Read a data byte <===
01247  0737
01248  0737  24 4d       read   bit via+ifr       wait for bit 7 to set
01249  0739  10 fc             bpl read          Read a data byte
01250  073b  a5 41             lda via+pra       port a data register
01251  073d  60                rts
01252  073e
01253  073e
01254  073e  ===> Loop to time non-sync area <===
01255  073e
01256  073e  a0 00       loop   ldy #$00          timer lo
01257  0740  8c 9f 07           sty counth        timer hi
01258  0743  24 82       loop1  bit rriot+oregb   look for a sync
01259  0745  50 16             bvc loop3         found a sync
01260  0747  24 4d             bit via+ifr       timed out?
01261  0749  10 f8             bpl loop1         keep looking
01262  074b  24 41             bit via+pra       port a data register
01263  074d  24 40             bit via+prb       port b data register
01264  074f  c8                iny               timer lo
01265  0750  d0 f1             bne loop1         do this 65535 times
01266  0752  ee 9f 07           inc counth        timer/counter hi
01267  0755  d0 03             bne loop2         keep looking
01268  0757  4c 82 06           jmp fmterr        no sync found: format error
01269  075a
01270  075a  4c 43 07   loop2  jmp loop1         keep looking
01271  075d
01272  075d  8c a0 07   loop3  sty count1        sync found, so store non-sync times
01273  0760  24 40             bit via+prb       port b data register
01274  0762  24 41             bit via+pra       port a data register
01275  0764  60                rts
01276  0765
01277  0765
```

```
line   addr  object      source code

01278  0765  ===> Create header image for first sector <===
01279  0765
01280  0765  a0 02       wrtshd ldy #$02          offset in header buffer
01281  0767  a5 1a              lda ftnum          track currently being formatted
01282  0769  91 18              sta (hdrpt),y      pointer to active values in header
                                                  table
01283  076b  a9 00              lda #$00
01284  076d  8d 9c 07           sta sector         number of current sector
01285  0770  c8                iny                offset to sector
01286  0771  91 18              sta (hdrpt),y      pointer to active values in header
                                                  table
01287  0773
01288  0773
01289  0773  ===> Calculate checksum <===
01290  0773
01291  0773  51 18       wrtshl eor (hdrpt),y     pointer to active values in header
                                                  table
01292  0775  88                dey
01293  0776  10 fb            bpl wrtshl         Calculate checksum
01294  0778  8d 9d 07         sta bkprty         header checksum
01295  077b  a0 04            ldy #$04           offset parity
01296  077d  91 18            sta (hdrpt),y      pointer to active values in header
                                                  table
01297  077f  60                rts
01298  0780
01299  0780
01300  0780  ===> Record 32*256 null bytes to current track <===
01301  0780
01302  0780  a2 00       wrtnul ldx #$00          null byte
01303  0782  a0 00              ldy #$00           counter lo
01304  0784  a9 20              lda #32
01305  0786  8d 9f 07           sta counth         timer/counter hi
01306  0789  a9 dc              lda #$dc
01307  078b  20 27 07           jsr pwrite         Switch to write
01308  078e  20 2e 07    wrtnu1 jsr write          Send value in .X to output register
01309  0791  88                dey
01310  0792  d0 fa              bne wrtnul
01311  0794  ce 9f 07           dec counth         timer/counter hi
01312  0797  d0 f5              bne wrtnul
01313  0799  60                rts
01314  079a
01315  079a
01316  079a  ===> step bits drive 0/1 <===
01317  079a
01318  079a  0c 03       stepbt .byte $0c, $03
01319  079c
01320  079c
01321  079c  ===> number of current sector <===
01322  079c
01323  079c  48          sector .byte $48
01324  079d
01325  079d
01326  079d
```

```
line   addr  object      source code

01327  079d  ===> header checksum <===
01328  079d
01329  079d  53          bkprty .byte $53
01330  079e
01331  079e
01332  079e  ===> control for header checksum <===
01333  079e
01334  079e  50          chksum .byte $50
01335  079f
01336  079f
01337  079f  ===> timer/counter hi <===
01338  079f
01339  079f  aa          counth .byte $aa
01340  07a0
01341  07a0
01342  07a0  ===> timer/counter lo <===
01343  07a0
01344  07a0  02          countl .byte $02
01345  07a1
01345  07a1              .end
01346  07a1                    .end

errors in pass 1 = 00000
errors in pass 2 = 00000

assembly completed
```

```
label    address  line numbers

acr     =   $0b:    74,   166
actjob  = $04a0:   106,   409
andb    = $ffe8:   845,   848,   864
andc    = $ffea:   201,   204,   810,   815,   865
andd    = $ffec:   343,   842,   849,   866
badbch  =   $05:    12,   561
badfmt  =   $0c:    17,  1119
badhch  =   $09:    15,   633
badid   =   $0b:    16,   639
bid     =   $07:    19,   469,   521,  1077,  1152
bkprty  = $079d:  1040,  1053,  1136,  1140,  1294,  1329
bmpfl   =   $c1:    26,   340,   894
bufpnt  =   $21:    23,   428,   914
bufpt   =   $16:    58,   169,   333,   334,   401,   455,   530,   552
bump    = $fd2a:   248,   339
bumpj   =   $40:    32,   232,   246,   286
byte    = $ff51:   558,   756,   757,   778,   780,   782
chkspd  = $054a:   940,   946
chksum  = $079e:  1022,  1176,  1334
counth  = $079f:   988,   995,  1171,  1257,  1266,  1305,  1311,  1339
countl  = $07a0:   986,  1180,  1272,  1344
cow     =   $07:    47,   840,   850
csect   =   $0c:    52,   354,   393,   397
csectr  =   $0b:    51,   391,   396
cserr   = $fecb:   594,   633
ctrack  =   $09:    49,   269,   277
cutmt   = $0402:    97,   156,   695
ddrb    =   $02:    71,   155
delay   = $0401:    96,   177,   207
done    = $fec6:   348,   565,   627
donei   = $fc02:   113,   129,  1209
drega   =   $01:    84,   157
dregb   =   $03:    86,   160
drive   =   $12:    54,   198,   243,   319,   321,   379,   596,   693
drvst   =   $03:    45,   174,   175,   213,   217,   253,   287,   290,   300
                   341,   597,   811,   822,   829,   831,   852,   895
dstrt   = $fddd:   447,   465,   548
dstrtx  = $fdef:   470,   477
err2    = $fec8:   573,   628,   634,   640
errcnt  =    $1f:   62,   917,   931,  1126,  1128
error   = $ff08:   129,   491,   628,   690,   747
errr    = $0706:  1129,  1207
errx    = $fdfa:   472,   491
eseek   = $febc:   600,   618
exl     = $fd20:   191,   324,   330
exec    =   $e0:    37,   323,   382
finis   = $06f8:  1186,  1192
finisl  = $0701:  1197
fmtend  = $0704:  1196,  1202
fmterl  = $0684:   941,  1125
fmterr  = $0682:   963,   997,  1015,  1119,  1139,  1148,  1153,  1162,  1166
                  1173,  1187,  1268
fmttol  = $053d:   936,  1130
```

```
label    address  line numbers

fmttrk = $0533:   892,   927
form0  = $0517:   907,   908
form01 = $052b:   920,   929,  1197
formt  = $0500:   111,   891
fsnum  = $fdbb:   437,   543,   700
fsnum1 = $fd3f:   353,   440,   611
fsnum2 = $fd4e:   359,   365
fsnum3 = $fd56:   371,   403
fsnum4 = $fd7b:   388,   391
fsnum5 = $fd8e:   372,   376,   380,   383,   394,   402
fsnum6 = $fd98:   405,   409
ftnum  =   $1a:    60,   158,   891,   919,  1192,  1195,  1208,  1281
gap1   = $049d:   102,   499,   970,  1063
gap2   = $049e:   103,  1013
gaps   = $0578:   962,   968
gaps1  = $0586:   975,   981
gaps2  = $058c:   977,   979
gaps3  = $059e:   987,   989
gaps4  = $05ae:   996,  1002
gaps5  = $05b2:  1005,  1011
gaps6  = $05ba:  1007,  1010
gaps7  = $05bd:  1009,  1012
gaps8  = $05c7:  1014,  1020
goodj  =   $01:     8,   627,  1202
gotu   = $fcf4:   256,   259,   300
gotu1  = $fcfc:   308,   310
hbid   =   $08:    20,   713,  1036,  1113
hdrpt  =   $18:    59,   179,   258,   303,   374,   386,   429,   438,   606
                  620,   651,   655,   667,   915,   928,  1055,  1282,  1286
                 1291,  1296
head   = $ff02:   581,   662,   682,   714
iderr  = $fecf:   608,   639
ier    =   $0e:    77,   164,   497,   777,  1220,  1232
ifr    =   $0d:    76,   452,   505,   517,   527,   549,   564,   585,   664
                  720,   756,   773,   787,  1041,  1048,  1056,  1064,  1082
                 1090,  1142,  1159,  1215,  1239,  1248,  1260
irq    = $ff82:   796,   872
irq01  = $ff8e:   807,   857
irq04  = $ffa5:   813,   818
irq05  = $ffaa:   809,   827
irq07  = $ffb7:   828,   837
irq08  = $ffcd:   843,   847
irq10  = $ffd2:   846,   849
irq12  = $ffe1:   832,   853,   856
irq40  = $ffa8:   818,   822
irqcnt =   $00:    43,   148,   149,   206,   215,   694,   801,   807,   854
job    =   $1e:    61,   245,   322,   412,   423,   598
jobnum =   $1f:    63,   190,   235,   274,   330,   367,   395,   402,   410
                  419,   539,   656,   690,   921
jobs   = $0403:    98,   151,   186,   420,   540,   542,   691
jumpc  =   $d0:    36,   188
jumpj  =   $50:    33
l100   = $fdcb:   452,   453,   459
```

```
label    address  line numbers

1200   = $fe08:   505,   506,   509
1202   = $fe22:   517,   518
1203   = $fe32:   527,   528,   535
1204   = $ff58:   538,   764
1210   = $fe5a:   549,   550,   557
1211   = $fe6b:   460,   558
12111  = $fe76:   560,   563
1212   = $fe7e:   553,   572
1213   = $fe80:   562,   566,   573
1250   = $fe8d:   585,   586,   592
1251   = $febe:   619,   622
1252   = $feaf:   606,   610
1310   = $0690:  1114,  1135
1330   = $06a3:  1142,  1143,  1146
1410   = $fee4:   662,   668
1411   = $fee9:   664,   665,   670
1412   = $fed7:   651,   653
1420   = $ff24:   684,   712
1421   = $ff1e:   699,   705
1423   = $ff2c:   718,   721,   725
1424   = $ff3d:   719,   727
1430   = $ff45:   746,   749
1442   = $ff62:   773,   774
loop   = $073e:   959,  1111,  1150,  1170,  1256
loop1  = $0743:  1258,  1261,  1265,  1270
loop2  = $075a:  1267,  1270
loop3  = $075d:  1259,  1272
maxtrk =   $24:    24,  1194
mtrtm  =   $01:    44,   208,   216,   696,   808,   855
nexts  =   $14:    56,   365,   387
nodblk =   $04:    11,   471
nohdr  =   $02:     9,   685
nosync =   $03:    10,   741
numsec = $0499:   101,   311
nxtrk  =   $0a:    50,   260,   268,   971,   976,  1012,  1088,  1178
offbyt =   $0f:    22,   955,   961
orega  =   $00:    83,   522,   531,   789,  1043,  1050,  1058,  1066,  1084
                 1092,  1241
oregb  =   $02:    85,   316,   320,   484,   718,   726,   748,   939,  1258
out    = $ff79:   514,   515,   516,   537,   764,   766,   787,   788
pcr    =   $0c:    75,   162,   511,   520,   775,   950,   957,  1031,  1038
                 1072,  1079,  1218,  1233
pra    =   $01:    70,   454,   507,   519,   529,   551,   587,   666,   722
                  751,   758,   790,  1044,  1051,  1059,  1067,  1085,  1093
                 1144,  1161,  1242,  1250,  1262,  1274
prb    =   $00:    69,   154,   200,   203,   205,   344,   345,   672,   674
                  676,   723,   750,   779,   781,   812,   816,   817,   838
                  841,   851,   898,   899,  1223,  1263,  1273
pwrite = $0727:  1231,  1307
que    = $fc92:   219,   231
que1   = $fc9b:   240,   275
que2   = $fccf:   241,   244,   266,   274
que4   = $fcaf:   247,   253
```

```
label    address  line numbers

                   205,   344,   345,   452,   454,   497,   505,   507,   511
                   517,   519,   520,   527,   529,   549,   551,   564,   585
                   587,   664,   666,   672,   674,   676,   720,   722,   723
                   740,   746,   750,   751,   756,   758,   773,   775,   777
                   779,   781,   787,   790,   812,   816,   817,   838,   841
                   851,   898,   899,   950,   957,  1031,  1038,  1041,  1044
                  1048,  1051,  1056,  1059,  1064,  1067,  1072,  1079,  1082
                  1085,  1090,  1093,  1142,  1144,  1159,  1161,  1215,  1218
                  1220,  1223,  1232,  1233,  1239,  1242,  1248,  1250,  1260
                  1262,  1263,  1273,  1274
work   =   $08:    48,   233,   265,   267,   276,   282,   289,   315,   318
                   373,   377,   456,   457,   466,   524,   532,   533,   554
                   555,   559,   579,   589,   590
wprot  = $fdf0:    446,   482
write  = $072e:    949,   951,   952,   954,   956,  1030,  1032,  1033,  1035
                  1037,  1071,  1073,  1074,  1076,  1078,  1089,  1214,  1239
                  1240,  1308
writej =   $10:     29,   496,  1231
wrprot =   $08:     14,   938
wrtnul = $078e:   1308,  1310,  1312
wrtnul = $0780:    946,  1023,  1302
wrtse1 = $05f0:   1041,  1042
wrtse2 = $0601:   1048,  1049
wrtse3 = $0611:   1055,  1061
wrtse4 = $0613:   1056,  1057
wrtse5 = $0623:   1064,  1065,  1069
wrtse6 = $064b:   1082,  1083,  1087
wrtse7 = $065b:   1090,  1091,  1095
wrtsec = $05d0:   1028,  1099
wrtshl = $0773:   1291,  1293
wrtshd = $0765:    920,   937,  1280
wverer =   $07:     13,   572
```

```
label    address  line numbers

que5   = $fcc5:   261,   265
que6   = $fcde:   278,   282
que7   = $fced:   291,   301
read   = $0737:   960,  1112,  1135,  1137,  1151,  1165,  1222,  1248,  1249
readb  = $070d:   958,  1104,  1214
readb1 = $0710:  1215,  1216
readb2 = $071e:  1222,  1225
readc  =   $80:    35,   173
readj  =   $fc:    28,   161,   767,  1217
reed   = $fdc4:   413,   445
resel  = $fc09:   143,   871
rese2  = $fc16:   150,   150
rese22 = $fc12:   148,   153
reset  = $fc04:   114,   134,   135
rite   = $fdfd:   486,   496
rriot  =   $80:    82,   157,   160,   172,   316,   320,   484,   522,   531
                  718,   726,   748,   789,   800,   939,  1043,  1050,  1058
                 1066,  1084,  1092,  1241,  1258
sector = $079c:  1045,  1046,  1047,  1052,  1096,  1106,  1138,  1149,  1167
                 1284,  1323
sectr  =   $15:    57,   312,   358,   390,   972,  1006,  1021,  1097,  1168
seek   = $fe82:   325,   578
seekj  =   $30:    31,   599
setjob = $fda5:   240,   371,   411,   419
srch   = $fed3:   467,   498,   645
stab   =   $0d:    53,   355,   439,   588,   595,   607,   619
start  = $fc54:   124,   184,   226,   406,   707
start1 = $fc56:   185,   225
start2 = $fc65:   189,   196
start3 = $fc80:   202,   213
start4 = $fc8a:   214,   218
start5 = $fc8d:   187,   224
start6 = $fc90:   226,   291
starti = $fc00:   112,   124,   922
stepbt = $079a:   897,  1318
steps  =   $05:    46,   285,   347,   827,   844,   847,   901,   907
stpcnt =   $8c:    25,   346,   900
sync1  = $ff3e:   468,   712,   738
t1ch   =   $05:    73,   740,   746
t11c1  =   $04:    72,   168
tabl   = $fcef:   294,   308,   839
tick   = $0400:    95,   134,   146,   171,   799
timer  =   $0f:    87,   172,   800
track  =   $13:    55,   255,   375
tries  =   $0a:    21,  1127
verfb1 = $0678:  1111,  1169
verfb2 = $06be:  1159,  1160,  1164
verfb3 = $06e0:  1172,  1176
verfb4 = $06f1:  1181,  1185
verfbk = $0670:  1098,  1104
verfy  = $fe57:   483,   548
verfyj =   $20:    30,   482
via    =   $40:    68,   154,   155,   162,   164,   166,   168,   200,   203
```