```
1
2
3
4
5  =                    include cpyright.def
6  =
7  =
8  =                ;*********************************************************
9  =                ;*
10 =                ;*                M P / M - 8 6   2 . 1
11 =                ;*                ====================
12 =                ;*
13 =                ;*                Copyright (c) 1981
14 =                ;*
15 =                ;*                 Digital Research
16 =                ;*                   P.O.Box 579
17 =                ;*           Pacific Grove, California 93950
18 =                ;*
19 =                ;*                   (408) 649-3896
20 =                ;*                   TWX 9103605001
21 =                ;*
22 =                ;* All Information contained in this source listing is
23 =                ;*
24 =                ;*                      PROPRIETORY
25 =                ;*                      ===========
26 =                ;*
27 =                ;* All rights reserved.  No part of this document
28 =                ;* may be reproduced, transmitted, stored in a
29 =                ;* retrieval system, or translated into any language
30 =                ;* or computer language, in any form or by any means
31 =                ;* without the prior written permission of Digital
32 =                ;* Research, P.O Box 579, Pacific Grove, California.
33 =                ;*
34 =                ;*********************************************************
35
36
37                  ;*********************************************************
38                  ;*
39                  ;*      BDOS - Basic Disk Operating System
40                  ;*
41                  ;*********************************************************
42
```

```
                                eject 1 include equ.bdo         ; symbol definitions
                        ;
                        ;               BDOS symbols:
                        ;
FFFF                    on      EQU     0ffffh
0000                    off     EQU     00000h
                        ;
                        ;               Special 8086 symbols:
                        ;
0000                    b       equ     byte ptr 0
0000                    w       equ     word ptr 0
                        ;
                        ;       literal constants
                        ;
FFFF                    enddir  EQU     0ffffh          ;end of directory
                        ;
                        ;       file control block (fcb) constants
                        ;
0020                    fcblen  EQU     32              ;fcb length
00E5                    empty   EQU     0e5h            ;empty directory entry
007F                    lstrec  EQU     127             ;last record# in extent
0080                    recsiz  EQU     128             ;record size
0004                    dirrec  EQU     recsiz/fcblen   ;directory elts / record
0002                    dskshf  EQU     2               ;log2(dirrec)
0003                    dskmsk  EQU     dirrec-1
0005                    fcbshf  EQU     5               ;log2(fcblen)
001F                    maxext  EQU     31              ;largest extent number
003F                    maxmod  EQU     63              ;largest module number
0080                    fwfmsk  EQU     80h             ;file write flag is high
                                                        ;order modnum
000F                    namlen  EQU     15              ;name length
                        ;lstfcb EQU     fcblen-1
                        ;
0000                    drv     EQU     0               ;drive field
0001                    f1      EQU     1               ;file name byte 1 to 8
                        ;f2     EQU     2
                        ;f3     EQU     3
                        ;f4     EQU     4
0005                    f5      EQU     5
                        ;f6     EQU     6
0007                    f7      EQU     7
0008                    f8      EQU     8
                        ;
                        ;       reserved file indicators
0009                    rofile  EQU     9               ;t1' -> read/only file
000A                    sysfil  EQU     10              ;t2' -> system file
000B                    ARCHIV  EQU     11              ;t3' -> FILE HAS BEEN ARCHIVED
000C                    extnum  EQU     12              ;extent number field
000D                    chksum  EQU     13              ;unfilled bytes field
000E                    modnum  EQU     14              ;data module number
000F                    reccnt  EQU     15              ;record count field
0010                    dskmap  EQU     16              ;disk map field
```

```
 96
 97  =  0020                nxtrec  EQU     fcblen
 98  =  0021                ranrec  EQU     nxtrec+1        ;random record field (3 bytes)
 99  =                      ;
100  =                      ;
101
```

```
102
103  =                          eject ! include system.def
104  =
105  =
106  =                    ;*****************************************************
107  =                    ;*
108  =                    ;*       SYSTEM DEFINITIONS
109  =                    ;*
110  =                    ;*****************************************************
111  =
112  =  FFFF              true        equ  0ffffh     ; value of TRUE
113  =  0000              false       equ  0          ; value of FALSE
114  =                    ;unknown    equ  0          ; value to be filled in
115  =  0080              dskrecl     equ  128        ; log. disk record len
116  =  0008              pnamsiz     equ  8          ; size of process name
117  =  0008              qnamsiz     equ  pnamsiz    ; size of queue name
118  =                    ;fnamsiz    equ  pnamsiz    ; size of file name
119  =                    ;ftypsiz    equ  3          ; size of file type
120  =  00E0              mpmint      equ  224        ; int vec for mpm ent.
121  =                    ;debugint   equ  mpmint+1   ; int vec for debuggers
122  =  0100              ulen        equ  0100h      ; size of uda
123  =  0030              pdlen       equ  030h       ; size of Process Descriptor
124  =                    ;todlen     equ  5          ; size of Time of Day struct
125  =                    ;flag_tick  equ  1          ; flag 0 = tick flag
126  =                    ;flag_sec   equ  2          ; flag 1 = second flag
127  =                    ;flag_min   equ  3          ; flag 2 = minute flag
128  =  00AA              ldtabsiz    equ  0aah       ; ldtablen=11, 10 entries
129
```

```
                        eject | include pd.def


                ;*******************************************************
                ;*
                ;*      Process Descriptor - with the UDA associated
                ;*              with the PD, describes the current
                ;*              state of a Process under MP/M-86
                ;*
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 00|   link    |  thread   |stat |prior|    flag     |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 08|                     Name                      |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 10|    uda    | dsk | user| ldsk|luser|    mem      |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 18|  dvract   |   wait    | org | net |  parent     |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 20| cns |abort| cin |cout | lst | sf3 | sf4 | sf5 |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 28|        reserved       |   pret    |  scratch    |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;*
                ;*      link    - Used for placement into System Lists
                ;*      thread  - link field for Thread List
                ;*      stat    - Current Process activity
                ;*      prior   - priority
                ;*      flag    - process state flags
                ;*      name    - name of process
                ;*      uda     - Segment Adress of User Data Area
                ;*      dsk     - Current default disk
                ;*      user    - Current default user number
                ;*      ldsk    - Disk program loaded from
                ;*      luser   - User number loaded from
                ;*      mem     - pointer to MD list of memory owned
                ;*                by this process
                ;*      dvract  - bit map of currently active drives
                ;*      wait    - parameter field while on System Lists
                ;*      org     - Network node that originated this process
                ;*      net     - Network node running this process
                ;*      parent  - process that created this process
                ;*      cns     - controlling console
                ;*      abort   - abort code
                ;*      cin     - standard file #0 (console input)
                ;*      cout    - standard file #1 (console output)
                ;*      lst     - standard file #2 (list output)
                ;*      sf3     - standard file #3
                ;*      sf4     - standard file #4
                ;*      sf5     - standard file #5
                ;*      reserved- not currently used
                ;*      pret    - return code at termination
                ;*      scratch - scratch word
```

```
183
184  =                      ;*
185  =                      ;*******************************************************
186  =
187  =                      ;p_link         equ     word ptr 0
188  =                      ;p_thread       equ     word ptr p_link + word
189  =                      ;p_stat         equ     byte ptr p_thread + word
190  =                      ;p_prior        equ     byte ptr p_stat + byte
191  =  0006                p_flag          equ     word ptr 6
192  =  0008                p_name          equ     byte ptr 8
193  =  0010                p_uda           equ     word ptr 10h
194  =  0012                p_dsk           equ     byte ptr 12h
195  =  0013                p_user          equ     byte ptr 13h
196  =                      ;p_ldsk         equ     byte ptr p_user + byte
197  =                      ;p_luser        equ     byte ptr p_ldsk + byte
198  =                      ;p_mem          equ     word ptr p_luser + byte
199  =  0018                p_cmod          equ     byte ptr 18h
200  =                      ;p_wait         equ     word ptr p_dvract + word
201  =                      ;p_org          equ     byte ptr p_wait + word
202  =                      ;p_net          equ     byte ptr p_org + byte
203  =                      ;p_parent       equ     word ptr p_net + byte
204  =  0020                p_cns           equ     byte ptr 20h
205  =                      ;p_abort        equ     byte ptr p_cns + byte
206  =                      ;p_cin          equ     byte ptr p_abort + byte
207  =                      ;p_cout         equ     byte ptr p_cin + byte
208  =                      ;p_lst          equ     byte ptr p_cout + byte
209  =                      ;p_sf3          equ     byte ptr p_lst + byte
210  =                      ;p_sf4          equ     byte ptr p_sf3 + byte
211  =                      ;p_sf5          equ     byte ptr p_sf4 + byte
212  =                      ;p_reserved     equ     word ptr p_sf5 + byte
213  =                      ;p_pret         equ     word ptr p_reserved + (2*word)
214  =                      ;p_scratch      equ     byte ptr p_pret + word
215  =                      ;p_wscrtch      equ     word ptr p_scratch
216  =
217  =                      ;
218  =                      ;       Process descriptor pd_status values
219  =                      ;
220  =
221  =                      ;ps_run         equ     00      ; in ready list root
222  =                      ;ps_poll        equ     01      ; in poll list
223  =                      ;ps_delay       equ     02      ; in delay list
224  =                      ;ps_swap        equ     03      ; in swap list
225  =                      ;ps_term        equ     04      ; terminating
226  =                      ;ps_sleep       equ     05      ; sleep processing
227  =                      ;ps_dq          equ     06      ; in dq list
228  =                      ;ps_nq          equ     07      ; in nq list
229  =                      ;ps_flagwait    equ     08      ; in flag table
230  =                      ;ps_ciowait     equ     09      ; in c_queue list
231  =                      ;
232  =                      ;       Process descriptor pd_flag bit values
233  =                      ;
234  =                      ;pf_sys         equ     00001h  ; system process
235  =                      ;pf_keep        equ     00002h  ; do not terminate
```

```
236
237  =                    ;pf_kernal      equ     00004h  ; resident in kernal
238  =                    ;pf_pure        equ     00008h  ; pure memory described
239  =                    ;pf_table       equ     00010h  ; from pd table
240  =                    ;pf_resource    equ     00020h  ; waiting for resource
241  =                    ;pf_raw         equ     00040h  ; raw console i/o
242  =  0080              pf_ctlc         equ     00080h  ; abort pending
243  =                    ;pf_active      equ     00100h  ; active tty
244  =  0200              pf_tempkeep     equ     00200h  ; don't terminate yet...
245  =                    ;pf_ctld        equ     00400h  ; explicit detach occured
246  =                    ;pf_childabort  equ     00800h  ; child terminated abnormally
247
```

```
                        eject | include qd.def


                ;*******************************************************
                ;*
                ;*      Queue Descriptor - This is structure is used
                ;*              to create a queue.  One is maintained
                ;*              in the system data area for each queue
                ;*
                ;*      +----+----+----+----+----+----+----+----+
                ;*   00 |  link   |net |org |  flags  |  name...
                ;*      +----+----+----+----+----+----+----+----+
                ;*   08              ...name             | msglen  |
                ;*      +----+----+----+----+----+----+----+----+
                ;*   10 |  nmsgs  |   dq    |   nq    | msgcnt  |
                ;*      +----+----+----+----+----+----+----+----+
                ;*   18 | msgout  | buffer  |
                ;*      +----+----+----+----+
                ;*
                ;*      link    - used to link QDs is system lists
                ;*      net     - which machine in the network
                ;*      org     - origin machine in the network
                ;*      flags   - Queue Flags
                ;*      name    - Name of Queue
                ;*      msglen  - # of bytes in one message
                ;*      nmsgs   - maximum # of messages in queue
                ;*      dq      - Root of PDs waiting to read
                ;*      nq      - Root of PDs list waiting to write
                ;*      msgcnt  - # of messages currently in queue
                ;*      msgout  - next message # to read
                ;*      buf     - pointer to queue message buffer
                ;*                (for MX queues, owner of queue)
                ;*
                ;*******************************************************

                ;q_link          equ     word ptr 0
                ;q_net           equ     byte ptr q_link + word
                ;q_org           equ     byte ptr q_net + byte
                ;q_flags         equ     word ptr q_org + byte
                ;q_name          equ     byte ptr q_flags + word
                ;q_msglen        equ     word ptr q_name + qnamsiz
                ;q_nmsgs         equ     word ptr q_msglen + word
                ;q_dq            equ     word ptr q_nmsgs + word
                ;q_nq            equ     word ptr q_dq + word
                ;q_msgcnt        equ     word ptr q_nq + word
                ;q_msgout        equ     word ptr q_msgcnt + word
                ;q_buf           equ     word ptr q_msgout + word
                ;qdlen           equ     q_buf + word
                ;
                ;       Q_FLAGS values
                ;
0001            qf_mx           equ     001h    ; Mutual Exclusion
```

```
                        qf_keep         equ     002h    ; NO DELETE
                        ;qf_hide        equ     004h    ; Not User writable
                        ;qf_rsp         equ     008h    ; rsp queue
                        ;qf_table       equu    010h    ; from qd table
                        ;qf_rpl         equ     020h    ; rpl queue
                        ;qf_dev         equ     040h    ; device queue
                        ;*********************************************************
                        ;*
                        ;*      QPB - Queue Parameter Block
                        ;*
                        ;*      +----+----+----+----+----+----+----+----+
                        ;*  00  |flgs|net |  qaddr  |  nmsgs  | buffptr |
                        ;*      +----+----+----+----+----+----+----+----+
                        ;*  08  |                 name                    |
                        ;*      +----+----+----+----+----+----+----+----+
                        ;*
                        ;*      flgs    - unused
                        ;*      net     - unused (which machine to use)
                        ;*      qaddr   - Queue ID, address of QD
                        ;*      nmsgs   - number of messages to read/write
                        ;*      buffptr - address to read/write into/from
                        ;*      name    - name of queue (for open only)
                        ;*
                        ;*********************************************************

                        ;qpb_flgs       equ     byte ptr 0
                        ;qpb_net        equ     byte ptr qpb_flgs + byte
                        ;qpb_qaddr      equ     word ptr qpb_net + byte
                        ;qpb_nmsgs      equ     word ptr qpb_qaddr + word
                        ;qpb_buffptr    equ     word ptr qpb_nmsgs + word
                        ;qpb_name       equ     byte ptr qpb_buffptr + word
                        ;qpblen         equ     qpb_name + qnamsiz
```

```
                        eject | include modfunc.def

                ;*********************************************************
                ;                                                        *
                ;               Sync Parameter Block                     *
                ;                                                        *
                ;*********************************************************

                ;sy_link         equ     word ptr 0
0002            sy_owner        equ     word ptr 2
                ;sy_wait         equ     word ptr 4
                ;sy_next         equ     word ptr 6
                ;synclen         equ     8

                ;*********************************************************
                ;*
                ;*      MP/M-86 Inter-Module Function Definitions
                ;*
                ;*      Same calling conventions as User programs
                ;*      except CX = function instead of CL
                ;*             BX = 2nd parameter on entry
                ;*      (CH=module, CL=function # in module)
                ;*
                ;*********************************************************

                        ; Module definitions

0000            user            equ     0
0001            sup             equ     1
0002            rtm             equ     2
0003            mem             equ     3
0004            cio             equ     4
0005            bdos            equ     5
0006            xios            equ     6
0007            net             equ     7

                        ; Bits that represent present modules
                        ; in module_map

                ;supmod_bit      equ     001h
                ;rtmmod_bit      equ     002h
                ;memmod_bit      equ     004h
                ;bdosmod_bit     equ     008h
                ;ciomod_bit      equ     010h
                ;xiosmod_bit     equ     020h
                ;netmod_bit      equ     040h

                        ; Supervisor Functions

                ;f_sysreset      equ     (user * 0100h) + 0
                ;f_conin         equ     (user * 0100h) + 1
                ;f_conout        equ     (user * 0100h) + 2
```

```
389
390  =                  ;f_rawconin      equ     (user * 0100h) + 3
391  =                  ;f_rawconout     equ     (user * 0100h) + 4
392  =                  ;f_lstout        equ     (user * 0100h) + 5
393  =                  ;f_rawconio      equ     (user * 0100h) + 6
394  =                  ;f_getiobyte     equ     (user * 0100h) + 7
395  =                  ;f_setiobyte     equ     (user * 0100h) + 8
396  =                  ;f_conwrite      equ     (user * 0100h) + 9
397  =                  ;f_conread       equ     (user * 0100h) + 10
398  =  000B            f_constat        equ     (user * 0100h) + 11
399  =                  ;f_getversion    equ     (user * 0100h) + 12
400  =                  ;f_diskreset     equ     (user * 0100h) + 13
401  =                  ;f_diskselect    equ     (user * 0100h) + 14
402  =                  ;f_fopen         equ     (user * 0100h) + 15
403  =                  ;f_fclose        equ     (user * 0100h) + 16
404  =                  ;f_searchfirst   equ     (user * 0100h) + 17
405  =                  ;f_searchnext    equ     (user * 0100h) + 18
406  =                  ;f_fdelete       equ     (user * 0100h) + 19
407  =                  ;f_freadseq      equ     (user * 0100h) + 20
408  =                  ;f_fwriteseq     equ     (user * 0100h) + 21
409  =                  ;f_fmake         equ     (user * 0100h) + 22
410  =                  ;f_frename       equ     (user * 0100h) + 23
411  =                  ;f_loginvector   equ     (user * 0100h) + 24
412  =                  ;f_getdefdisk    equ     (user * 0100h) + 25
413  =                  ;f_setdma        equ     (user * 0100h) + 26
414  =                  ;f_getallocvec   equ     (user * 0100h) + 27
415  =                  ;f_writeprotect  equ     (user * 0100h) + 28
416  =                  ;f_getrovector   equ     (user * 0100h) + 29
417  =                  ;f_setfileattr   equ     (user * 0100h) + 30
418  =                  ;f_getdpb        equ     (user * 0100h) + 31
419  =                  ;f_usercode      equ     (user * 0100h) + 32
420  =                  ;f_freadrda      equ     (user * 0100h) + 33
421  =                  ;f_fwriterdm     equ     (user * 0100h) + 34
422  =                  ;f_filesize      equ     (user * 0100h) + 35
423  =                  ;f_setrndrec     equ     (user * 0100h) + 36
424  =                  ;f_resetdrive    equ     (user * 0100h) + 37
425  =                  ;f_accessdrive   equ     (user * 0100h) + 38
426  =                  ;f_freedrive     equ     (user * 0100h) + 39
427  =                  ;f_writerndzero  equ     (user * 0100h) + 40
428  =                  ;f_callbios      equ     (user * 0100h) + 50
429  =                  ;f_setdmab       equ     (user * 0100h) + 51
430  =                  ;f_getdma        equ     (user * 0100h) + 52
431  =                  ;f_getmaxmem     equ     (user * 0100h) + 53
432  =                  ;f_getabsmaxmem  equ     (user * 0100h) + 54
433  =                  ;f_allocmem      equ     (user * 0100h) + 55
434  =                  ;f_allocabsmem   equ     (user * 0100h) + 56
435  =                  ;f_freemem       equ     (user * 0100h) + 57
436  =                  ;f_freeallmem    equ     (user * 0100h) + 58
437  =                  ;f_userload      equ     (user * 0100h) + 59
438  =                  ;f_malloc        equ     (user * 0100h) + 128
439  =                  ;f_memfree       equ     (user * 0100h) + 130
440  =                  ;f_polldev       equ     (user * 0100h) + 131
441  =                  ;f_flagwait      equ     (user * 0100h) + 132
```

```
                        ;f_flagset      equ     (user * 0100h) + 133
                        ;f_qmake        equ     (user * 0100h) + 134
                        ;f_qopen        equ     (user * 0100h) + 135
                        ;f_qdelete      equu    (user * 0100h) + 136
0089                    f_qread         equ     (user * 0100h) + 137
                        ;f_cqread       equ     (user * 0100h) + 138
008B                    f_qwrite        equ     (user * 0100h) + 139
                        ;f_cqwrite      equ     (user * 0100h) + 140
                        ;f_delay        equ     (user * 0100h) + 141
                        ;f_dispatch     equ     (user * 0100h) + 142
008F                    f_terminate     equ     (user * 0100h) + 143
                        ;f_createproc   equ     (user * 0100h) + 144
                        ;f_setprior     equ     (user * 0100h) + 145
                        ;f_conattach    equ     (user * 0100h) + 146
                        ;f_condetach    equ     (user * 0100h) + 147
                        ;f_setdefcon    equ     (user * 0100h) + 148
                        ;f_conassign    equ     (user * 0100h) + 149
                        ;f_clicmd       equ     (user * 0100h) + 150
                        ;f_callrsp      equ     (user * 0100h) + 151
                        ;f_parsefilename equ    (user * 0100h) + 152
                        ;f_getdefcon    equ     (user * 0100h) + 153
                        ;f_sdataddr     equ     (user * 0100h) + 154
                        ;f_timeofday    equ     (user * 0100h) + 155
                        ;f_pdaddress    equ     (user * 0100h) + 156
                        ;f_abortprocess equ     (user * 0100h) + 157
                        ;f_lstattach    equ     (user * 0100h) + 158
                        ;f_lstdetach    equ     (user * 0100h) + 159
                        ;f_setdeflst    equ     (user * 0100h) + 160
                        ;f_clstattch    equ     (user * 0100h) + 161
00A2                    f_cconattch     equ     (user * 0100h) + 162
                        ;f_mpmvernum    equ     (user * 0100h) + 163
                        ;f_getdeflst    equ     (user * 0100h) + 164

                                ; Internal RTM functions

                        ;f_sleep        equ     (rtm * 0100h) + 18
                        ;f_wakeup       equ     (rtm * 0100h) + 19
                        ;f_findpdname   equ     (rtm * 0100h) + 20
0215                    f_sync          equ     (rtm * 0100h) + 21
0216                    f_unsync        equ     (rtm * 0100h) + 22

                                ; Internal MEM functions

                        ;f_share        equ     (mem * 0100h) + 8
                        ;f_maualloc     equ     (mem * 0100h) + 9
                        ;f_maufree      equ     (mem * 0100h) + 10
                        ;f_mlalloc      equ     (mem * 0100h) + 11
                        ;f_mlfree       equ     (mem * 0100h) + 12

                                ; Internal SUP functions

                        ;f_load         equ     (sup * 0100h) + 10
```

```
495
496   =
497   =                                ; Internal CIO functions
498   =
499   =  040E                  f_conprint      equ     (cio * 0100h) + 14
500
```

```
                        eject 1 include xiosch.def
                ;*****************************************************
                ;*
                ;*      XIOS function jump table offsets
                ;*
                ;*****************************************************
                ;io_const        equ     0
                ;io_conin        equ     1
                ;io_conout       equ     2
                ;io_listst       equ     3               ;not used
                ;io_list         equ     4
                ;io_auxin        equ     5               ;not used
                ;io_auxout       equ     6               ;not used
                ;io_switch       equ     7
                ;io_statline     equ     8
0009            io_seldsk        equ     9
000A            io_read          equ     10
000B            io_write         equ     11
000C            io_flush         equ     12
                ;io_polldev      equ     13
                ;*****************************************************
                ;*
                ;*      XIOS Parameter Block for CALL XIOS functions
                ;*
                ;*****************************************************
0000            xcb_func         equ     0
0001            xcb_cx           equ     word ptr xcb_func + byte
0003            xcb_dx           equ     word ptr xcb_cx + word
0005            xcblen           equ     xcb_dx + word
```

```
                                eject 1 include bdosif.bdo      ; system initialization
                        ;*****************************************************
                        ;*
                        ;*      bdos interface
                        ;*
                        ;*****************************************************
 FFFF                   BMPM    equ     true
 0000                   BCPM    equ     false

                                cseg
                                org     0

0000 E94600        0049         jmp init                ;bdos initialization
0003 E9CE00        0004         jmp entry               ;inter module entry pt.

0006 0000              sysdat          dw      0       ;seg address of sysdat
0008                   supervisor      rw      2       ;supervisor offset and segment

000C                   dev_ver         rw      1       ;set by sysdat.bdo, chk'd by GENSYS

000E 434F50595249              db      'COPYRIGHT(C)1983,'
     474854284329
     313938332C
001F 444947495441              db      'DIGITAL RESEARCH(04/06/83)'
     4C2052455345
     415243482830
     342F30362F38
     3329
0039 585858582D30              db      'XXXX-0000-654321'
     30302D363534
     33323231

                        ;====
                        init:           ; initialize bdos/xios modules
                        ;====
                        ;       entry:  DS = system data area

                        ;       code segment values setup by gensys
                        ;       mov sysdat,ds
                        ;       mov ax,supmod
                        ;       mov cs:supervisor,ax
                        ;       mov ax,supmod+2
                        ;       mov cs:supervisor+2,ax

0049 CB                         retf

                        ;*****************************************************
                        ;*
                        ;* bdos function table
```

```
590
591  =                      ;*
592  =                      ;*****************************************************
593  =
594  =                              ; structure of entry in functab
595  =
596  =  0000                btab_addr       equ     word ptr 0
597  =  0002                btab_flag       equ     byte ptr (btab_addr + word)
598  =
599  =  0001                bf_getmx        equ     00001b          ;get mxdisk queue
600  =  0002                bf_cshell       equ     00010b          ;conditional shell function
601  =  0004                bf_fcb33        equ     00100b          ;33 byte fcb flag
602  =  0008                bf_fcb36        equ     01000b          ;36 byte fcb flag
603  =  0010                bf_resel        equ     10000b          ;disk reselect flag
604  =
605  =                              ; bdos function table
606  =
607  =004A 2D2301           functab dw func13   ! db 00001b ; fxi13  equ 00h ;disk reset
608  =004D 642301                   dw func14   ! db 00001b ; fxi14  equ 01h ;select disk
609  =000002                        dw func15   ! db 10101b ! fxi15  equ 02h ;open file
610  =000003                        dw func16   ! db 10101b ! fxi16  equ 03h ;close file
611  =000004                        dw func17   ! db 00101b ! fxi17  equ 04h ;search first
612  =000005                        dw func18   ! db 00001b ! fxi18  equ 05h ;search next
613  =000006                        dw func19   ! db 10101b ! fxi19  equ 06h ;delete file
614  =000007                        dw func20   ! db 10111b ! fxi20  equ 07h ;read sequential
615  =000008                        dw func21   ! db 10111b ! fxi21  equ 08h ;write sequential
616  =000009                        dw func22   ! db 00101b ! fxi22  equ 09h ;make file
617  =00000A                        dw func23   ! db 10101b ! fxi23  equ 0Ah ;rename file
618  =006B A52D00                   dw func24   ! db 00000b ; fxi24  equ 0Bh ;return login vector
619  =006E AA2D00                   dw func25   ! db 00000b ; fxi25  equ 0Ch ;return current disk
620  =0071 B22D00                   dw func26   ! db 00000b ; fxi26  equ 0Dh ;set dma address
621  =00000E                        dw func27   ! db 00001b ! fxi27  equ 0Eh ;get alloc addr
622  =0077 B72801                   dw func28   ! db 00001b ; fxi28  equ 0Fh ;write protect disk
623  =007A B82D00                   dw func29   ! db 00000b ; fxi29  equ 10h ;get r/o vector
624  =000011                        dw func30   ! db 00101b ! fxi30  equ 11h ;set file attributes
625  =000012                        dw func31   ! db 00001b ! fxi31  equ 12h ;get disk parm addr
626  =0083 BD2D00                   dw func32   ! db 00000b ; fxi32  equ 13h ;set/get user code
627  =000014                        dw func33   ! db 11011b ! fxi33  equ 14h ;read random
628  =000015                        dw func34   ! db 11011b ! fxi34  equ 15h ;write random
629  =000016                        dw func35   ! db 11001b ! fxi35  equ 16h ;compute file size
630  =008F 732909                   dw func36   ! db 01001b ; fxi36  equ 17h ;set random record
631  =0092 832901                   dw func37   ! db 00001b ; fxi37  equ 18h ;reset drive
632  =0095 B72901                   dw func38   ! db 00001b ; fxi38  equ 19h ;access drive
633  =00001A                        dw func39   ! db 00001b ! fxi39  equ 1Ah ;free drive
634  =000018                        dw func40   ! db 11011b ! fxi40  equ 1Bh ;write random w/zero fill
635  =009E 732A19                   dw func42   ! db 11001b ; fxi42  equ 1Ch ;lock record
636  =00A1 7B2A19                   dw func43   ! db 11001b ; fxi43  equ 1Dh ;unlock record
637  =00A4 D12D00                   dw func44   ! db 00000b ; fxi44  equ 1Eh ;set multi-sector count
638  =00A7 E42D00                   dw func45   ! db 00000b ; fxi45  equ 1Fh ;set bdos error mode
639  =00AA 832A01                   dw func46   ! db 00001b ; fxi46  equ 20h ;get disk free space
640  =000021                        dw func48   ! db 00001b ! fxi48  equ 21h ;flush buffers
641  =00B0 EA2D00                   dw func51   ! db 00000b ; fxi51  equ 22h ;set dma base
642  =00B3 F02D00                   dw func52   ! db 00000b ; fxi52  equ 23h ;get dma base
```

```
                        dw func98   ! db 00001b  ! fxi98  equ 24h  ;reset alloc vector
                        dw func99   ! db 01001b  ! fxi99  equ 25h  ;truncate file
                        dw func100  ! db 00101b  ! fxi100 equ 26h  ;set directory label
                        dw func101  ! db 00001b  ; fxi101 equ 27h  ;return directory label data
                        dw func102  ! db 00101b  ! fxi102 equ 28h  ;read file xfcb
                        dw func103  ! db 00101b  ! fxi103 equ 29h  ;write or update file xfcb
                        dw func104  ! db 00000b  ; fxi104 equ 2ah  ;set current date and time
                        dw func105  ! db 00000b  ; fxi105 equ 2bh  ;get current date and time
                        dw func106  ! db 00001b  ; fxi106 equ 2ch  ;set default password
                        dw pr_term  ! db 00001b  ! fxi143 equ 20h  ;terminate process
                ;=====
                entry:          ; bdos module entry point
                ;=====
                ;       entry:  CL = internal bdos function number
                ;               DX = argument
                ;               DS = system data area
                ;               ES = user data area
                ;       exit:   AX = BX = return code
00D4 32ED8BF1                   xor ch,ch! mov si,cx
00D8 D1E603F1                   shl si,1! add si,cx     ; multiply by 3
00DC 83C64A                     add si,offset functab
00DF 2EF684020001               test cs:btab_flag[si],bf_getmx
00E5 7405           00EC        jz nomx
00E7 E80C00         00F6          call getdiskmx
00EA EB07           00F3          jmps exit
                nomx:
00EC 33DB                       xor bx,bx                       ;initialize return parameter
00EE 2EFF940000                 call cs:btab_addr[si]
                exit:
00F3 8BC3                       mov ax,bx
00F5 CB                         retf
                getdiskmx:
                ;---------
                ;       entry:  SI = offset of bdos functab entry
                ;               CX = internal bdos function number
                ;               DX = argument
0F6 881E6800                    mov bx,rlr
0FA 8B4706250002                mov ax,p_flag[bx]! and ax,pf_tempkeep
0100 50                         push ax                         ;save tempkeep flag
0101 814F060002                 or p_flag[bx],pf_tempkeep       ;do not allow ctrl c while in bdos
0106 565251                     push si! push dx! push cx
0109 B9B900BA900B               mov cx,f_qread ! mov dx,offset mxdiskqpb
010F E80803         041A        call mpmif                      ;get mxdisk queue
0112 595A5E                     pop cx! pop dx! pop si
0115 0BC07403       011C        or ax,ax! jz $+5                ;was MXQ read successful?
0119 E9FB01         0317          jmp retmonx1                  ;no
```

```
696
697  =011C 880E7A08                  mov fx_intrn,cl              ;save internal bdos func #
698  =0120 881E6800                  mov bx,rlr
699  =0124 F747068000                test p_flag[bx],pf_ctlc
700  =0129 7408          0133        jz not_ctlc
701  =012b 32C0                        xor al,al                  ;AL = no error message to print
702  =012D BBFFFF                      mov bx,0ffffh
703  =0130 E9E700        021A          jmp retmonxa
704  =                         not_ctlc:
705  =0133 9C58FA                    pushf ! pop ax  ! cli        ;switch to internal bdos stack
706  =0136 8C16380B                  mov ss_save,ss
707  =013A 89263A0B                  mov sp_save,sp
708  =013E 2E8E160600                mov ss,sysdat
709  =0143 BC350B                    mov sp,offset bdosstack
710  =0146 5090                      push ax ! popf
711  =
712  =0148 891EA408                  mov pdaddr,bx                ;BX = rlr
713  =
714  =014C 8B4712                    mov ax,word ptr p_dsk[bx]
715  =014F A37F08                    mov word ptr seldsk,ax       ;set default disk and user code
716  =
717  =0152 26A12E00                  mov ax,u_wrkseg              ;initialize bdos data area for user
718  =0156 A32B09                    mov parametersegment,ax
719  =0159 26A13000                  mov ax,u_retseg
720  =015D A32D09                    mov returnseg,ax
721  =
722  =0160 06                        push es
723  =0161 8C062909                  mov uda_save,es              ;save uda segment
724  =0165 56                        push si
725  =0166 8CD88CC3                  mov ax,ds! mov bx,es         ;copy uda bdos vars to local area
726  =016A 8ED88EC0                  mov ds,bx! mov es,ax
727  =016E BE0200                    mov si,offset u_dma_ofst
728  =0171 BF8508                    mov di,offset dma_ofst
729  =0174 B91900                    mov cx,uda_ovl_len
730  =0177 F3A4                      rep movsb
731  =0179 8ED8                      mov ds,ax
732  =
733  =017B A18508                    mov ax,dma_ofst
734  =017E B104D3E8                  mov cl,4! shr ax,cl
735  =0182 01068708                  add dma_seg,ax
736  =0186 812685080F00              and dma_ofst,000fh
737  =
738  =018C B9150032C0                mov cx,zerolength ! xor al,al ;zero local variables
739  =0191 BF0809                    mov di,offset fcbdsk
740  =0194 F3AA                      rep stosb
741  =
742  =0196 F6069D080174 01A0         test pdcnt,11 jz pdcnt_ok    ;reset pdcnt if low order bit set
743       03            01A0
744  =019D E8891C       1E29           call inc_pdcnt
745  =                         pdcnt_ok:
746  =01A0 89168108                  mov info,dx                  ;info=dx
747  =01A4 88151009                  mov linfo,dl                 ;linfo = low(info) - don't equ
748  =01A8 5E56                      pop si! push si
```

```
                                        mov ah,cs:btab_flag[si]
=01AA 2E8AA40200                        test ah,bf_fcb33! jz notfcb33   ;if 33 byte fcb
=01AF F6C4047405    01B9                  call parsave33! jmps notfcb36 ; copy 33 bytes to local FCB
=01B4 E82D02E80B    03E4              notfcb33:
=                                       test ah,bf_fcb36! jz notfcb36   ;else if 36 byte fcb
=01B9 F6C4087406    01C4                  call parsave36! call save_rr  ; copy 36 bytes to local FCB
=01BE E82702E80F02  03E8              notfcb36:                         ; and save random record bytes
=                                       pop si
=01C4 5E                                cmp mult_cnt,1 ! je noshell     ;if mult_cnt <> 1 and
=01C5 803E94080174  01D9
      0D            01D9
=01CC 2EF6840200C2                      test cs:btab_flag[si],bf_cshell ; func uses mult_cnt then
=01D2 7405          01D9                jz noshell
=01D4 E8A901        0380                  call shell                    ; use bdos multi sector shell
=01D7 EB03          01DC                  jmps retmon
=                                     noshell:
=01D9 E85801        0334                call call_bdos
=                                     retmon:
=01DC 8A0E0C08                          mov cl,parlg
=01E0 0AC9740F      01F3                or cl,cl! jz retmon6
=01E4 32ED                                xor ch,ch                     ;copy local FCB back to user segment?
=01E6 BE3C0B                              mov si,offset info_fcb
=01E9 8B3E8108                            mov di,info
=01ED 8E062809                            mov es,parametersegment
=01F1 F3A4                                rep movsb
=                                     retmon6:
=01F3 07                                pop es                          ;restore uda segment
=01F4 BE89088BF0600                     mov si,offset dma_ofst+4 ! mov di,offset u_dma_ofst+4
=01FA B91500                            mov cx,uda_ovl_len-4
=01FD F3A4                              rep movsb
=
=01FF A12D09                            mov ax,returnseg                ;setup return registers
=0202 26A33000                          mov u_retseg,ax
=0206 8B1E0D08                          mov bx,aret
=
=020A 9C58FA                            pushf ! pop ax ! cli            ;switch back to user's stack
=020D 8E16380B                          mov ss,ss_save
=0211 8B263A0B                          mov sp,sp_save
=0215 5090                              push ax ! popf
=
=0217 A01808                            mov al,err_type
=                                     retmonxa:
=021A 53                                push bx                         ;save return code
=021B A8FF                              test al,true
=021D 741F          023E                jz retmonx
=021F 7815          0236                  js denied_typ
=0221 FF362809                              push parametersegment       ;fcb segment
=0225 BBFFFF                                mov bx,0ffffh               ;0ffffh => no fcb to print in message
=0228 F6060F08FF                            test resel,true
=022D 7404          0233                    jz no_fcb
=022F 8B1E8108                                mov bx,info               ;fcb offset
=                                     no_fcb:
=0233 53                                    push bx
```

```
=0234 EB04          023A            jmps comn_dat
=                           denied_typ:
=0236 FF363009                      push err_pd_addr            ;denied error pd address
=                           comn_dat:
=023A 8A262F09                      mov ah,err_drv
=                           retmonx:
=023E 50                    push ax                             ;AL = error message type to print
=023F B98B00BA9008          mov cx,f_qwrite | mov dx,offset mxdiskqpb
=0245 E8D201        041A    call mpmif                          ;release mxdisk queue
=0248 58                    pop ax
=0249 A8FF                  test al,0ffh
=024B 7503E9C600    0250    jnz $+5| jmp retmonx1a
=0250 50                      push ax
=0251 BBA00B                  mov bx,offset msg_spb
=0254 B91502                  mov cx,f_sync
=0257 E8C001        041A      call mpmif
=
=025A 833E940000              cmp word ptr err_intercept+2,0  ;check if error interception
=025F 7407          0268      je ret_err0                     ; no
=0261 FF1E9200                  callf dword ptr err_intercept ;call intercept routine
=0265 E99F00        0307        jmp free_spb
=                           ret_err0:
=0268 58                    pop ax
=0269 80C441                add ah,'A'
=026C A8FF                  test al,0ffh
=026E 7903E97200    0273    jns $+5| jmp denied_err
=0273 50                      push ax
=0274 88264209                mov dskerr,ah                 ;set D: field
=0278 BA3209                  mov dx,offset dskmsg
=027B E88D01        0408      call xprint                   ;print "CP/M Error On D:"
=
=027E 58                      pop ax
=027F 8AD8                    mov bl,al
=0281 32FF                    xor bh,bh
=0283 D1E3                    shl bx,1
=0285 8B974609                mov dx,xerr_list[bx]          ;compute extended error message offset
=0289 E87F01        0408      call xprint                   ;print error message
=
=028C 26A00600                mov al,u_func                 ;convert func# to character
=0290 B530                    mov ch,30h
=0292 BB5A0A                  mov bx,offset pr_fx1
=0295 3C647206      029F      cmp al,100| jb ret_err1
=0299 C60731                    mov b[bx],31h
=029C 43                        inc bx
=029D 2C64                      sub al,100
=                           ret_err1:
=029F 2C0A7204      02A7      sub al,10| jb ret_err2
=02A3 FEC5                      inc ch
=02A5 EBF8          029F        jmps ret_err1
=                           ret_err2:
=02A7 882F                    mov [bx],ch
=02A9 43                      inc bx
```

```
                                        add al,'a'
=02AA 043A                              mov [bx],al
=02AC 8807                              inc ax
=02AE 43                                mov b[bx],20h
=02AF C60720                            mov bx,offset pr_fcb        ;0 = message delimiter
=02B2 BB5D0A                            mov b[bx],0
=02B5 C60700                            pop si! pop dx              ;DX,SI = offset of fcb
=02B8 5E5A                              inc si                      ;was reselect called?
=02BA 46                                jz ret_err3                 ;no - don't print fcb
=02BB 7420          02DD                  mov b[bx],20h             ;remove delimiter
=02BD C60720                              mov di,offset pr_fcb1
=02C0 BF650A                              mov ax,ds! mov bx,es
=02C3 8CD88CC3                            mov es,ax                 ;ES = DS
=02C7 8EC0                                mov ds,dx                 ;DS = parametersegment
=02C9 8EDA                                mov cx,4                  ;move file name to message
=02CB B90400                              rep movsw
=02CE F3A5                                mov es:b[di],'.'          ;move '.' to message
=02D0 26C6052E                            inc di
=02D4 47                                  mov cl,3                  ;move file type to message
=02D5 B103                                rep movsb
=02D7 F3A4                                mov ds,ax! mov es,bx      ;restore DS,ES
=02D9 8ED88EC3                      ret_err3:
=                                       mov dx,offset pr_fx         ;print "bdos function = ### "
=02DD BA480A                            call xprint                 ; + "file = ffffffff.ttt"
=02E0 E82801        0408                jmps free_spb
=02E3 EB22          0307              denied_err:
=                                       mov denieddrv,ah
=02E5 88268D0A                          pop si
=02E9 5E                                mov al,p_cns[si]
=02EA 8A4420                            add al,'0'
=02ED 0430                              mov deniedcns,al
=02EF A2980A                            add si,p_name
=02F2 83C608                            mov di,offset deniedprc
=02F5 BFA20A                            push es! push ds! pop es    ;ES = DS
=02F8 061E07                            mov cx,4                    ;copy process name to message
=02FB B90400                            rep movsw
=02FE F3A5                              pop es                      ;restore ES
=0300 07                                mov dx,offset deniedmsg
=0301 BA720A                            call xprint
=0304 E80401        0408              free_spb:
=                                       mov dx,offset crlf_str
=0307 BA450A                            call xprint
=030A E8FE00        0408                mov bx,offset msg_spb
=030D BBA00B                            mov cx,f_unsync
=0310 B91602                            call mpmif
=0313 E80401        041A              retmonx1a:
=                                       pop bx                      ;restore retcode
=0316 5B                          retmonx1:
=                                       mov si,rlr
=0317 8B366800                          pop ax                      ;restore tempkeep flag
=031B 58                                or ax,not pf_tempkeep
=031C 0DFFFF                            and p_flag[si],ax
=031F 214406
```

```
908
909  =0322 F744068000              test p_flag[si],pf_ctlc         ;see if control C occured
910  =0327 740A          0333      jz mxdiskexit
911  =0329 B98F0033D2                mov cx,f_terminate! xor dx,dx
912  =032E 53E8E80058    0414        push bx! call mpmif! pop bx
913  =                         mxdiskexit:
914  =0333 C3                        ret
915  =
916  =                         call_bdos:
917  =                         ;---------
918  =                         ;       entry:  DX = argument
919  =                         ;               SI = offset of bdos functab entry
920  =
921  =0334 89263606                  mov save_sp,sp
922  =0338 2EF684020010              test cs:btab_flag[si],bf_resel
923  =033E 7405          0345        jz cbdos1
924  =0340 56                        push si
925  =0341 E8AD16        19F1        call reselect
926  =0344 5E                        pop si
927  =                         cbdos1:
928  =0345 2EFF940000                call cs:btab_addr[si]
929  =                         bdos_return:
930  =034A 803E0F0800                cmp resel,0
931  =034F 742E          037F        je retmon5
932  =0351 803E1108FF75  0358        cmp comp_fcb_cks,true! jne retmon1
933       03             0358
934  =0358 E89103        06EC          call set_chksum_fcb
935  =                         retmon1:
936  =035B A09F08                    mov al,xfcb_read_only
937  =035E BB3C0B                    mov bx,offset info_fcb
938  =0361 084707                    or f7[bx],al
939  =0364 A09E08                    mov al,high_ext
940  =0367 3C607506      0371        cmp al,60h! jne retmon3
941  =036B 804F0880                    or byte ptr f8[bx],80h
942  =036F EB03          0374        jmps retmon4
943  =                         retmon3:
944  =0371 08470C                    or extnum[bx],al
945  =                         retmon4:
946  =0374 A0DA08                    mov al,actual_rc
947  =0377 08470F                    or reccnt[bx],al
948  =037A A00B08                    mov al,fcbdsk
949  =037D 8807                      mov drv[bx],al
950  =                         retmon5:
951  =037F C3                        ret
952  =
953  =                         shell:
954  =                         ;-----
955  =                         ;       entry:  SI = offset of functab entry
956  =
957  =0380 89362409                  mov shell_si,si
958  =0384 2E8AA40200                mov ah,cs:btab_flag[si]
959  =0389 A09408                    mov al,mult_cnt
960  =                         mult_io1:
```

```
 961
 962  =038C A21808                    MOV MULT_NUM,AL
 963  =038F 50                        push ax
 964  =0390 8B362409                  mov si,shell_si
 965  =0394 8B168108                  mov dx,info
 966  =0398 E899FF        0334        call call_bdos
 967  =039b 8A1E0D08                  mov bl,byte ptr aret
 968  =039F 0ADB                      or bl,bl
 969  =03A1 58                        pop ax
 970  =03A2 740D          03B1        jz no_shell_err
 971  =03A4 80FBFF7420    03C9          cmp bl,0ffh! je shell04
 972  =03A9 8A3E9408                    mov bh,mult_cnt
 973  =03AD 2AF8                        sub bh,al                     ;BH = # of successfull records
 974  =03AF EB14          03C5          jmps shell03
 975  =                         no_shell_err:
 976  =03B1 F6C4087403    03B9        test ah,bf_fcb36! jz mult_io2   ;AH = entry functab flags
 977  =03B6 E84700        0400          call incr_rr
 978  =                         mult_io2:
 979  =03B9 810685088000                add dma_ofst,80h
 980  =03BF FEC8                        dec al
 981  =03C1 75C9          038C        jnz mult_io1
 982  =03C3 33DB                      xor bx,bx
 983  =
 984  =                         shell03:
 985  =03C5 891E0D08                    mov aret,bx
 986  =                         shell04:
 987  =03C9 F6C4087431    03FF        test ah,bf_fcb36! jz parret
 988  =                               ;jmps reset_rr
 989  =
 990  =                         reset_rr:
 991  =                               ;--------
 992  =03CE E80C00EB05    03DD        call save_rr2! jmps save_rr1
 993  =                         save_rr:
 994  =                               ;-------
 995  =03D3 E80700870A    03DD        call save_rr2! xchg bx,dx
 996  =                         save_rr1:
 997  =03D8 B103E90501    04E2        mov cl,3! jmp move
 998  =                         save_rr2:
 999  =03DD 8B5D08                    mov bx,offset info_fcb+ranrec
1000  =03E0 8A2609                    mov dx,offset shell_rr
1001  =03E3 C3                        ret
1002  =
1003  =                         parsave33:      ;copy 33 byte length FCB
1004  =                               ;---------
1005  =03E4 B121                      mov cl,33
1006  =03E6 EB02          03EA        jmps parsave
1007  =
1008  =                         parsave36:      ;copy 36 byte length FCB
1009  =                               ;---------
1010  =03E8 B124                      mov cl,36
1011  =
1012  =                         parsave:        ;copy FCB from user segment to bdos segment
1013  =                               ;-------
```

```
1014
1015  =                         ;       entry:  CL = length of FCB to save
1016  =
1017  =03EA 880E0C08                     mov parlg,cl
1018  =03EE 32ED                         xor ch,ch
1019  =03F0 8B368108                     mov si,info
1020  =03F4 BF3C0B                       mov di,offset info_fcb
1021  =03F7 1E                           push ds
1022  =03F8 8E1E2B09                     mov ds,parameterse,ment
1023  =03FC F3A4                         rep movsb
1024  =03FE 1F                           pop ds
1025  =                         parret:
1026  =03FF C3                           ret
1027  =
1028  =                         incr_rr:
1029  =                         ;-------
1030  =                         ;       exit:   AX = preserved
1031  =
1032  =0400 8B5D0B                       mov bx,offset info_fcb+ranrec
1033  =0403 FF077503      040A           inc w[bx]l jnz incr_rr_ret
1034  =0407 FE4702                         inc byte ptr 2[bx]
1035  =                         incr_rr_ret:
1036  =040A C3                           ret
1037  =
1038  =                         xprint:         ;print message at DX
1039  =                         ;------
1040  =                         ;       entry:  DX = offset of message to print
1041  =
1042  =040B 52                           push dx
1043  =040C B9A200                       mov cx,f_cconattch              ;make sure we own console before
1044  =040F E80800        041A           call mpmif                      ; attempting to print message
1045  =0412 5A                           pop dx
1046  =0413 0AC075E8      03FF           or al,al1 jnz parret
1047  =0417 B90E04                         mov cx,f_conprint
1048  =                                    ;jmps mpmif
1049  =
1050  =                                  ;these functions added for mpm interface
1051  =
1052  =                         ;=====
1053  =                         mpmif:          ;call mpm function
1054  =                         ;=====
1055  =041A 88366800                     mov si,rlr
1056  =041E 068E4410                     push es1 mov es,p_uda[si]
1057  =0422 2EFF1E0800                   callf cs:dword ptr supervisor
1058  =0427 07                           pop es
1059  =0428 C3                           ret
1060  =
1061  =                         ;*******************************************************
1062  =                         ;*
1063  =                         ;*       bdos - xios interface
1064  =                         ;*
1065  =                         ;*******************************************************
1066  =
```

```
1067
1068  =                       ;======          ========================
1069  =                       xiosif:          ; xios interface routine
1070  =                       ;======          ========================
1071  =                       ;       entry:   AL = function number
1072  =                       ;                CX = argument 1
1073  =                       ;                DX = argument 2
1074  =                       ;       exit:    AX = BX = output
1075  =
1076  =0429 068E062909                push es! mov es,uda_save
1077  =042E FF1E2800                  callf dword ptr xiosmod
1078  =0432 FC07                      cldl pop es
1079  =0434 C3                        ret
1080  =
1081  =                       ;========        ================================
1082  =                       rwxiosif:        ; disk read/write xios interface
1083  =                       ;========        ================================
1084  =                       ;       entry:   AL = function number
1085  =                       ;       exit:    AX = BX = output
1086  =
1087  =0435 8B161709                  mov dx,arecord
1088  =0439 8A2E1909                  mov ch,byte ptr arecord+2
1089  =043D 8A1EA008                  mov bl,curdsk
1090  =0441 B701                      mov bh,1
1091  =0443 863E2008                  xchg bh,mult_sec        ;BH = multi sector count
1092  =                                                       ; stack on entry to the xios
1093  =0447 53                        push bx                 ;    +C | DRV  | MCNT |
1094  =0448 FF367B08                  push track              ;    +A |    TRACK    |
1095  =044C FF357D08                  push sector             ;    +8 |   SECTOR    |
1096  =0450 FF361E09                  push cur_dma_seg        ;    +6 |   DMA_SEG   |
1097  =0454 FF362009                  push cur_dma            ;    +4 |   DMA_OFF   |
1098  =                                                       ;    +2 |   RET_SEG   |
1099  =                                                       ; SP+0 |   RET_OFF   |
1100  =0458 8E062909                  mov es,uda_save
1101  =045C FF1E2800                  callf dword ptr xiosmod
1102  =0460 83C40A                    add sp,10               ;remove parameters from stack
1103  =0463 FC1E07                    cldl push dsl pop es
1104  =0466 C3                        ret
1105
```

```
1106
1107  =                       eject 1 include file1.bdo       ; file system part 1
1108  =
1109  =
1110  =                       ;*****************************************************************
1111  =                       ;*****************************************************************
1112  =                       ;**                                                             **
1113  =                       ;**   b a s i c   d i s k   o p e r a t i n g   s y s t e m     **
1114  =                       ;**                                                             **
1115  =                       ;*****************************************************************
1116  =                       ;*****************************************************************
1117  =
1118  =                       ;************ bdos file system part 1 ************
1119  =
1120  =                       ;       error message handlers
1121  =
1122  =                       pererror:       ;report permanent error
1123  =                       ;--------
1124  =0467 B401                      mov ah,1
1125  =0469 EB0F          047A        jmps goerr
1126  =
1127  =                       roderror:       ;report read/only disk error
1128  =                       ;--------
1129  =046B B402                      mov ah,2
1130  =046D EB03          047A        jmps goerr
1131  =
1132  =                       roferror:       ;report read/only file error
1133  =                       ;--------
1134  =046F B403                      mov ah,3
1135  =0471 EB07          047A        jmps goerr
1136  =
1137  =                       selerror:       ;report select error
1138  =                       ;--------
1139  =0473 C606A008FF                mov curdsk,0ffh
1140  =0478 B404                      mov ah,4
1141  =                       goerr:
1142  =                       ;-----
1143  =                       ;       entry:  AH = error type
1144  =
1145  =047A B0FF                      mov al,0ffh
1146  =047C A30D08                    mov aret,ax                     ;set aret
1147  =047F 38069308                  cmp error_mode,al               ;if error_mode = 0ffh then
1148  =0483 7541          04C6        jne report_err                  ;report error to user
1149  =
1150  =                       rtn_phy_errs:                           ;return physical error to user
1151  =
1152  =                       if BNPM
1153  =0485 32C0                      xor al,al
1154  =0487 8606F208                  xchg lock_shell,al
1155  =048B 84C0                      test al,al
1156  =048D 7405          0494        JZ rtn_phy0
1157  =048F 8B26F008                    mov sp,lock_sp
1158  =0493 C3                          ret
```

```
1159
1160  =                       rtn_phy0:
1161  =
1162  =                       endif
1163  =0494 A07A08                    mov al,fx_intrn                 ;if func# = 27,31 then
1164  =0497 3C0E7404    049F          cmp al,fx127l je rtn_phy1       ; aret = 0ffffh
1165  =049B 3C127506    04A5          cmp al,fx131l jne goback
1166  =                       rtn_phy1:
1167  =049F C7060D08FFFF                mov aret,0ffffh
1168  =                       goback:
1169  =04A5 8B263608                    mov sp,save_sp
1170  =04A9 E99EFE      034A          jmp bdos_return
1171  =
1172  =                       set_lret1:
1173  =                       ;---------
1174  =04AC B001                        mov al,1
1175  =                       set_lret:
1176  =                       ;--------
1177  =04AE A20D08                      mov lret,al
1178  =                       funcret:
1179  =                       ;-------
1180  =04B1 C3                          ret
1181  =
1182  =                       file_exists:                            ;file already exists
1183  =                       ;-----------
1184  =04B2 B408                        mov ah,8
1185  =
1186  =                       set_aret:
1187  =                       ;--------
1188  =                       ;       entry:  AH = extended error type
1189  =
1190  =04B4 B0FF                        mov al,0ffh
1191  =04B6 A30D08                      mov aret,ax
1192  =04B9 80FC037502  04C0            cmp ah,3l jne set_al
1193  =04BE B40C                          mov ah,12
1194  =                       set_al:
1195  =04C0 38069308                    cmp error_mode,al
1196  =04C4 74DF        04A5            je goback                       ;return physical & extended errors
1197  =
1198  =                       report_err:
1199  =                       ;----------
1200  =                       ;       entry:  AH = error type
1201  =
1202  =04C6 88261B08                    mov err_type,ah
1203  =04CA A07F08                      mov al,seldsk
1204  =04CD A22F09                      mov err_drv,al                  ;error drive
1205  =04D0 803E9308FE                  cmp error_mode,0feh             ;is error mode print &
1206  =                                                                 ;   return errors?
1207  =04D5 74AE        0485            je rtn_phy_errs                 ;yes
1208  =
1209  =                       if BMPM
1210  =04D7 8B366800                    mov si,rlr
1211  =04DB 814C068000                  or p_flag[si],pf_ctlc           ;set process ^c flag
```

```
1212
1213  =                         endif
1214  =                         if BCPM
1215  =                                 or p_flag,pf_ctlc
1216  =                         endif
1217  =04E0 EBA3          0485          jmps rtn_phy_errs
1218  =
1219  =
1220  =                         ;-----------------------------------------------------
1221  =
1222  =                         move:             ;block move data
1223  =                         ;----
1224  =                         ;       entry:    CL = length of data to move
1225  =                         ;                 DX = source offset
1226  =                         ;                 BX = destination offset
1227  =
1228  =04E2 32ED                        xor ch,ch
1229  =04E4 8BF2                        mov si,dx
1230  =04E6 8BFB                        mov di,bx
1231  =04E8 F3A4                        rep movsb
1232  =04EA C3                          ret
1233  =
1234  =                         compare:          ;compare strings
1235  =                         ;-------
1236  =                         ;       entry:    CL = length of strings
1237  =                         ;                 BX,DX = offset of strings
1238  =                         ;       exit:     Z flag set if strings match
1239  =
1240  =04EB 32ED                        xor ch,ch
1241  =04ED 8BF3                        mov si,bx
1242  =04EF 8BFA                        mov di,dx
1243  =04F1 F3A6                        repe cmps al,al
1244  =04F3 C3                          ret
1245  =
1246  =                         seek:             ;seek the track and sector given by arecord
1247  =                         ;----
1248  =04F4 A11709                      mov ax,arecord                  ;compute track/sector
1249  =04F7 33D2                        xor dx,dx
1250  =04F9 8A151909                    mov dl,byte ptr arecord+2
1251  =04FD F7358808                    div sectpt                      ;dx=sector, ax=track
1252  =0501 0306C508                    add ax,offsetv
1253  =0505 A37308                      MOV TRACK,ax                    ;save bios/xios track
1254  =0508 8A0EC708                    MOV CL,PHYSHF
1255  =050C D3EA                        SHR dx,CL                       ;PHY SECTOR = SHR(LOG SECTOR,PHYSHF)
1256  =050E 89167D08                    MOV SECTOR,dx                   ;save bios/xios sector
1257  =0512 C3                          ret
1258  =
1259  =                         ;       utility functions for file access
1260  =
1261  =                         atran:            ;compute actual record address, assuming index called
1262  =                         ;-----
1263  =0513 8A0EBA08                    mov cl,blkshf                   ;shift count to reg cl
1264  =0517 A11709                      mov ax,arecord
```

```
1265
1266  =051A A32308                  mov blk_num,ax              ;save for pre read check
1267  =051D 32FF                    xor bh,bh
1268  =051F 8ADC                    mov bl,ah
1269  =0521 D3E0                    shl ax,cl
1270  =0523 D3E3                    shl bx,cl
1271  =0525 A31A09                  MOV ARECORD1,ax             ;save shifted block #
1272  =0528 93                      xchg ax,bx                  ; for write rand w/zero fill
1273  =0529 A01509                  mov al,vrecord
1274  =052C 2206BB08                and al,blkmsk               ;masked value in al
1275  =0530 A22208                  mov blk_off,al
1276  =0533 08D8                    or bl,al
1277  =0535 891E1709                mov arecord,bx              ;arecord=bx or
1278  =                                                         ;(vrecord and blkmsk)
1279  =0539 88261909                mov byte ptr arecord+2,ah
1280  =053D C3                      ret
1281  =
1282  =                     dmposition:     ;compute disk map position for vrecord
1283  =                     ;---------
1284  =                     ;       exit:   AL = disk map position of vrecord
1285  =
1286  =053E 8A0EBA08                mov cl,blkshf               ;shift count to CL
1287  =0542 8A2E1509                mov ch,vrecord              ;current virtual record to a
1288  =0546 D2E8                    shr ch,cl                   ;CH = shr(vrecord,blkshf)
1289  =                                                         ;   = vrecord/2**(sect/block)
1290  =0548 F6D9                    neg cl
1291  =054A 80C107                  add cl,7                    ;CL = 7 - blkshf
1292  =054D A01409                  mov al,extval               ;extent value and extmsk
1293  =                                                         ;blkshf = 3,4,5,6,7
1294  =                                                         ;CL = 4,3,2,1,0
1295  =                                                         ;shift is 4,3,2,1,0
1296  =0550 D2E0                    shl al,cl                   ;AL = shl(ext and extmsk,7-blkshf)
1297  =0552 02C5                    add al,ch                   ;add the previous
1298  =                                                         ;shr(vrecord,blkshf) value
1299  =                                                         ;AL is one of the following
1300  =                                                         ;values, depending upon alloc
1301  =                                                         ;bks blkshf
1302  =                                                         ;1k   3     v/8 + extval * 16
1303  =                                                         ;2k   4     v/16+ extval * 8
1304  =                                                         ;4k   5     v/32+ extval * 4
1305  =                                                         ;8k   6     v/64+ extval * 2
1306  =                                                         ;16k  7     v/128+extval * 1
1307  =0554 C3                      ret                         ;with dm$position in a
1308  =
1309  =                     getdm:          ;return disk map value from position given by cx
1310  =                     ;-----
1311  =                     ;       entry:  CX = index into disk map
1312  =                     ;       exit:   BX = disk map value at position CX
1313  =
1314  =0555 BB4C0B                  mov bx,offset info_fcb+dskmap
1315  =0558 03D9                    add bx,cx                   ;index by asingle byte value
1316  =055A 803E120900              cmp single,0                ;single byte/map entry?
1317  =055F 7405          0566      jz getdmd                   ;get disk map single byte
```

```
1318
1319  =0561 8A1F                    mov bl,[bx]
1320  =0563 32FF                    xor bh,bh
1321  =0565 C3                      ret                             ;with bx=00bb
1322  =                       getdmd:
1323  =0566 03D9                    add bx,cx                       ;bx=.fcb(dm+1*2)
1324  =0568 8B1F                    mov bx,[bx]                     ;return double precision value
1325  =056A C3                      ret
1326  =
1327  =                       index:          ;compute disk block number from current fcb
1328  =                       ;-----
1329  =                       ;       exit:   BX = disk map value for vrecord in current fcb
1330  =                       ;               Z flag set according to value in BX
1331  =
1332  =056B E8D0FF    053E           call dmposition                 ;0...15 in register al
1333  =056E A21109                    MOV DMINX,AL
1334  =0571 8AC8                      mov cl,al
1335  =0573 32ED                      xor ch,ch
1336  =0575 E8D0FF    0548           call getdm                      ;value to bx
1337  =0578 891E1709                  mov arecord,bx
1338  =057C 0BDB                      or bx,bx
1339  =057E C3                        ret
1340  =
1341  =                           get_atts:
1342  =                           ;--------
1343  =                           ;       get interface attributes (f5' - f8') from fcb
1344  =                           ;       zero interface attribute bits in fcb
1345  =                           ;       exit:   AL = attributes f5' - f8' in high nibble
1346  =
1347  =057F BF4408                    mov di,offset info_fcb+f8
1348  =0582 B90400                    mov cx,4
1349  =0585 32D2                      xor dl,dl
1350  =0587 FD                        std                             ;direction from f8 to f5
1351  =                           get_atts_loop:
1352  =0588 8A05                      mov al,[di]                     ;get character
1353  =058A D0E0                      shl al,1                        ;attribute bit into carry
1354  =058C D0DA                      rcr dl,1                        ; then into DL
1355  =058E D0E8                      shr al,1                        ;clear attribute bit
1356  =0590 AA                        stosb                           ; and restore character
1357  =0591 E2F5      0588            loop get_atts_loop              ;next character
1358  =0593 FC                        cld                             ;restore direction
1359  =0594 8AC2                      mov al,dl
1360  =0596 A2DE08                    mov attributes,al               ;save attributes
1361  =0599 C3                        ret
1362  =
1363  =                           get_dir_ext:
1364  =                           ;-----------
1365  =                           ;       compute directory extent from fcb
1366  =                           ;       scan fcb disk map backwards
1367  =                           ;       upon return dminx = 0 if no blocks are in fcb
1368  =                           ;       exit:   AL = directory extent number
1369  =                           ;               BX = .fcb(extnum)
1370  =
```

```
1371
1372  =059A BB5C0B              mov bx,offset info_fcb+nxtrec   ;BX = .fcb(vrecord)
1373  =059D BA0110              mov dx,1001h                    ;DH = disk map position (rel to 1)
1374  =                                                         ;DL = no blocks switch
1375  =                   get_de1:
1376  =05A0 FECE                mov dh
1377  =05A2 4B                  dec bx                          ;decrement disk map ptr
1378  =05A3 803F00              cmp b[bx],0                     ;is disk map byte non-zero ?
1379  =05A6 7506          05AE  jne get_de2                     ; yes
1380  =05A8 0AF6                  or dh,dh                      ; no - continue scan
1381  =05AA 75F4          05A0    jnz get_de1
1382  =05AC FECA                  dec dl                        ;DL = 0 if no blocks found
1383  =                   get_de2:
1384  =05AE 88161109            mov dminx,dl                    ;dminx = 0 if no blocks in fcb
1385  =05B2 803E1209FF          cmp single,true                 ;are disk block indexes single byte ?
1386  =05B7 8AC6                mov al,dh                       ;al = block offset in disk map
1387  =05B9 7402          05BD  jz get_de3                      ;yes
1388  =05BB D0E8                  shr al,1                      ;divide block offset by 2
1389  =
1390  =                         ;       al = last non-zero block index in fcb (rel to 0)
1391  =                         ;       compute ext offset from last non-zero block index by
1392  =                         ;       shifting block index right 7-blkshf
1393  =
1394  =                   get_de3:
1395  =05BD B107                mov cl,7
1396  =05BF 2A0E8A08            sub cl,blkshf
1397  =05C3 D2E8                shr al,cl                       ;al = ext offset
1398  =
1399  =05C5 8A26BC08            mov ah,extmsk                   ;if ext offset > extmsk then
1400  =05C9 3AE0                cmp ah,al                       ;  continue scan
1401  =05CB 72D3          05A0  jb get_de1
1402  =
1403  =                         ;       dir ext = (fcb ext & (~extmsk) & maxext) | ext offset
1404  =
1405  =05CD BB4808              mov bx,offset info_fcb+extnum   ;bx = .fcb(ext)
1406  =05D0 8A0F                mov cl,[bx]                     ;cl = fcb extent value
1407  =05D2 F6D4                not ah                          ;ah = ~extmsk
1408  =05D4 80E41F              and ah,maxext                   ;ah = ah & maxext
1409  =05D7 22E1                and ah,cl                       ;ah = ah & fcb extent
1410  =05D9 0AC4                or al,ah                        ;al = dir ext
1411  =05DB C3                  ret
1412  =
1413  =                   compext:        ;compare extent# in al with that in cl
1414  =                   ;-------
1415  =                         ;       entry:  AL,CL = extent numbers to compare
1416  =                         ;       exit:   Z flag set if extent numbers match
1417  =                         ;               BX,CX,DX = preserved
1418  =
1419  =05DC 51                  push cx                         ;save cx's original value
1420  =05DD 8A2EBC08            mov ch,extmsk
1421  =05E1 F6D5                not ch                          ;ch has negated form of
1422  =                                                         ;extent mask
1423  =05E3 22C0                and cl,ch                       ;low bits removed from cl
```

```
=05E5 22C5                      and al,ch                       ;low bits removed from al
=05E7 2AC1                      sub al,cl
=05E9 241F                      and al,maxext                   ;set flags
=05EB 59                        pop cx                          ;restore cx
=05EC C3                        ret
=
=                       getfcb:         ;set local variables from currently addressed fcb
=                       ;------
=05ED A05C0B                    mov al,info_fcb+nxtrec
=05F0 A21509                    mov vrecord,al                  ;vrecord=fcb(nxtrec)
=05F3 803E4B0800                cmp info_fcb+reccnt,0
=05F8 7508              0602    jne getfcb0
=05FA E890FF            059A      call get_dir_ext
=05FD 8AC8                        mov cl,al
=05FF E8A30C            12A5      call set_rc
=                       getfcb0:
=0602 A04B0B                    mov al,info_fcb+reccnt
=0605 3C817202          060B    cmp al,81h! jb getfcb1
=0609 B080                        mov al,80h
=                       getfcb1:
=060B A21309                    mov rcount,al                   ;rcount=fcb(reccnt)
=060E A0BC0B                    mov al,extmsk                   ;extent mask to a
=0611 22064B0B                  and al,info_fcb+extnum          ;fcb(extnum) and extmsk
=0615 A21409                    mov extval,al
=0618 C3                        ret
=
=                       setfcb:         ;place local values back into current fcb
=                       ;------
=0619 32C0                      xor al,al
=061B 803E7A0809                cmp fx_intrn,fxi22              ;if func# < make then
=0620 7302              0624    jae setfc1
=0622 FEC0                        inc al                        ; AL=1 - sequential i/o
=                       setfc1:
=0624 02061509                  add al,vrecord
=0628 A25C0B                    mov info_fcb+nxtrec,al          ;fcb(nxtrec)=vrecord+seqio
=062B 803E4B0B80                cmp info_fcb+reccnt,80h
=0630 7306              0638    jnb ret41                       ;dont reset if fcb(rc) > 7fh
=0632 A01309                    mov al,rcount
=0635 A24B0B                    mov info_fcb+reccnt,al          ;fcb(reccnt)=rcount
=0638 C3                ret41:  ret
=
=                       cmpec0:         ;compute checksum
=                       ;------
=                       ;       entry:  BX = offset of string to checksum
=                       ;               CL = number of bytes to checksum
=                       ;               AL = initial value of checksum
=                       ;       exit:   AL = checksum value
=
=0639 32ED                      xor ch,ch
=                       cmpec2:
=063B 0207                      add al,[bx]
=063D 43                        inc bx
```

```
1477
1478  =063E E2F8          0638          loop cmpec2
1479  =0640 C3                          ret                                 ;with checksum in AL
1480  =
1481  =                           cmpecs:         ;compute checksum for current directory buffer
1482  =                           ;------
1483  =                           ;       exit:   AL = checksum value
1484  =
1485  =0641 8B1EAA08                    mov bx,buffa                        ;current directory buffer
1486  =0645 B90400                      mov cx,4                            ;# of directory in buffer
1487  =0648 32E4                        xor ah,ah                           ;clear checksum value
1488  =                           cmpec1:
1489  =064A 5132C0                      push cx! xor al,al
1490  =064D B120                        mov cl,32                           ;size of directory entry
1491  =064F E8E7FF          0639        call cmpec0
1492  =0652 32E0                        xor ah,al
1493  =0654 59                          pop cx
1494  =0655 E2F3          064A          loop cmpec1
1495  =0657 86C4                        xchg al,ah                          ;AL = checksum value
1496  =0659 C3                          ret
1497  =
1498  =                           chek_fcb:
1499  =                           ;--------
1500  =                           ;       exit:   Z flag set if valid checksum
1501  =
1502  =065A 803E9E0860                  cmp high_ext,01100000b              ;does high_ext = 60h
1503  =065F 7505          0666          jne chksum_fcb                      ;no
1504  =0661 32C0                          xor al,al
1505  =0663 A23C0B                        mov info_fcb,al                   ;yes - set fcb(0) to zero
1506  =                                   ;jmps chksum_fcb
1507  =
1508  =                           chksum_fcb:     ;compute checksum for fcb
1509  =                           ;----------
1510  =                           ;       add 1st 12 bytes of fcb + curdsk + high_ext + xfcb_readonly + bbh
1511  =                           ;       exit:   Z flag set if valid checksum
1512  =
1513  =                           if BMPM
1514  =0666 2AC0                        sub al,al
1515  =0668 BB9D08                      mov bx,offset pdcnt
1516  =066B B104                        mov cl,4
1517  =066D E8C9FF          0639        call cmpec0
1518  =0670 0488                        add al,0bbh                         ;add bias
1519  =0672 BB3C0B                      mov bx,offset info_fcb              ;add 1st 12 bytes of fcb
1520  =0675 B10C                        mov cl,12
1521  =0677 E8BFFF          0639        call cmpec0
1522  =067A 43                          inc bx                              ;skip extent
1523  =067B 0207                        add al,[bx]                         ;add s1
1524  =067D 83C303                      add bx,3                            ;skip modnum & reccnt
1525  =0680 B110                        mov cl,16                           ;checksum disk map
1526  =0682 E8B4FF          0639        call cmpec0
1527  =0685 8B1EA808                    mov bx,drvlbla
1528  =0689 024702                      add al,2[bx]                        ;add in login sequence number
1529  =068C 0AC0                        or al,al                            ;zero flag set if checksum valid
```

```
1530
1531  =                       endif
1532  =                       if BCPM
1533  =                               mov bx,drvlbla
1534  =                               mov al,info_fcb+chksum
1535  =                               cmp 2[bx],al
1536  =                       endif
1537  =
1538  =068E C3                ret42:  ret
1539  =
1540  =                       if BMPM
1541  =
1542  =                       get_cmp_mode:
1543  =                       ;------------
1544  =                       ;       exit:   AL = process compatability mode
1545  =                       ;               BX = preserved
1546  =
1547  =068F 8B36A408                  mov si,pdaddr
1548  =0693 8A4418                    mov al,p_cmod[si]
1549  =0696 C3                        ret
1550  =
1551  =                       cond_check_fcb:
1552  =                       ;--------------
1553  =0697 E8F5FF        068F        call get_cmp_mode
1554  =069A 241075F0      068E        and al,10h| jnz ret42           ;if f4' set dont check fcb
1555  =                               ;jmp check_fcb
1556  =
1557  =                       endif
1558  =
1559  =                       check_fcb:
1560  =                       ;---------
1561  =
1562  =                       if BMPM
1563  =069E C606F30800                mov check_fcb_ret,false
1564  =                       check_fcb1:
1565  =
1566  =                       endif
1567  =
1568  =06A3 E884FF        065A        call chek_fcb                   ;compute fcb checksum
1569  =06A6 74E6          068E        jz ret42                        ;valid if zero
1570  =
1571  =                       if BCPM
1572  =                               mov dx,rlog
1573  =                               call test_vector
1574  =                               jz ret42
1575  =                       endif
1576  =                       if BMPM
1577  =06A8 240F                      and al,0fh                      ;is mod(chksum,16) = 0 ?
1578  =06AA 7533          06DF        jnz check_fcb3                  ;no - invalid checksum
1579  =06AC 803E9D0800                cmp pd_cnt,0                    ;is pdcnt = 0 ?
1580  =06B1 742C          06DF        jz check_fcb3                   ;yes - invalid checksum
1581  =06B3 C6050A09FF                  mov byte ptr sdcnt+1,0ffh
1582  =06B8 E8AA00        0765          call chk_inv_fcb
```

```
1583
1584  =06BB 7505          06C2          jne check_fcb2                ;is invalid fcb
1585  =06BD E87F09        103F            call search_name            ; yes - search for directory fcb
1586  =06C0 EB08          06CA            jmps check_fcb25
1587  =                             check_fcb2:
1588  =06C2 C606FB08FF                    mov dont_close,true
1589  =06C7 E8E80C        13B2            call close1                 ; no - attempt partial close
1590  =                             check_fcb25:
1591  =06CA BBDD08                        mov bx,offset lret
1592  =06CD FE07                          inc b[bx]
1593  =06CF 740E          06DF            jz check_fcb3               ;partial close or search failed
1594  =06D1 C60700                          mov b[bx],0               ;zero lret
1595  =06D4 E81916        1CF0            call pack_sdcnt             ;look for file in lock list
1596  =06D7 B505                          mov ch,5
1597  =06D9 E82016        1CFC            call search_olist
1598  =06DC 7501          06DF            jnz check_fcb3              ;not found - invalid checksum
1599  =06DE C3                              ret                       ;found - checksum ok
1600  =                             endif
1601  =
1602  =                             check_fcb3:
1603  =06DF 5B                            pop bx                      ;discard return address
1604  =
1605  =                             if BMPM
1606  =06E0 F606F308FF                    test check_fcb_ret,true
1607  =06E5 7513          06FA            jnz ret45
1608  =                             endif
1609  =
1610  =                             chk_media2:
1611  =06E7 B00A                          mov al,10                   ;10 = checksum error
1612  =06E9 E9C2FD        04AE            jmp set_lret
1613  =
1614  =                             if BMPM
1615  =
1616  =                             set_chksum_fcb: ;validate fcb checksum
1617  =                             ;--------------
1618  =06EC E877FF        0666            call chksum_fcb             ;compute fcb checksum
1619  =06EF 7409          06FA            jz ret45                    ;return if valid
1620  =06F1 2A054908                      sub al,info_fcb+chksum      ;subtract from checksum value
1621  =06F5 F6D8                          neg al                      ;negate result
1622  =06F7 A24908                        mov info_fcb+chksum,al      ;restore s1
1623  =06FA C3                      ret45:  ret
1624  =
1625  =                             reset_checksum_fcb:     ;invalidate fcb checksum
1626  =                             ;------------------
1627  =06FB C606110800                    mov comp_fcb_cks,0
1628  =0700 E863FF        0666            call chksum_fcb             ;compute fcb checksum
1629  =0703 75F5          06FA            jnz ret45                   ;return if invalid
1630  =0705 FE054908                      inc byte ptr info_fcb+chksum ;invalidate s1
1631  =0709 C3                            ret
1632  =
1633  =                             endif
1634  =                             if BCPM
1635  =
```

```
1636
1637  =                     SET_LSN:
1638  =                     ;-------
1639  =                             MOV BX,DRVLBLA
1640  =                             MOV CL,?[BX]
1641  =                             mov info_fcb+chksum,cl
1642  =                             RET
1643  =                     endif
1644  =
1645  =                     setcdisk:       ;set a "1" value in curdsk position of [bx]
1646  =                     ;--------
1647  =                     ;       entry:  BX = offset of vector to set
1648  =
1649  =070A 8A0EA008                mov cl,curdsk
1650  =
1651  =                     set_cdisk1:     ;set a "1" value in cl position of [bx]
1652  =                     ;----------
1653  =                     ;       entry:  BX = offset of vector to set
1654  =                     ;               CL = position in vector to set
1655  =                     ;       exit:   AX = bit set in position CL
1656  =
1657  =070E B80100                  mov ax,1                        ;number to shift
1658  =0711 D3E0                    shl ax,cl                       ;ax = mask to integrate
1659  =0713 0907                    or [bx],ax                      ;[bx] = mask or rol(1,curdsk)
1660  =0715 C3                      ret
1661  =
1662  =                     nowrite:        ;check if disk is marked read/only
1663  =                     ;-------
1664  =                     ;       exit:   Z flag reset if disk is read/only.
1665  =
1666  =0716 8B150608                mov dx,rodsk
1667  =
1668  =                     test_vector:    ;test current disk bit in vector
1669  =                     ;-----------
1670  =                     ;       entry:  DX = vector to test
1671  =                     ;       exit:   DL = curdsk vector bit (in low order bit)
1672  =                     ;               Z flag reset if bit is on
1673  =
1674  =071A 8A0EA008                mov cl,curdsk
1675  =                     test_vector1:
1676  =071E D3EA                    shr dx,cl
1677  =0720 81E20100                and dx,1
1678  =0724 C3                      ret                             ;non zero if nowrite
1679  =
1680  =                     get_dptra:
1681  =                     ;---------
1682  =                     ; compute the address of a directory element at
1683  =                     ;  positon dptr in the buffer
1684  =                     ;       exit:   BX = buffa + dptr
1685  =                     ;               CX = preserved
1686  =
1687  =0725 8A1E2209                mov bl,dptr
1688  =0729 32FF                    xor bh,bh
```

```
1689
1690  =072B 031EAA08                add bx,buffa            ;bx = buffa + dptr
1691  =072F C3                      ret
1692  =
1693  =                       ro_test:
1694  =                       ;-------
1695  =                       ;       entry:  BX = offset of fcb to test
1696  =                       ;       exit:   BX = .fcb(9)
1697  =                       ;               C flag set if file is read/only
1698  =
1699  =0730 83C309                   add bx,rofile                   ;offset to r/o bit
1700  =0733 8A07                     mov al,[bx]
1701  =0735 D0D0                     rcl al,1
1702  =0737 C3                ret5:  ret
1703  =
1704  =                       ckrodir:        ;check current directory element for read/only status
1705  =                       ;-------
1706  =0738 E8EAFF       0725         call get_dptra                  ;address of element
1707  =
1708  =                       ckrofile:       ;check current buff(dptr) or fcb(0) for r/o status
1709  =                       ;--------
1710  =                       ;       entry:  BX = offset of fcb to test
1711  =
1712  =073B E8F2FF       0730         call ro_test
1713  =073E 73F7         0737         jnc ret5                        ;rnc
1714  =0740 E92CFD       046F         jmp roferror
1715  =
1716  =                       checkwrite:     ;check for write protected disk
1717  =                       ;----------
1718  =0743 E8D0FF       0716         call nowrite
1719  =0746 74EF         0737         jz ret5                         ;rz
1720  =0748 E920FD       046B         jmp roderror
1721  =
1722  =                       getmodnum:
1723  =                       ;---------
1724  =                       ;       compute the address of the module number
1725  =                       ;       bring module number to accumulator
1726  =                       ;       high order bit is fwf (file write flag)
1727  =                       ;       exit:   BX = offset fcb(modnum)
1728  =                       ;               AL = fcb(modnum)
1729  =
1730  =074B BB4A0B                   mov bx,offset info_fcb+modnum
1731  =074E 8A07                     mov al,[bx]
1732  =0750 C3                       ret                             ;al=fcb(modnum)
1733  =
1734  =                       clrmodnum:      ;clear the module number field for user open/make
1735  =                       ;---------
1736  =0751 C6054A0B00               mov info_fcb+modnum,0           ;fcb(modnum)=0
1737  =0756 C3                       ret
1738  =
1739  =                       clr_ext:
1740  =                       ;-------
1741  =0757 8026480B1F               and info_fcb+extnum,1fh
```

```
1742
1743  =075C C3                          ret
1744  =
1745  =                           setfwf:         ;set file write flag
1746  =                           ;------
1747  =                           ;       exit:   BX = .fcb(modnum)
1748  =                           ;               AL = fcb(modnum)
1749  =
1750  =075D E8E8FF        0748           call getmodnum                  ;bx=.fcb(modnum),
1751  =                                                                  ;al=fcb(modnum)
1752  =                                                                  ;set fwf(file write flag) to 1
1753  =
1754  =0760 0C80                         or al,fwfmsk
1755  =0762 8807                         mov [bx],al                     ;fcb(modnum)=fcb(modnum) + 80h
1756  =                                                                  ;also returns non zero
1757  =                                                                  ;in accumulator
1758  =0764 C3                           ret
1759  =
1760  =                           chk_inv_fcb:    ;check for invalid fcb
1761  =                           ;-----------
1762  =0765 BB4C08                       mov bx,offset info_fcb+dskmap
1763  =0768 EB0E          0778           jmps test_ffff
1764  =
1765  =                           tst_inv_fcb:    ;test for invalid fcb
1766  =                           ;-----------
1767  =076A E8F8FF        0765           call chk_inv_fcb
1768  =076D 750C          077B           jnz ret8
1769  =076F 5B                           pop bx
1770  =0770 B009                         mov al,9
1771  =0772 E939FD        04AE           jmp set_lret
1772  =
1773  =                           endofdir:       ;check if end of directory (dcnt = 0ffffh)
1774  =                           ;--------
1775  =                           ;       exit:   Z flag set if at end of directory
1776  =
1777  =0775 BB8F08                       mov bx,offset dcnt
1778  =
1779  =                           test_ffff:
1780  =                           ;---------
1781  =0778 833FFF                       cmp w[bx],0ffffh
1782  =077B C3                    ret8:  ret
1783  =
1784  =                           setenddir:      ;set dcnt to the end of directory (dcnt = 0ffffh)
1785  =                           ;---------
1786  =077C C7068F08FFFF                 mov dcnt,enddir
1787  =0782 C3                           ret
1788  =
1789  =                           set_dcnt:
1790  =                           ;--------
1791  =0783 A11508                       mov ax,xdcnt
1792  =0786 24FC                         and al,0fch
1793  =0788 48                           dec ax
1794  =0789 A38F08                       mov dcnt,ax
```

```
1795
1796  =078C C3                          ret
1797  =
1798  =                          compcdr:        ;return cy if cdrmax > dcnt
1799  =                          ;-------
1800  =                          ;       exit:   C flag set if current dir max > dir count
1801  =                          ;               BX = .cdrmax
1802  =                          ;               DX = dcnt
1803  =
1804  =078D 8B168F08                     mov dx,dcnt                     ;dx = directory counter
1805  =0791 8B1EA608                     mov bx,cdrmaxa                  ;bx=.cdrmax
1806  =0795 3B17                         cmp dx,[bx]
1807  =                                                                  ;condition dcnt - cdrmax
1808  =0797 C3                   ret6:   ret                             ;produces cy if cdrmax>dcnt
1809  =
1810  =                          setcdr:         ;if not (cdrmax > dcnt) then cdrmax = dcnt+1
1811  =                          ;------
1812  =0798 E8F2FF        0780          call compcdr
1813  =079B 72FA          0797          jb ret6                         ;return if cdrmax > dcnt
1814  =                                                                  ;otherwise, bx = .cdrmax+1,
1815  =                                                                  ;dx = dcnt
1816  =079D 42                          inc dx
1817  =079E 8917                        mov [bx],dx
1818  =07A0 C3                          ret
1819  =
1820  =                          TST_LOG_FXS:
1821  =                          ;-----------
1822  =                          ;       exit:   Z flag set if removable and func# = 15,17,19,etc.
1823  =
1824  =07A1 F606C40880                  TEST BYTE PTR CHKSIZ+1,80H      ;IS DRIVE PERMANENT?
1825  =07A6 75EF          0797          JNZ RET6                        ;YES - RET WITH Z FLAG RESET
1826  =07A8 BF5008                      MOV DI,OFFSET LOG_FXS           ;RETURN WITH Z FLAG SET
1827  =                          tst_log_01:
1828  =07AB 8A0D                        mov cl,[di]
1829  =07AD 4732ED                      inc dil xor ch,ch
1830  =07B0 A07A08                      MOV AL,fx_intrn                 ;IF func# = 15,17,19,etc.
1831  =07B3 F2AE                        REPNE SCAS AL
1832  =07B5 C3                          RET
1833  =
1834  =                          chk_exit_fxs:
1835  =                          ;------------
1836  =07B6 BRA504                      mov bx,offset goback
1837  =07B9 53                          push bx                         ;setup for error return
1838  =07BA BF6008                      mov di,offset rw_fxs
1839  =07BD E8EBFF        07AB          call tst_log_01                 ;does fx = read or write ?
1840  =07C0 7503          07C5          jnz $+5                         ; and is drive removable ?
1841  =07C2 E922FF        06E7            jmp chk_media2
1842  =07C5 BF6A08                      mov di,offset sc_fxs
1843  =07C8 E8E0FF        07AB          call tst_log_01                 ;does fx = searchn or close ?
1844  =07CB 7503          07D0          jnz $+5                         ; and is drive removable ?
1845  =07CD E9DF08        10AF            jmp lret_eq_ff
1846  =07D0 5B                          pop bx                          ;remove error return address
1847  =07D1 C3                          ret
```

```
1848
1849  =
1850  =                         TST_RELOG:
1851  =                         ;---------
1852  =07D2 32C0                        xor al,al
1853  =07D4 86062108                    xchg relog,al
1854  =07D8 84C0                        test al,al                      ;IF RELOG <> 0 THEN RELOG
1855  =07DA 74BB          0797          JZ RET6                         ;CURRENT DRIVE
1856  =                         drv_relog:
1857  =07DC E8D711        1986          CALL curselect
1858  =07DF 33C0                        xor ax,ax
1859  =07E1 A38F08                      MOV DCNT,ax                     ;SET DCNT & DPTP TO ZERO TO
1860  =07E4 A22209                      MOV DPTR,al                     ;FORCE SEARCHES TO BEGINNING
1861  =07E7 C3                          RET                             ;OF DIRECTORY
1862  =
1863  =                         wrbuff:         ;write buffer and check condition
1864  =                         ;------
1865  =                         ;       entry:  CL = wrtype = 0 - normal write operation
1866  =                         ;                    wrtype = 1 => directory write operation
1867  =                         ;                    wrtype = 2 => start of new block
1868  =
1869  =07E8 B008                        mov al,io_write
1870  =07EA E848FC        0435          call rwxiosif                   ;current drive, track, ...
1871  =07ED 84FF                        mov ah,true                     ;mark as a write operation
1872  =07EF EB07          07F8          jmps diocomp                    ;check for i/o errors
1873  =
1874  =                         rdbuff:         ;read buffer and check if ok
1875  =                         ;------
1876  =07F1 B00A                        mov al,io_read
1877  =07F3 E83FFC        0435          call rwxiosif                   ;current drive, track,....
1878  =                         diocomp0:
1879  =07F6 B400                        mov ah,false                    ;not a write operation
1880  =
1881  =                         diocomp:        ;check for disk errors
1882  =                         ;-------
1883  =                         ;       entry:  AL = xios return code
1884  =                         ;               AH = true if coming a from write operation
1885  =
1886  =07F8 0AC07506      0802          or al,all jnz dioc
1887  =07FC C6061A08FF                    mov ff_flag,true
1888  =0801 C3                            ret
1889  =                         dioc:
1890  =0802 50                          push ax                         ;save write flag
1891  =0803 3CFF7541      0848          cmp al,0ffh! jne dioc2          ;has media changed
1892  =0807 813EC3080080                cmp chksiz,8000h                ; and is drive removable or
1893  =080D 7439          0848          je dioc2                        ; checksummed ?
1894  =080F 88160808                      mov dx,dlog                   ;  yes
1895  =0813 E804FF        071A            call test_vector              ;in initialize ?
1896  =0816 7430          0848            jz dioc2                      ; yes - treat as perm error
1897  =0818 E87204        0C8D            call media_change
1898  =081B 803E7A0821                    cmp fx_intrn,fxi48            ;does func = flush buffer
1899  =0820 740E          0830            je dioc0                      ; yes
1900  =0822 A01509                        mov al,adrive                 ;is operation on
```

```
1901
1902  =0825 3A057F08                    cmp al,selusk                   ; different drive?
1903  =0829 7407            0832        je dioc1                        ;  no
1904  =082B C606210800                    mov relog,0                   ;no relog for different drive
1905  =                           dioc0:
1906  =0830 58                           pop ax
1907  =0831 C3                           ret
1908  =                           dioc1:
1909  =0832 E881FF          0786        call chk_exit_fxs
1910  =0835 F6051A08FF                   test ff_flag,true               ;if not first disk operation
1911  =083A 7406            0842        jz dioc15
1912  =083C E87008          10AF          call lret_eq_ff               ; yes - return error = 0ffh
1913  =083F E963FC          04A5          jmp goback
1914  =                           dioc15:
1915  =0842 58                           pop ax                          ;restore write flag
1916  =0843 0AE47509        0850        or ah,ah! jnz dioc3             ; if write treat as r/o error
1917  =0847 C3                           ret
1918  =                           dioc2:
1919  =0848 58                         pop ax                            ;discard write flag
1920  =0849 3C027403        0850       cmp al,2! je dioc3                ;is error read/only
1921  =084D E917FC          0467         jmp pererror                    ; no
1922  =                           dioc3:
1923  =0850 E918FC          046B       jmp roderror
1924  =
1925  =                           discard_data:
1926  =                           ;------------
1927  =0853 8B1EB408                     mov bx,dat_bcba
1928  =0857 B104                         mov cl,4
1929  =0859 EB06            0861         jmps discard0
1930  =
1931  =                           discard_dir:    ;discard matching dir bcb's
1932  =                           ;-----------
1933  =085B 8B1EB208                     mov bx,dir_bcba
1934  =
1935  =                           DISCARD:                                ;DISCARD BCB'S MATCHING
1936  =                           ;-------
1937  =085F B101                         MOV CL,1                        ;ADRIVE FOR CL BYTES
1938  =
1939  =                           DISCARD0:
1940  =                           ;--------
1941  =                           ;       entry:  BX = bcb root address
1942  =                           ;               CL = length of bytes to match
1943  =
1944  =0861 0BDB                         or bx,bx                        ;no entries in bcb list
1945  =0863 7416            087B         JZ RET7A                        ;RET IF ZERO
1946  =0865 8B1F                         MOV BX,[BX]
1947  =                           DISCARD1:
1948  =0867 51                           PUSH CX
1949  =0868 BA1609                       MOV DX,OFFSET ADRIVE            ;COMPARE BCB TO ADRIVE FOR
1950  =086B E87DFC          04E8         CALL COMPARE                    ;CL BYTES
1951  =086E 59                           POP CX
1952  =086F 7503            0874         JNZ DISCARD2
1953  =0871 C607FF                       MOV B[BX],0FFH                  ;DISCARD BCB
```

```
1954
1955  =                         DISCARD2:
1956  =0874 8B5F0C                      MOV BX,12[BX]                   ;ADVANCE TO NEXT BCB
1957  =0877 0BD3                        or bx,bx                        ;IS IT LAST BCB?
1958  =0879 75EC          0867          JNZ DISCARD1                    ;NO
1959  =087B C3            RET7A:        RET
1960  =
1961  =                         DEACTIVATE:     ;deactivate bcb's in list
1962  =                         ;----------
1963  =                         ;       entry:  BX = bcb root address
1964  =
1965  =087C 0BDB                      or bx,bx                        ;DEACTIVATE BCB'S IN LIST BX
1966  =087E 74F8          0878          JZ RET7A                        ;BELONGING TO CURRENT PROCESS
1967  =0880 8A1F                      MOV BX,[BX]                     ;AND DRIVE
1968  =                         DEACT1:
1969  =0882 8A07                      MOV AL,[BX]                     ;IS BCB FOR CURRENT DRIVE?
1970  =0884 3A061609                  CMP AL,ADRIVE
1971  =0888 751C          0886        JNZ DEACT2                      ;NO
1972  =
1973  =                         if BMPM
1974  =088A 88470E                      MOV AX,14[BX]                   ;DOES CURRENT PROCESS OWN BCB?
1975  =088D 3B056800                    CMP AX,RLR
1976  =0891 7513          08A6          JNZ DEACT2                      ;NO
1977  =                         endif
1978  =
1979  =0893 C7470E0000                    MOV WORD PTR 14[BX],0         ;DEACTIVATE BCB
1980  =0898 A07A08                        mov al,fx_intrn
1981  =0898 3C217404      08A3            cmp al,fxi48! je deact11      ;IS func# = flush or free drive ?
1982  =089F 3C1A7503      08A6            cmp al,fxi39! jne deact2
1983  =                         deact11:
1984  =08A3 C607FF                        MOV B[BX],0FFH                ;yes - DISCARD BCB
1985  =                         DEACT2:
1986  =08A6 8B5F0C                      MOV BX,12[BX]           ;ADVANCE TO NEXT BCB
1987  =08A9 0BDB                        or bx,bx
1988  =08AB 75D5          0882        JNZ DEACT1
1989  =08AD C3                        RET
1990  =
1991  =                         ;       BLOCKING/DEBLOCKING BUFFER CONTROL BLOCK (BCB) FORMAT
1992  =
1993  =                         ;           +-------+-------+-------+-------+-------+-------+
1994  =                         ;       00h |  DRV  |        RECORD         | PEND  |  SEQ  |
1995  =                         ;           +-------+-------+-------+-------+-------+-------+
1996  =                         ;       06h |      TRACK    |     SECTOR    |  BUFFER_ADDR  |
1997  =                         ;           +-------+-------+-------+-------+-------+-------+
1998  =                         ;       0Ch |      LINK     |  PROCESS_OFF  |
1999  =                         ;           +-------+-------+-------+-------+
2000  =
2001  =                         GET_BCBA:       ; get buffer control block address
2002  =                         ;--------
2003  =                         ;       entry:  BX = .bcb list root
2004  =                         ;       exit:   BX = .bcb       SEQ,LINK,PD fields updated
2005  =
2006  =08AE 891E7308                    MOV ROOT_BCBA,BX
```

```
2007
2008  =08B2 8BFB                    mov di,bx
2009  =08B4 83EF0C                  sub di,12                 ;DI = last bcb (previous curbcb)
2010  =08B7 8B1F                    mov bx,[bx]               ;BX = 1st curbcb
2011  =08B9 837F0C00                cmp word ptr 12[bx],0     ;IS THERE ONLY 1 BCB IN BCB LIST?
2012  =08BD 7503          08C2      jnz $+5                   ;YES
2013  =08BF E90301        09C5        jmp get_bcb35
2014  =08C2 33C0                    xor ax,ax
2015  =08C4 A37508                  MOV EMPTY_BCBA,ax
2016  =
2017  =                         if BMPM
2018  =08C7 A37708                  mov p_last_bcba,ax
2019  =08CA A27908                  MOV P_BCB_CNT,al
2020  =                         endif
2021  =
2022  =                         GET_BCB1:
2023  =08CD 803FFF                  CMP BCBX],0FFH            ;IS CURBCB DISCARDED?
2024  =
2025  =                         if BMPM
2026  =08D0 7417          08E9      je get_bcb10              ;yes
2027  =08D2 8B470E                    mov ax,14[bx]
2028  =08D5 08C0                      or ax,ax                ;IS BCB INACTIVE?
2029  =08D7 751A          08F3      JNZ get_bcb11             ;NO
2030  =08D9 8B367508                  MOV SI,EMPTY_BCBA
2031  =08DD 08F6                      OR SI,SI                ;IS EMPTY_BCBA = 0?
2032  =08DF 7408          08E9      JZ get_bcb10              ;YES - SAVE INACTIVE BCB ADDR
2033  =08E1 8B740C                    MOV SI,12[SI]
2034  =08E4 803CFF                    CMP BC[SI],0FFH         ;is empty_bcb a discarded bcb?
2035  =08E7 7418          0901        JZ get_bcb12            ;yes - DONT SAVE INACTIVE BCB ADDR
2036  =                         endif
2037  =                         if BCPM
2038  =                             jne get_bcb12
2039  =                         endif
2040  =
2041  =                         GET_BCB10:
2042  =08E9 893E7508                MOV EMPTY_BCBA,di         ;EMPTY_BCBA = LAST_BCBA
2043  =08ED C6470500                mov byte ptr 5[bx],0      ;reset sequence counter
2044  =
2045  =                         if BMPM
2046  =08F1 EB0E          0901      JMPS get_bcb12
2047  =                         GET_BCB11:
2048  =08F3 3B056800                CMP AX,RLR                ;DOES CURRENT PROCESS OWN BCB?
2049  =08F7 7508          0901      jne get_bcb12             ;NO
2050  =08F9 FE067908                  INC P_BCB_CNT           ;P_BCB_CNT = P_BCB_CNT + 1
2051  =08FD 893E7708                  MOV P_LAST_BCBA,di
2052  =                         endif
2053  =
2054  =                         GET_BCB12:
2055  =0901 891E7108                MOV CUR_BCBA,BX
2056  =0905 57                      push di
2057  =0906 E8D402        0BDD      CALL DEBLOCK9             ;BX=CURBCBA, DX=.ADRIVE, CL=4
2058  =0909 E8DFF8        04EB      CALL COMPARE              ;DOES CURBCB(0-3) = ADRIVE || ARECORD
2059  =090C 5F                      pop di
```

```
2060
2061  =0900 8B1E7108                    mov bx,cur_bcba
2062  =0911 751D            0930        JNZ get_bcb15                   ;NO
2063  =0913 8A4705                        mov al,5[bx]
2064  =0916 3CFF7413        092D          cmp al,0ffh| je get_bcb14     ;does bcb(5) = 0ffh? (not seq)
2065  =091A 8A267008                      mov ah,phy_off
2066  =091E 3AC47408        092D          CMP al,ah| JZ GET_BCB14       ;DOES BCB(5) = PHY_OFF? (same record)
2067  =0922 FEC0                          INC al                        ;INCR BCB SEQUENCE
2068  =0924 3AC47402        092A          CMP AL,ah| JZ GET_BCB13       ;DOES BCB(5)+1 = PHY_OFF? (next record)
2069  =0928 B0FF                          mov al,0ffh                   ;no - mark bcb as not seq
2070  =                           GET_BCB13:
2071  =092A 884705                        mov 5[bx],al
2072  =                           GET_BCB14:
2073  =092D E98300          0983        JMP GET_BCB3                    ;MOVE CUR BCB TO HEAD OF BCB LIST
2074  =
2075  =                           GET_BCB15:
2076  =
2077  =                           if BMPM
2078  =0930 8B470E                      mov ax,14[bx]
2079  =0933 3B056800                    cmp ax,rlr                      ;DOES CURRENT PROCESS OWN BCB?
2080  =0937 7538            0971        JNZ GET_BCB17                   ;NO
2081  =                           endif
2082  =
2083  =0939 A01609                        mov al,adrive
2084  =093C 3A077531        0971          cmp al,[bx]| jne get_bcb17    ;does the drive match?
2085  =0940 A0C808                        MOV AL,phymsk
2086  =0943 0AC0742A        0971          or al,al| jz get_bcb17        ;does phymsk = 0?
2087  =0947 3A4705                        CMP AL,5[bx]                  ;DOES BCB(5) = PHYMSK?
2088  =094A 7525            0971          jne GET_BCB17                 ;NO
2089  =094C C6470500                      MOV byte ptr 5[BX],0          ;yes - seq bcb - bcb(5) = 0
2090  =                                                                 ;  move bcb to end of list
2091  =0950 837F0C00                      cmp word ptr 12[bx],0         ;IS CUR BCB ALREADY AT END OF LIST?
2092  =0954 7429            097F          JZ GET_BCB2                   ;YES
2093  =
2094  =                           if BMPM
2095  =0956 FE0E7908                      DEC P_BCB_CNT
2096  =                           endif
2097  =
2098  =095A 33C0                         xor ax,ax                     ;BCB(LINK) = 0
2099  =095C 87470C                        xchg ax,12[bx]                ;remove cur bcb from bcb list
2100  =095F 89450C                        mov 12[di],ax
2101  =0962 93                            xchg ax,bx                    ;place cur bcb at end of bcb list
2102  =                           GET_BCB16:
2103  =0963 8BF3                          mov si,bx
2104  =0965 8B5C0C                        mov bx,12[si]
2105  =0968 0BDB75F7        0963          or bx,bx| jnz get_bcb16
2106  =096C 89440C                        mov 12[si],ax
2107  =096F EB08            0979          jmps get_bcb18
2108  =
2109  =                           GET_BCB17:
2110  =0971 837F0C00                      cmp word ptr 12[bx],0         ;is bcb at end of list
2111  =0975 7408            097F          je get_bcb2                   ; yes
2112  =0977 8BFB                          mov di,bx                     ;DI = new last bcb
```

```
2113
2114  =                       GET_BCB18:
2115  =0979 8B5D0C                    mov bx,12[di]           ;BX = new cur bcb
2116  =097C E94EFF      08CD          JMP GET_BCB1
2117  =
2118  =                       GET_BCB2:                       ;END OF BCB SCAN - NO MATCH
2119  =
2120  =                       if BMPM
2121  =097F 8B470E                  mov ax,14[bx]           ;DOES LAST BCB BELONG TO
2122  =0982 3B056800                CMP AX,RLR              ;CURRENT PROCESS?
2123  =0986 7422        09AA        JE GET_BCB26            ;YES - USE IT
2124  =                       endif
2125  =
2126  =0988 8B367508                MOV si,EMPTY_BCBA       ;WAS A DISCARDED OR INACTIVE BCB
2127  =098C 08F6                    or si,si                ;FOUND?
2128  =098E 740A        099A        jz GET_BCB25            ;NO
2129  =
2130  =                       if BMPM
2131  =0990 8B5C0C                    MOV bx,12[si]
2132  =0993 803FFF                    CMP B[bx],0FFH          ;WAS IT DISCARDED?
2133  =0996 7412        09AA          JE GET_BCB26            ;YES - TAKE IT
2134  =                       endif
2135  =
2136  =0998 8BFE                      mov di,si               ;LAST_BCBA = EMPTY_BCBA
2137  =                       GET_BCB25:
2138  =
2139  =                       if BMPM
2140  =099A 8B367308                MOV SI,ROOT_BCBA        ;IS THE NUMBER OF BCB'S OWNED
2141  =099E A07908                  MOV AL,P_BCB_CNT        ;BY THIS PROCESS GREATER THAN
2142  =09A1 3A4402                  CMP AL,2[SI]            ;THE MAXIMUM SET BY GENSYS?
2143  =09A4 7204        09AA        JC GET_BCB26            ;NO
2144  =09A6 8B3E7708                  MOV di,P_LAST_BCBA    ;USE THE LAST BCB OWNED BY THE
2145  =                       GET_BCB26:                      ;CURRENT PROCESS
2146  =
2147  =                       endif
2148  =
2149  =09AA 8B5D0C                  mov bx,12[di]
2150  =09AD A07008                  mov al,phy_off
2151  =09B0 884705                  mov 5[bx],al            ;sequence field = phy_off
2152  =                       GET_BCB3:
2153  =09B3 8B367308                MOV SI,ROOT_BCBA        ;PUT CURBCB ON HEAD OF BCB LIST
2154  =09B7 8B04                    MOV AX,[SI]
2155  =09B9 3BC3                    cmp ax,bx               ;is cur bcb at head of bcb list
2156  =09BB 7408        09C5        je get_bcb35            ;yes
2157  =09BD 87470C                    xchg ax,12[bx]
2158  =09C0 89450C                    mov 12[di],ax
2159  =09C3 891C                      MOV [SI],BX
2160  =                       GET_BCB35:
2161  =
2162  =                       if BMPM
2163  =09C5 A16800                  MOV AX,RLR              ;SET THE BCB PROCESS FIELD
2164  =09C8 89470E                  MOV 14[BX],AX
2165  =                       endif
```

```
2166
2167  =
2168  =09CB C3                        RET
2169  =
2170  =                          ;
2171  =                          ;       unallocated block entry structure
2172  =                          ;
2173  =                          ;       +-----+-----+-----+-----+-----+-----+
2174  =                          ;       |    LINK   |   BLOCK   | DRV | HWM |
2175  =                          ;       +-----+-----+-----+-----+-----+-----+
2176  =                          ;
2177  =                          ;       LINK  - link to next entry or 0 if end of list
2178  =                          ;       BLOCK - allocation block number
2179  =                          ;       DRV   - drive number or 0FFh if not valid
2180  =                          ;       HWM   - high water mark of sectors written to block
2181  =                          ;
2182  =  0000                   ua_link         equ     word ptr 0
2183  =  0002                   ua_block        equ     word ptr 2
2184  =  0004                   ua_drv          equ     byte ptr 4
2185  =  0005                   ua_hwm          equ     byte ptr 5
2186  =
2187  =
2188  =                          ua_setup:       ;set up unallocated block entry in list
2189  =                          ;--------
2190  =                          ;       entry:  DX = block number
2191  =                          ;       exit:   DX = block number
2192  =
2193  =09CC BB2508                        mov bx,offset ua_lroot  ;get unallocated list root
2194  =                          ua_in1:
2195  =09CF 8BF8                         mov di,bx               ;save last entry address
2196  =09D1 8B1F                         mov bx,ua_link[bx]
2197  =09D3 833F00                       cmp ua_link[bx],0       ;loop to end of list
2198  =09D6 75F7          09CF           jne ua_in1
2199  =
2200  =09D8 A01609                       mov al,adrive           ;initialize entry values
2201  =09DB 884704                       mov ua_drv[bx],al
2202  =09DE 895702                       mov ua_block[bx],dx
2203  =09E1 33C0                         xor ax,ax
2204  =09E3 884705                       mov ua_hwm[bx],al
2205  =
2206  =09E6 8905                         mov ua_link[di],ax      ;zero link of last entry
2207  =09E8 8BC3                         mov ax,bx
2208  =09EA 87062508                     xchg ua_lroot,ax        ;place entry at beginning of list
2209  =09EE 8907                         mov ua_link[bx],ax
2210  =09F0 C3                           ret
2211  =
2212  =                          ua_discard:     ;invalidates entrys for the current disk
2213  =                          ;----------
2214  =                          ;       entry:  none
2215  =
2216  =09F1 A01609                       mov al,adrive           ;get current disk
2217  =09F4 BB2508                       mov bx,offset ua_lroot
2218  =                          ua_fl1:
```

```
2219
2220  =09F7 8B1F                    mov bx,ua_link[bx]
2221  =09F9 384704                  cmp ua_drv[bx],al
2222  =09FC 7504          0A02      jne ua_fl2
2223  =09FE C64704FF                  mov ua_drv[bx],0ffh
2224  =                       ua_fl2:
2225  =0A02 833F00                  cmp ua_link[bx],0       ;loop to end of list
2226  =0A05 75F0          09F7      jne ua_fl1
2227  =0A07 C3                      ret
2228  =
2229  =                       ua_preread:     ;check if pre read is required
2230  =                       ;----------
2231  =                       ;       entry:  CL = vrecord number & blkmsk
2232  =                       ;       exit:   C flag set if pre read required
2233  =
2234  =0A08 A01609                  mov al,adrive
2235  =0A0B BB2508                  mov bx,offset ua_lroot
2236  =0A0E 8B162308                mov dx,blk_num          ;DX = block number
2237  =                       ua_pr0:
2238  =0A12 8B1F                    mov bx,ua_link[bx]
2239  =0A14 384704                  cmp ua_drv[bx],al       ;if ua_drv = curdsk
2240  =0A17 7518          0A34      jne ua_pr2              ; and ua_block = block# then
2241  =0A19 395702                  cmp ua_block[bx],dx     ; check high water mark
2242  =0A1C 7516          0A34      jne ua_pr2
2243  =0A1E 3A4F05                    cmp cl,ua_hwm[bx]     ;if ua_hwm <= cl then
2244  =0A21 7210          0A33      jb ua_pr1
2245  =0A23 A0C808                      mov al,phymsk
2246  =0A26 8AE0F6D4                    mov ah,all not ah
2247  =0A2A 22CCFEC0                    and cl,ah! inc al   ; compute next physical offset
2248  =0A2E 02C1                        add al,cl           ; save new high water mark
2249  =0A30 884705                      mov ua_hwm[bx],al   ; and return with carry reset
2250  =                       ua_pr1:
2251  =0A33 C3                          ret                 ; else return with carry set
2252  =                       ua_pr2:
2253  =0A34 833F00                  cmp ua_link[bx],0
2254  =0A37 75D9          0A12      jne ua_pr0
2255  =0A39 F9                      stc                     ;return with carry set
2256  =0A3A C3                      ret
2257  =
2258  =
2259  =                       DEBLOCK_IO:
2260  =                       ;----------
2261  =                       ;       entry:  AL = 0 -> SEEK ONLY
2262  =                       ;                  = 1 -> WRITE
2263  =                       ;                  = 2 -> READ
2264  =
2265  =0A3B 50                      PUSH AX
2266  =0A3C E8B5FA          04F4    CALL SEEK
2267  =0A3F 58                      POP AX
2268  =0A40 FEC8                    DEC AL
2269  =0A42 780A          0A4E      JS deblk_io2
2270  =0A44 7505          0A4B        JNZ deblk_io1
2271  =0A46 B101                        mov cl,1
```

```
2272
2273  =0A48 E999FD        07E8            JMP WRBUFF
2274  =                   deblk_io1:
2275  =0A4B E8A3FD        07F1              CALL RDBUFF
2276  =                   deblk_io2:
2277  =0A4E BE7308                        mov si,offset track
2278  =0A51 8B3E7108                      mov di,cur_bcba
2279  =0A55 83C706                        add di,6                    ;MOVE TRACK & SECTOR
2280  =0A58 B90200                        mov cx,2                    ;TO BCB
2281  =0A58 F3A5                          rep movsw
2282  =0A5D C3                            ret
2283  =
2284  =                         READ_DEBLOCK:
2285  =                         ;------------
2286  =0A5E B401                          mov ah,1                    ;AH = 1
2287  =0A60 E81500        0A79            CALL DEBLOCK_DTA
2288  =0A63 E9B3FB        0619            JMP SETFCB
2289  =
2290  =                         DEBLOCK_DIR:
2291  =                         ;-----------
2292  =0A66 8C1E1E09                      MOV CUR_dma_seg,DS          ;BUFFERS IN SYSTEM DATA AREA
2293  =0A6A 8B1EB208                      MOV BX,DIR_BCBA
2294  =0A6E 80FC05                        CMP AH,5                    ;IS FX = DIRECTORY UPDATE?
2295  =0A71 7569          0ADC            JNZ DEBLOCK                 ;NO
2296  =0A73 8B1E7108                      MOV BX,CUR_BCBA             ;YES - CUR_BCBA KNOWN FROM
2297  =0A77 EB63          0ADC            jmps DEBLOCK                ;PREVIOUS LOCATE CALL
2298  =
2299  =                         DEBLOCK_DTA:
2300  =                         ;-----------
2301  =0A79 8B1EB408                      MOV BX,DAT_BCBA
2302  =0A7D C7061E090000                  MOV CUR_dma_seg,0           ;get segment from BCB
2303  =0A83 80FC04                        CMP AH,4                    ;IS FX = FLUSH?
2304  =0A86 7554          0ADC            JNZ DEBLOCK                 ;NO
2305  =
2306  =                         DEBLOCK_FLUSH:
2307  =                         ;-------------
2308  =0A88 8B1F                          MOV BX,[BX]
2309  =0A8A C7067B08FFFF                  MOV TRACK,0FFFFH
2310  =                         DEBLOCK_FLUSH0:
2311  =0A90 A01509                        MOV AL,ADRIVE
2312  =0A93 3A07                          CMP AL,[BX]                 ;DOES BCB(0) = ADRIVE?
2313  =0A95 751F          0AB6            JNZ DEBLOCK_FLUSH1          ;NO
2314  =
2315  =0A97 F64704FF                      TEST BYTE PTR 4[BX],0FFH    ;IS BUFFER WRITE PENDING?
2316  =0A9B 7419          0AB6            JZ DEBLOCK_FLUSH1           ;NO
2317  =
2318  =                               if BMPM
2319  =0A9D 8B470E                          MOV ax,14[BX]           ;IS BCB OWNED BY CALLING PROCESS?
2320  =0AA0 3B056800                        CMP ax,RLR
2321  =0AA4 7510          0AB6              JNZ DEBLOCK_FLUSH1        ;NO
2322  =                               endif
2323  =
2324  =0AA6 8B4706                          MOV AX,6[BX]              ;IS BCB(6) < TRACK?
```

```
2325
2326  =0AA9 36067B08                    CMP AX,TRACK
2327  =0AAD 7307          0AB6          JNC DEBLOCK_FLUSH1          ;NO
2328  =
2329  =0AAF A37B08                      MOV TRACK,AX                ;YES - TRACK = BCB(6)
2330  =0AB2 891E7D08                    MOV SECTOR,BX               ;SECTOR = .BCB
2331  =
2332  =                           DEBLOCK_FLUSH1:
2333  =0AB6 8B5F0C                      mov bx,12[bx]
2334  =0AB9 08DB                        or bx,bx
2335  =0ABB 7503          0AC0          JNZ DEBLOCK_FLUSH0
2336  =
2337  =0ABD 833E7B08FF                  CMP TRACK,0FFFFH            ;WAS A DIRTY BCB FOUND?
2338  =0AC2 7501          0AC5          JNZ $+3                     ;YES
2339  =0AC4 C3                            RET
2340  =0AC5 8B1E7D08                    MOV BX,SECTOR               ;FLUSH THE BCB'S BUFFER
2341  =0AC9 32C0                        xor al,al                   ;set z flag
2342  =0ACB B404                        MOV AH,4
2343  =0ACD C7061E090000                mov cur_dma_seg,0           ;get segment from BCB
2344  =0AD3 E80600        0ADC          CALL DEBLOCK
2345  =
2346  =0AD6 891EB408                    MOV BX,DAT_BCBA
2347  =0ADA EBAC          0A88          jmps DEBLOCK_FLUSH
2348  =
2349  =                           DEBLOCK:        ;BDOS BLOCKING/DEBLOCKING ROUTINE
2350  =                           ;-------
2351  =                           ;       entry:  Z flag reset -> get new BCB address
2352  =                           ;               AH = 1 -> READ COMMAND
2353  =                           ;                  = 2 -> WRITE COMMAND
2354  =                           ;                  = 3 -> LOCATE COMMAND
2355  =                           ;                  = 4 -> FLUSH COMMAND
2356  =                           ;                  = 5 -> DIRECTORY UPDATE
2357  =
2358  =0ADC 88266F08                    MOV DEBLOCK_FX,AH           ;SAVE DEBLOCK FX
2359  =0AE0 9F                          lahf                        ;save Z flag
2360  =0AE1 8A0EC808                    mov cl,phymsk               ;CL = PHYMSK
2361  =0AE5 A01709                      MOV AL,BYTE PTR ARECORD
2362  =0AE8 22C1                        AND AL,cl
2363  =0AEA A27008                      MOV PHY_OFF,AL              ;PHY_OFF = LOW(ARECORD) & PHYMSK
2364  =0AED F6D1                        not cl                      ;CL = ~PHYMSK
2365  =0AEF 200E1709                    and BYTE PTR ARECORD,cl     ;LOW(ARECORD) = LOW(ARECORD) & ~PHYMSK
2366  =0AF3 9E                          sahf
2367  =0AF4 7403          0AF9          JZ $+5                      ;IF DEBLOCK CALLED WITH Z FLAG RESET
2368  =0AF6 E885FD        08AE            CALL GET_BCBA             ;THEN GET BCB ADDRESS
2369  =0AF9 891E7108                    MOV CUR_BCBA,BX
2370  =0AFD 88470A                      MOV ax,10[BX]               ;dma address field - offset/segment
2371  =0B00 833E1E0900                  cmp cur_dma_seg,0           ;if cur_dma_seg <> 0, deblocking is
2372  =0B05 7505          0B0C          jne deblock0                ; for dir buf and addr is offset
2373  =0B07 A31E09                        mov cur_dma_seg,ax        ;else deblocking data buffer and
2374  =0B0A 33C0                          xor ax,ax                 ; addr is segment, offset = 0
2375  =                           deblock0:
2376  =0B0C A32009                      MOV CUR_DMA,ax              ;save current buffer address
2377  =0B0F A06F08                      mov al,deblock_fx
```

```
2378
2379  =0B1C 3C037503      0B21          cmp al,31 jne deblock01          ;if (deblock func = locate)
2380  =0B16 32E4                          xor ah,ah                      ; and (must read dir = true)
2381  =0B18 86261008                      xchg rd_dir_flag,ah            ; force sector read from disk
2382  =0B1C F6C4F0                        test ah,0f0h                   ; to verify media has not changed
2383  =0B1F 7569          0B8A            jnz deblock25
2384  =                             deblock01:
2385  =0B21 E8B900        0BDD          CALL DEBLOCK9                    ;BX=CUR_BCBA, DX=.ADRIVE, CL=4
2386  =0B24 803FFF7445    0B6E          cmp b[bx],0ffh| je deblock2
2387  =0B29 3C04                          CMP al,4                       ;IS DEBLOCK_FX >=4?
2388  =0B2B 7305          0B32            JNC DEBLOCK1                   ;YES
2389  =0B2D E8B3F9        04E8              CALL COMPARE                 ;DOES BCB(0-3) = ADRIVE || ARECORD?
2390  =0B30 746D          0B9F              JZ DEBLOCK45                 ;YES
2391  =                             DEBLOCK1:
2392  =0B32 3C05                          CMP al,5                       ;IS DEBLOCK_FX = DIR UPDATE?
2393  =0B34 7406          0B3C            JZ DEBLOCK15                   ;YES
2394  =0B36 F64704FF                        TEST BYTE PTR 4[BX],0FFH     ;IS BCB(4) WRITE PENDING FLAG SET?
2395  =0B3A 7432          0B6E              JZ DEBLOCK2                  ;NO
2396  =                             DEBLOCK15:
2397  =0B3C C6470400                      MOV BYTE PTR 4[BX],0           ;WRITE BCB BUFFER
2398  =0B40 FF361609                      PUSH WORD PTR ADRIVE           ;SAVE ADRIVE & ARECORD(0)
2399  =0B44 FF351809                      PUSH WORD PTR ARECORD+1        ;SAVE ARECORD(1,2)
2400  =0B48 8B4702                        mov ax,2[bx]
2401  =0B4B A31809                        mov word ptr arecord+1,ax
2402  =0B4E 8B07                          mov ax,[bx]
2403  =0B50 A31609                        mov word ptr adrive,ax         ;ADRIVE || ARECORD = CURBCB(0-3)
2404  =0B53 3806A008                      cmp curdsk,al
2405  =0B57 7403          0B5C            je $+5
2406  =0B59 E8420E        199E              call disk_select1
2407  =0B5C B001                          mov al,1
2408  =0B5E 7503          0B63            jnz $+5
2409  =0B60 E8D8FE        0A3B              CALL DEBLOCK_IO              ;write buffer if drive logged in
2410  =0B63 8F061809                      POP WORD PTR ARECORD+1         ;RESTORE ARECORD(1,2)
2411  =0B67 8F051609                      POP WORD PTR ADRIVE            ;RESTORE ADRIVE & ARECORD(0)
2412  =0B6B E8480E        1986            CALL CURSELECT
2413  =                             DEBLOCK2:
2414  =0B6E A06F08                      MOV al,DEBLOCK_FX                ;DOES DEBLOCK_FX = DIR UPDATE OR FLUSH
2415  =0B71 3C04                        CMP al,4
2416  =0B73 7201          0B76        JC $+3
2417  =0B75 C3                          RET                              ;YES
2418  =0B76 3C02                        CMP al,2                         ;DOES DEBLOCK_FX = WRITE?
2419  =0B78 7510          0B9A        JNZ DEBLOCK25                      ;NO
2420  =0B7A 8A0E2208                    MOV cl,BLK_OFF
2421  =0B7E E887FE        0A08        call ua_preread                    ;is preread required?
2422  =0B81 7207          0B8A        jc DEBLOCK25                       ; yes
2423  =0B83 32C0                        XOR AL,AL
2424  =0B85 E8B3FE        0A3B          CALL DEBLOCK_IO                  ;SEEK ONLY
2425  =0B88 EB0C          0B96          JMPS DEBLOCK4
2426  =                             DEBLOCK25:
2427  =0B8A 881E7108                    mov bx,cur_bcba
2428  =0B8E C607FF                      mov b[bx],0ffh                   ;discard in case of error
2429  =0B91 B002                        MOV AL,2
2430  =0B93 E8A5FE        0A3B        CALL DEBLOCK_IO                    ;READ PHYSICAL RECORD
```

```
2431
2432  =                         DEBLOCK4:
2433  =0B96 E84400        0BDD          CALL DEBLOCK9                   ;BX=CUR_BCBA, DX=.ADRIVE, CL=4
2434  =0B99 E845F9        04E2          CALL MOVE
2435  =0B9C C60500                      MOV BYTE PTR [DI],0             ;CUR_BCBA->BCB(4) = 0
2436  =                         DEBLOCK45:
2437  =0B9F 32C0                        xor al,al
2438  =0BA1 8A267008                    MOV ah,PHY_OFF
2439  =0BA5 D1E8                        shr ax,1                        ;AX = phy_off * 080h
2440  =0BA7 8B362009                    MOV SI,CUR_DMA
2441  =0BAB 03F0                        ADD SI,AX                       ;SI = CUR_DMA + PHY_OFF*80H
2442  =0BAD A06F08                      MOV al,DEBLOCK_FX
2443  =0BB0 3C03                        CMP al,3                        ;IS DEBLOCK_FX = LOCATE?
2444  =0BB2 7505          0BB9          JNZ DEBLOCK6                    ;NO
2445  =0BB4 8936AA08                      MOV BUFFA,SI                  ;YES
2446  =0BB8 C3                            RET
2447  =                         DEBLOCK6:
2448  =0BB9 B94000                      MOV CX,40H                      ;transfer 40h words
2449  =0BBC 8B3E8508                    MOV DI,dma_ofst
2450  =0BC0 3C01                        CMP al,1                        ;IS DEBLOCK_FX = READ?
2451  =0BC2 A18708                      mov ax,dma_seg
2452  =0BC5 8B161F09                    mov dx,cur_dma_seg
2453  =0BC9 1E06                        push ds! push es
2454  =0BCB 7407          0BD4          je DEBLOCK7                     ; yes
2455  =0BCD C64704FF                      MOV BYTE PTR 4[BX],0FFH       ; no - write
2456  =0BD1 87FE                          xchg di,si                    ;exchange dma offsets
2457  =0BD3 92                            xchg ax,dx                    ;exchange dma segments
2458  =                         DEBLOCK7:
2459  =0BD4 8EDA                        mov ds,dx                       ;setup source segment
2460  =0BD6 8EC0                        mov es,ax                       ;setup destination segment
2461  =0BD8 F3A5                        rep movsw                       ;transfer data
2462  =0BDA 071F                        pop esi pop ds
2463  =0BDC C3                          RET
2464  =
2465  =                         DEBLOCK9:                               ;SETUP FOR MOVE OR COMPARE
2466  =0BDD 8B1E7108                    MOV BX,CUR_BCBA
2467  =0BE1 BA1609                      MOV DX,OFFSET ADRIVE
2468  =0BE4 B104                        MOV CL,4
2469  =                         RETD1:
2470  =0BE6 C3                          RET
2471  =
2472  =                         read_dir:
2473  =                         ;--------
2474  =0BE7 E83D00        0C27          call rdir
2475  =0BEA F6062108FF                  test relog,0ffh
2476  =0BEF 74F5          0BE6          jz retd1
2477  =0BF1 E8C2FB        07B6          call chk_exit_fxs
2478  =0BF4 E8D8FB        07D2          CALL TST_RELOG                  ;RELOG IN DRIVE
2479  =                                 ;JMP RDDIR                      ;READ DIR REC ZERO & RET
2480  =
2481  =                         rddir:          ;read the current directory record
2482  =                         ;-----
2483  =0BF7 A18F08                      MOV ax,DCNT                     ;seek the record containing
```

```
2484
2485  =0BFA B102D3E8              MOV CL,DSKSHF! SHR ax,cL        ; the current dir entry
2486  =0BFE A31C09                mov drec,ax
2487  =0C01 A31709                MOV ARECORD,ax                  ;ARECORD = SHR(DCNT,DSKSHF)
2488  =0C04 C606190900            MOV BYTE PTR ARECORD+2,0
2489  =0C09 B403                  MOV AH,3                        ;LOCATE COMMAND
2490  =0C0B EB0A          0C17    JMPS WRDIRO
2491  =
2492  =                     wrdir:        ;write the current directory entry, set checksum
2493  =                     ;-----
2494  =0C0D E833F8        0743    call checkwrite                 ;verify disk is read/write
2495  =0C10 B1FF                  mov cl,true
2496  =0C12 E84D00        0C60    call checksum                   ;initialize entry
2497  =
2498  =0C15 B405                  MOV AH,5                        ;DIRECTORY UPDATE CODE
2499  =                     WRDIRO:
2500  =0C17 E84CFE        0A66    CALL DEBLOCK_DTR
2501  =                           ;JMPS SETDATA
2502  =
2503  =                     setdata:      ;set data dma address
2504  =                     ;-------
2505  =0C1A A18708                MOV ax,dma_seg
2506  =0C1D A31E09                MOV CUR_dma_seg,ax
2507  =0C20 A18508                MOV ax,dma_ofst
2508  =0C23 A32009                MOV CUR_DMA,ax
2509  =0C26 C3                    RET
2510  =
2511  =                     rdir:         ;read next directory entry
2512  =                     ;----
2513  =                     ;     entry:  CL = true if initializing
2514  =
2515  =0C27 8B168F08              mov dx,dirmax                   ;in preparation for subtract
2516  =0C2B 8B1E8F08              mov bx,dcnt
2517  =0C2F 43                    inc bx
2518  =0C30 891E8F08              mov dcnt,bx                     ;dcnt=dcnt+1
2519  =                                                           ;continue while dirmax >= dcnt
2520  =                                                           ;(dirmax-dcnt no cy)
2521  =0C34 2BD3                  sub dx,bx
2522  =0C36 7303          0C3B    jnb $+5                         ; no
2523  =0C38 E941F8        077C      jmp setenddir                 ; yes, set dcnt to end
2524  =                                                           ; of directory
2525  =                     ;     not at end of directory, seek next element
2526  =                                                           ;cl=initialization flag
2527  =0C3B A08F08                mov al,ldcnt
2528  =0C3E 2403                  and al,dskmsk                   ;low(dcnt) and dskmsk
2529  =0C40 51                    push cx
2530  =0C41 B105                  mov cl,fcbshf                   ;to multiply by fcb size
2531  =0C43 D2E0                  shl al,cl
2532  =0C45 59                    pop cx
2533  =                                                           ;a = (low(dcnt) and dskmsk)
2534  =                                                           ;shl fcbshf
2535  =0C46 A22209                mov dptr,al                     ;ready for next dir operation
2536  =0C49 F6051008FF            test rd_dir_flag,true           ;if must read or locate dir
```

```
2537
2538  =0C4E 7504          0C54          jnz rddir2                      ; read next directory record
2539  =0C50 0AC0                          or al,al                        ;else
2540  =0C52 7575          0CC9          jnz ret71                       ; return if not a new record
2541  =                           RDDIR2:
2542  =0C54 51                          push cx                         ;save initialization flag cl
2543  =0C55 E89FFF        0BF7          call rddir                      ;read the directory record
2544  =0C58 59                          pop cx                          ;recall initialization flag
2545  =0C59 F6062108FF                  test relog,0ffh
2546  =0C5E 7569          0CC9          jnz ret71
2547  =                                 ;jmps checksum                  ;checksum the directory element
2548  =
2549  =                           checksum:
2550  =                           ;--------
2551  =                           ;       compute current checksum record and update the
2552  =                           ;       directory element if cl=true, or check for = if not
2553  =                           ;       drec < chksiz?
2554  =                           ;       entry:  CL = true update checksum
2555  =                           ;                  = false if check for equal
2556  =
2557  =0C60 8B161C09                    mov dx,drec
2558  =0C64 8B1EC308                    mov bx,chksiz
2559  =0C68 80E77F                      and bh,7fh                      ;remove permanent drive bit
2560  =0C6B 2BD3                        sub dx,bx
2561  =0C6D 7342          0CB1          jae ret7                        ;skip checksum if past
2562  =                                                                 ;checksum vector size
2563  =                                                                 ;drec < chksiz, so continue
2564  =0C6F 51                          push cx                         ;save init flag
2565  =0C70 E8CEF9        0641          call cmpecs                     ;check sum value to al
2566  =0C73 8B1E1C09                    mov bx,drec                     ;value of arecord
2567  =0C77 031EAE08                    add bx,checka                   ;bx=.check(arecord)
2568  =0C7B 59                          pop cx                          ;recall true or false to cl
2569  =0C7C FEC1                        inc cl                          ;true produces zero flag
2570  =0C7E 742F          0CAF          jz initcs
2571  =
2572  =                           if BMPM
2573  =0C80 FEC1                        inc cl                          ;0feh produces zero flag
2574  =0C82 7424          0CA8          jz test_dir_cs
2575  =                           endif
2576  =                                                                 ;not initializing, compare
2577  =0C84 3A07                          cmp al,[bx]                   ;compute$cs=check(arecord)?
2578  =0C86 7429          0CB1            jz ret7                               ;no message if ok
2579  =0C88 E883FA        0716            CALL NOWRITE
2580  =0C8B 7524          0CB1            JNZ RET7                      ;RETURN IF ALREADY READ/ONLY
2581  =
2582  =                           media_change:
2583  =0C8D 8B1EB408                      mov bx,dat_bcba
2584  =0C91 E8CBF8        085F            call discard
2585  =
2586  =                           if BMPM
2587  =0C94 E8C811        1E5F            call flush_file0                ;flush files
2588  =                           endif
2589  =
```

```
2590
2591  =0C97 B0FF                        mov al,0ffh
2592  =0C99 A22108                      MOV RELOG,al
2593  =0C9C A24F08                      mov nashl,al
2594  =0C9F BB0008                      mov bx,offset rlog
2595  =0CA2 E865FA        070A          call setcdisk
2596  =                         ;       mov cl,curdsk           ;ax is setup in setcdisk
2597  =                         ;       mov ax,1
2598  =                         ;       shl ax,cl
2599  =0CA5 E9E11C        2989          jmp reset_37x
2600  =
2601  =                         if BMPM
2602  =
2603  =                         test_dir_cs:
2604  =0CA8 3A07                        cmp al,[bx]             ;compute_cs=check(arecord)
2605  =0CAA 7405          0CB1          jz ret7
2606  =0CAC E9A611        1E55          jmp flush_files
2607  =                         endif
2608  =
2609  =                         initcs:
2610  =                                                         ;initializing the checksum
2611  =0CAF 8807                      mov [bx],al
2612  =0CB1 C3                  ret7: ret
2613  =
2614  =
2615  =                         ;********** end bdos file system part 1 **********
2616
```

```
2617
2618  =                       eject 1 include file2.bdo       ; file system part 2
2619  =
2620  =                   ;************ bdos file system part 2 ************
2621  =
2622  =                   getallocbit:
2623  =                   ;-----------
2624  =                   ;       given allocation vector position on cx, return byte
2625  =                   ;       containing cx shifted so that the least significant
2626  =                   ;       bit is in the low order accumulator position. bx is
2627  =                   ;       the address of the byte for possible replacement in
2628  =                   ;       memory upon return, and dh contains the number of shifts
2629  =                   ;       required to place the returned value back into position
2630  =                   ;       entry:  CX = allocation vector bit index
2631  =                   ;       exit:   AL = alloc bit index in low order bit
2632  =                   ;               BX = allocation vector byte offset
2633  =                   ;               CL = # of shifts required to restore bit in byte
2634  =
2635  =0CB2 8BD9                  mov bx,cx
2636  =0CB4 80E107                and cl,111b
2637  =0CB7 FEC1                  inc cl
2638  =0CB9 8AE9                  mov ch,cl                       ;ch and cl both contain the
2639  =                                                           ;number of bit positions to
2640  =                                                           ;shift
2641  =0CBB B103                  mov cl,3                        ;shift bit address right 3
2642  =0CBD D3EB                  shr bx,cl                       ; for byte address
2643  =                                                           ;dx shr 3 to dx
2644  =0CBF 031EB008              add bx,alloca                   ;base addr. of alloc. vector
2645  =                                                           ;byte to a, hl =
2646  =0CC3 8A07                  mov al,[bx]                     ;.alloc(cx shr 3)
2647  =                                                           ;now move the bit to the
2648  =0CC5 8AC1                  mov cl,ch                       ;low order position of al
2649  =0CC7 D2C0                  rol al,cl
2650  =0CC9 C3            ret71:  ret
2651  =
2652  =                   setallocbit:
2653  =                   ;-----------
2654  =                   ;       entry:  CX = allocation vector bit index to set or reset
2655  =                   ;               DL = low order bit is value to set into vector
2656  =
2657  =0CCA 52                    push dx
2658  =0CCB E8E4FF        0CB2    call getallocbit                ;shifted val al,count in dl
2659  =0CCE 24FE                  and al,11111110b                ;mask low bit to zero
2660  =                                                           ;(may be set)
2661  =0CD0 5A                    pop dx
2662  =0CD1 0AC2                  or al,dl                        ;low bit of cl masked into al
2663  =
2664  =                   rotr:           ;rotate and replace byte into allocation vector
2665  =                   ;----
2666  =                   ;       entry:  AL = byte from allocation vector
2667  =                   ;               BX = allocation vector byte offset
2668  =                   ;               CL = # of rotates required to restore alloc vector byte
2669  =
```

```
2670
2671  =0CD3 D2C8                    ror al,cl
2672  =0CD5 8807                    mov [bx],al
2673  =0CD7 C3                      ret
2674  =
2675  =                     get_nalbs:      ;get # of allocation vector bytes
2676  =                     ;---------
2677  =                     ;       exit:   BX = # of bytes in allocation vector
2678  =
2679  =0CD8 8B1E8D08                mov bx,maxall
2680  =0CDC B103                    mov cl,3                        ;perform maxall/8
2681  =0CDE D3EB                    shr bx,cl
2682  =0CE0 43                      inc bx
2683  =0CE1 C3                      ret
2684  =
2685  =                     copy_alv:       ;copy allocation vector
2686  =                     ;--------
2687  =                     ;       entry:  Z flag = 1 - copy 1st ALV to 2nd
2688  =                     ;                        = 0 - copy 2nd ALV to 1st
2689  =
2690  =0CE2 9C                      pushf
2691  =0CE3 E8F2FF      0CD8        call get_nalbs
2692  =0CE6 8B368D08                mov si,alloca
2693  =0CEA 8BFE                    mov di,si
2694  =0CEC 03FB                    add di,bx
2695  =0CEE 8BCB                    mov cx,bx
2696  =0CF0 9D7402      0CF5        popf! jz copya0
2697  =0CF3 87F7                      xchg si,di
2698  =                     copya0:
2699  =0CF5 F3A4                    rep movsb
2700  =0CF7 E8F7FC      09F1        call ua_discard                 ;discard unallocated check blocks
2701  =0CFA C3              ret8a:  ret
2702  =
2703  =                     scndma:         ;set/reset 1st ALV
2704  =                     ;------
2705  =                     ;       scan the disk map addressed by dptr for non-zero
2706  =                     ;       entries, the allocation vector entry corresponding
2707  =                     ;       to a non-zero entry is set to the value of cl (0,1)
2708  =                     ;       entry:  CL = 1/0 - set/reset blocks in alloc vector
2709  =
2710  =0CFB E827FA      0725        call get_dptra                  ;bx = buffa + dptr
2711  =0CFE 83C310                  add bx,dskmap                   ;bx = diskmap address
2712  =0D01 51                      push cx                         ;save the 0/1 bit to set
2713  =0D02 B111                    mov cl,fcblen-dskmap+1          ;size of single byte diskmap+1
2714  =                     scndm0:
2715  =0D04 5A                      pop dx                          ;loop for each diskmap entry
2716  =0D05 FEC9                    dec cl                          ;recall bit parity
2717  =0D07 750E        0D17        jnz scndma0a
2718  =0D09 0AD275ED    0CFA          or dl,dl! jnz ret8a           ;all done scanning?
2719  =0D0D 8B1E8D08                  mov bx,alloca
2720  =0D11 A1C1080907                mov ax,dirblk! or [bx],ax     ;restore directory blocks
2721  =0D16 C3                        ret
2722  =                     scndma0a:
```

```
2723
2724  =0D17 52                          push dx                         ;no, get next entry for scan
2725  =0D18 803E120900                  cmp single,0                    ;replace bit parity
2726  =0D1D 7408            0D27        jz scndm1
2727  =
2728  =0D1F 51                            push cx                       ;single byte scan operation
2729  =0D20 53                            push bx                       ;save counter
2730  =0D21 8A0F                          mov cl,[bx]                   ;save map address
2731  =0D23 B500                          mov ch,0                      ;cx=block#
2732  =0D25 EB07            0D2E          jmps scndm2
2733  =                           scndm1:                               ;double byte scan operation
2734  =0D27 FEC9                        dec cl                          ;count for double byte
2735  =0D29 51                          push cx                         ;save counter
2736  =0D2A 8B0F                        mov cx,[bx]                     ;cx=block
2737  =0D2C 43                          inc bx
2738  =0D2D 53                          push bx                         ;save map address
2739  =                           scndm2:                               ;CX = block#, DL = 0/1
2740  =0D2E 08C9                        or cx,cx                        ;skip if = 0000
2741  =0D30 7408            0D3D        jz scan3
2742  =0D32 8B1E8D08                      mov bx,maxall                 ;check invalid index
2743  =0D36 3BD9                          cmp bx,cx
2744  =0D38 7203            0D3D          jnae $+5
2745  =0D3A E88DFF          0CCA            call setallocbit
2746  =                           scan3:                                ;bit set to 0/1
2747  =0D3D 5B                          pop bx
2748  =0D3E 43                          inc bx                          ;to next bit position
2749  =0D3F 59                          pop cx                          ;recall counter
2750  =0D40 EBC2            0D04        jmps scndm0                     ;for another item
2751  =
2752  =                           scndmab:        ;set/reset 1st and 2nd ALV
2753  =                           ;-------
2754  =0D42 51                          push cx
2755  =0D43 E885FF          0CFB        call scndma
2756  =0D46 59                          pop cx
2757  =                                 ;jmp scndmb
2758  =
2759  =                           scndmb:         ;set/reset 2nd ALV
2760  =                           ;------
2761  =0D47 51                          push cx
2762  =0D48 E88DFF          0CD8        call get_nalbs
2763  =0D4B 59                          pop cx
2764  =0D4C A1B00850                    mov ax,allocal push ax
2765  =0D50 03C3                        add ax,bx
2766  =0D52 A3B008                      mov alloca,ax
2767  =0D55 E8A3FF          0CFB        call scndma
2768  =0D58 8F06B008                    pop alloca
2769  =0D5C C3                          ret
2770  =
2771  =                           initialize:
2772  =                           ;----------
2773  =                           ;       initialize the current disk
2774  =                           ;       lret = false ;set to true if $ file exists
2775  =                           ;       compute the length of the allocation vector - 2
```

```
2776
2777  =
2778  =                         if BMPM
2779  =0D5D 88160208                    mov dx,tlog
2780  =0D61 E8B5F9        071A          call test_vector          ;is tlog(curdsk) on
2781  =0D64 7427          0D8D          jz initialize1            ;no
2782  =0D66 BB0208                        mov bx,offset tlog      ;tlog(curdsk) = off
2783  =0D69 E81B10        1087          call remove_drive
2784  =0D6C C606020900                  mov flushed,false
2785  =0D71 8501                        mov ch,1                  ;is a file currently open
2786  =0D73 E8860F        1CFC          call search_dlist          ;on this drive?
2787  =0D76 7515          0D8D          jnz initialize1           ;no - relog in the drive
2788  =0D78 E801FA        077C            call set_end_dir
2789  =                         test_dir1:
2790  =0D7B B1FE                        mov cl,0feh
2791  =0D7D E867FE        0BE7          call read_dir
2792  =0D80 803E020900                  cmp flushed,false         ;has a change in media occurred?
2793  =0D85 7505          0D8D          jne initialize1           ; yes - login drive
2794  =0D87 E8EBF9        0775            call end_of_dir
2795  =0D8A 75EF          0D7B            jnz test_dir1
2796  =0D8C C3                          ret                       ; no - do not relog in drive
2797  =                         initialize1:                      ;log in drive
2798  =
2799  =                         endif
2800  =
2801  =0D8D 813EC3080080                cmp chksiz,8000h          ;if chksiz = 8000h
2802  =0D93 7509          0DA2          jne init1
2803  =0D95 8B1EA808                      mov bx,drv1bla          ; and lsn <> 0 skip relog
2804  =0D99 8001864702                    mov al,11 xchg 2[bx],al
2805  =0D9E 0AC07541      0DE3            or al,all jnz init23    ; only copy allocation vector
2806  =                         init1:
2807  =0DA2 8B1EB408                    MOV BX,DAT_BCBA           ;DISCARD ACTIVE DATA BCB'S
2808  =0DA6 E8B6FA        085F          CALL DISCARD
2809  =0DA9 E8AFFA        0858          CALL discard_dir          ;DISCARD ACTIVE DIRECTORY BCB'S
2810  =
2811  =0DAC E829FF        0CD8          call get_nalbs            ;bx = # of allocation vector bytes
2812  =0DAF 8BCB                        mov cx,bx
2813  =0DB1 8B3EB008                    mov di,alloca             ;base of allocation vector
2814  =0DB5 A1C108                      mov ax,dirblk
2815  =0DB8 AB                          stosw                     ;sets reserved directory blks
2816  =0DB9 4949                        dec cx! dec cx
2817  =0DBB 33C0                        xor ax,ax                 ;fill the allocation vector
2818  =0DBD F3AA                        rep stosb                 ;with zeros
2819  =                                                           ;allocation vector initialized
2820  =                                                           ;set the reserved space for
2821  =0DBF 8B1EA808                    mov bx,drv1bla            ;the directory
2822  =0DC3 8807                        mov [bx],al               ;zero directory label data byte
2823  =0DC5 884701                      mov byte ptr 1[bx],al     ;ZERO DRIVE'S MEDIA FLAG
2824  =0DC8 FE4702                      INC BYTE PTR 2[BX]        ;INCREMENT DRIVE'S LOGIN SEQ #
2825  =
2826  =0DCB A31A09                      MOV ARECORD1,ax           ;hash table offset
2827  =
2828  =0DCE 8B1EA608                    mov bx,cdrmaxa            ;cdrmax = 3 (scans at least
```

```
2829
2830  =0DD2 C7070300                    mov w[bx],3                     ;one directory record)
2831  =                                 ;cdrmax = 0000
2832  =0DD6 E8A3F9      077C            call setenddir                  ;dcnt = enddir
2833  =                                 ;read directory entries and
2834  =                                 ;check for allocated storage
2835  =                       init2:
2836  =0DD9 B1FF                        mov cl,true
2837  =0DDB E809FE      0BE7            call read_dir
2838  =0DDE E894F9      0775            call endofdir
2839  =0DE1 7503        0DE6            jnz $+5
2840  =                       init23:
2841  =0DE3 E9FCFE      0CE2              jmp copy_alv                  ;copy allocation vector and return
2842  =                                 ;not end of directory,
2843  =                                 ;valid entry ?
2844  =
2845  =0DE6 E8EA01      0FD3            call fix_hash                   ;COMPUTE HASH FOR DIRECTORY ITEM
2846  =0DE9 E839F9      0725            call get_dptra                  ;bx = buffa + dptr
2847  =0DEC 8A07                        mov al,[bx]
2848  =0DEE 3C21                        CMP al,21H                      ;IS ENTRY SFCB?
2849  =0DF0 74E7        0DD9            JZ INIT2                        ;YES
2850  =
2851  =0DF2 3CE5                        cmp al,empty
2852  =0DF4 74E3        0DD9            jz init2                        ;go get another item
2853  =                                 ;not empty,user code the same?
2854  =
2855  =0DF6 3C20                        cmp al,20h                      ;is entry directory label ?
2856  =0DF8 740E        0E08            je drv_lbl                      ;yes
2857  =0DFA A810                          test al,10h                   ;is entry special type of fcb ?
2858  =0DFC 7505        0E03              jnz initial3                  ;yes
2859  =
2860  =                                 ;now scan the disk map for
2861  =                                 ;allocated blocks
2862  =0DFE B101                            mov cl,1                    ;set to allocated
2863  =0E00 E8F8FE      0CFB                call scndma
2864  =                       initial3:
2865  =0E03 E892F9      0798              call setcdr                   ;set cdrmax to dcnt
2866  =0E06 EBD1        0DD9              jmps init2                    ;for another entry
2867  =                       drv_lbl:
2868  =0E08 8A470C                      mov al,extnum[bx]
2869  =0E0B 8B1EA808                    mov bx,drvlbla
2870  =0E0F 8807                        mov [bx],al
2871  =0E11 EBF0        0E03            jmps initial3
2872  =
2873  =
2874  =                       cpydirloc:
2875  =                       ;---------
2876  =                       ;       copy directory location to lret following
2877  =                       ;       delete, rename, ... ops
2878  =
2879  =0E13 A00F09                      mov al,dirloc
2880  =0E16 E995F6      04AE            jmp set_lret
2881  =                                 ;ret
```

```
2882
2883  =
2884  =                         chk_wild:       ;check fcb for ? marks
2885  =                         ;--------
2886  =                         ;       entry:  BX = .fcb(0)
2887  =                         ;       exit:   BX = preserved
2888  =                         ;               Z flag = 1 if ? mark found
2889  =
2890  =0E19 B90B00                      mov cx,11
2891  =0E1C 88F346                      mov si,bx1 inc si
2892  =                         chk_wild1:
2893  =0E1F AC                          lodsb
2894  =0E20 247F                        and al,07fh
2895  =0E22 3C3F                        cmp al,"?"
2896  =0E24 7404          0E2A          jz ret10                        ;rtn with z flag set if ? fnd
2897  =0E26 E2F7          0E1F          loop chk_wild1
2898  =0E28 04C0                        or al,al                        ;rtn with z flag reset - no ?s
2899  =0E2A C3                  ret10:  ret
2900  =
2901  =                         check_wild:     ;check for ? in file name or type
2902  =                         ;-----------
2903  =0E2B BB3C0B                      mov bx,offset info_fcb
2904  =                         check_wild0:                            ;entry point used by rename
2905  =0E2E E8E8FF        0E19          call chk_wild
2906  =0E31 75F7          0E2A          jnz ret10
2907  =0E33 B409                        mov ah,9                        ;extended error 9
2908  =0E35 E97CF6        04B4          jmp set_aret
2909  =
2910  =
2911  =                         SET_HASH:       ; set hash length and fill in hash
2912  =                         ;--------
2913  =                         ;       hash format:  xxsuuuuu xxxxxxxx xxxxxxxx ssssssss
2914  =                         ;         x = hash code of fcb name field
2915  =                         ;         u = low 5 bits of user number
2916  =                         ;         s = shr(mod || ext,extshf)
2917  =                         ;       entry:  CL = searchl
2918  =                         ;               BX = .info_fcb
2919  =                         ;       exit:   hashl = 0,2 or 3
2920  =                         ;               hash(*) = hash code for length of hashl+1
2921  =
2922  =0E38 833EB60800                  CMP HASH_SEG,0                  ;DOES HASH TBL EXIST ON DRIVE?
2923  =0E3D 74E8          0E2A          JE ret10                        ;NO
2924  =0E3F 0AC9                        OR cl,cl                        ;DOES SEARCHL = 0?
2925  =0E41 74E7          0E2A          JZ ret10                        ;YES
2926  =0E43 80F90C                      CMP cl,12                       ;IS SEARCHL < 12?
2927  =0E46 721E          0E66          jb SET_HASH2                    ;YES
2928  =0E48 B002                        MOV AL,2
2929  =0E4A 7402          0E4E          je SET_HASH1                    ;SEARCHL = 12
2930  =0E4C B003                          MOV AL,3
2931  =                         SET_HASH1:
2932  =0E4E A24F08                        MOV HASHL,AL                  ;HASHL = 2 OR 3
2933  =0E51 A07A08                        MOV AL,fx_intrn
2934  =                         if BCPM
```

```
2935
2936  =                             cmp al,fxi16            ;is func# = close?
2937  =                             je get_hash             ; yes
2938  =                       endif
2939  =0E54 3C16                    cmp al,fxi35            ;is func# = compute file size?
2940  =0E56 7404          0E5C      je set_hash15           ; yes
2941  =0E58 3C07                      CMP AL,fxi20          ;is func# >= read seqn?
2942  =0E5A 730F          0E6B        jae GET_HASH        ; yes
2943  =                       SET_HASH15:
2944  =0E5C C6064F0802              MOV HASHL,2             ;HASHL = 2 FOR SEARCH, SEARCHN,
2945  =                                                     ;AND DELETE IF NOT WILD
2946  =                                                     ;HASHL = 2 FOR OPENS
2947  =0E61 E885FF        0E19      CALL CHK_WILD           ;IS FCB WILD?
2948  =0E64 7505          0E6B      JNZ GET_HASH            ;NO
2949  =                       SET_HASH2:                    ;HASHL = 0 IF SEARCHL = 1 OR
2950  =0E66 C6064F0800            mov hashl,0             ;SEARCH OR DELETE IS WILD
2951  =
2952  =                       GET_HASH:       ; fill in hash
2953  =                       ;--------
2954  =                       ;       entry:  BX = .info_fcb or .dir_fcb
2955  =                       ;       exit:   hash(*) = hash code for fcb
2956  =
2957  =0E6B 8BF3                    MOV SI,BX               ;SI = .FCB(0)
2958  =0E6D AC                      lods al
2959  =0E6E A24B08                  MOV HASH,AL             ;HASH(0) = USER #
2960  =0E71 33DB                    XOR BX,BX               ;DL,BH,BL = 3 BYTE NAME HASH
2961  =0E73 2420                    AND AL,20H              ;IS FCB(0) = E5H,20H,21H?
2962  =0E75 7406          0E7D      jz get_hash0            ;no
2963  =0E77 800E4B0810              or byte ptr hash,10h    ;???1????? = hash code for E5h,20h,21h
2964  =0E7C C3                      ret
2965  =                       GET_HASH0:
2966  =0E7D 8AD0                    mov dl,al
2967  =0E7F B90800                  MOV CX,11               ;LOOP COUNT
2968  =                       GET_HASH1:
2969  =0E82 80F906                  CMP CL,6                ;IS LOOP COUNT = 6?
2970  =0E85 7412          0E99      je GET_HASH2            ;YES
2971  =0E87 80F904                  CMP CL,4                ;IS LOOP COUNT = 4?
2972  =0E8A 740D          0E99      je GET_HASH2            ;YES
2973  =0E8C D1E3                      shl bx,1              ;HASH = shl(HASH,1)
2974  =0E8E D0D2                      rcl dl,1
2975  =0E90 F6C101                    TEST CL,1             ;IS LOOP COUNT EVEN?
2976  =0E93 7504          0E99        JNZ GET_HASH2       ;NO
2977  =0E95 D1E3                        shl bx,1          ;HASH = shl(HASH,1)
2978  =0E97 D0D2                        rcl dl,1
2979  =                       GET_HASH2:
2980  =0E99 AC                      lods al                 ;I = I + 1
2981  =0E9A 247F                    AND al,7FH              ;HASH = HASH +
2982  =0E9C 2C20                    SUB al,20H              ; (FCB(I) & 7FH) -20H
2983  =0E9E D0C8                    ROR al,1                ; SHIFTED RIGHT ONE IF EVEN
2984  =0EA0 7302          0EA4      JNC GET_HASH3
2985  =0EA2 D0C0                      ROL al,1
2986  =                       GET_HASH3:
2987  =0EA4 32E4                    XOR ah,ah
```

```
2988
2989  =0EA6 03D8                        ADD BX,ax
2990  =0EA8 80D200                      ADC dl,0
2991  =0EAB E2D5          0E82          LOOP GET_HASH1
2992  =
2993  =0EAD 891E4C08                    MOV WORD PTR HASH+1,BX          ;HASH(1,2) = BX
2994  =0EB1 BB4B08                      MOV BX,OFFSET HASH
2995  =0EB4 80E203                      AND dl,3
2996  =0EB7 D0CAD0CA                    ror dl,1| ror dl,1
2997  =0EBB 0817                        OR [BX],dl                      ;HASH(0) = HASH(0) | ROR((AL & 3),2)
2998  =0EBD AC                          LODS AL
2999  =0EBE 241F                        AND AL,1FH                      ;AL = FCB(EXT) & 1FH
3000  =0EC0 46                          INC SI
3001  =0EC1 8A24                        MOV AH,[SI]                     ;AH = FCB(MOD) & 3FH
3002  =0EC3 80E43F                      AND AH,3FH
3003  =0EC6 B103                        MOV CL,3
3004  =0EC8 D2E0                        shl AL,CL
3005  =0ECA D3E8                        shr AX,CL                       ;AX = MOD || EXT
3006  =0ECC 8A168C08                    MOV dl,EXTMSK
3007  =0ED0 D1E0                        shl ax,1                        ;setup to shr(ax,extshf)
3008  =                         GET_HASH5:
3009  =0ED2 D1E8                        shr ax,1
3010  =0ED4 D0EA                        shr dl,1
3011  =0ED6 72FA          0ED2          jc get_hash5                    ;AX = SHR(AX,EXTSHF)
3012  =
3013  =0ED8 80E401                      AND AH,1                        ;AH = AH & 1
3014  =0EDB D2CC                        ROR AH,CL                       ;CL = 3
3015  =0EDD 0827                        OR [BX],AH                      ;HASH(0) = HASH(0) | ROR(AH,3)
3016  =0EDF 884703                      MOV 3[BX],AL                    ;HASH(3) = AL
3017  =0EE2 C3                  RET8D:  RET
3018  =
3019  =
3020  =                         SEARCH_HASH:    ;search hash table for hash(*) of length hashl
3021  =                         ;---------------
3022  =                         ;       exit:   Z flag = 0 if not successful
3023  =
3024  =0EE3 833EB60800                  CMP HASH_SEG,0                  ;DOES HASH TBL EXIST ON DRIVE?
3025  =0EE8 74F8          0EE2          JE RET8D                        ;NO
3026  =0EEA A08A08                      MOV AL,searchL                  ;DOES SEARCHL = 0?
3027  =0EED 0AC0                        OR AL,AL
3028  =0EEF 74F1          0EE2          JZ RET8D                        ;YES
3029  =0EF1 803E4F08FF                  CMP HASHL,0FFH                  ;IS HIGHL = 0FFH?
3030  =0EF6 74EA          0EE2          JZ RET8D                        ;YES - SEARCH_HASH TEMPORARILY DISABLED
3031  =0EF8 8B1EA608                    MOV BX,CDRMAXA
3032  =0EFC 8B0F                        MOV cx,[BX]
3033  =0EFE FEC8                        DEC AL                          ;IF SEARCHL <> 1
3034  =0F00 7504          0F06          JNZ SEARCH_H0                   ;   THEN CX = CDRMAX
3035  =0F02 8B0E8F08                      MOV cx,DIRMAX                 ;   ELSE CX = DIRMAX
3036  =                         SEARCH_H0:
3037  =0F06 8B1E8F08                    MOV bx,DCNT
3038  =0F0A 2BC8                        SUB cx,bx                       ;CX = CX - DCNT = loop count
3039  =0F0C 74D4          0EE2          JZ RET8D
3040  =0F0E 8E068B08                    MOV ES,HASH_SEG
```

```
3041
3042  =
3043  =0F12 43                        inc bx                          ;DCNT = DCNT + 1
3044  =0F13 8BF8                      MOV di,bx                       ;SAVE DCNT + 1
3045  =0F15 D1E7D1E7                  shl di,1l shl di,1              ;multiply by 4
3046  =0F19 83EF04                    sub di,4                        ;setup for search_h2
3047  =                                                               ;DI = .HASH(DCNT)
3048  =                                                               ;BX = DCNT + 1
3049  =                                                               ;CX = LOOP COUNT
3050  =0F1C E80500        0F24        CALL SEARCH_H2                  ;Z FLAG SET IF FOUND
3051  =
3052  =0F1F 8CD8                      MOV AX,DS
3053  =0F21 8EC0                      MOV ES,AX
3054  =0F23 C3                        RET
3055  =
3056  =                           SEARCH_H2:
3057  =0F24 83C704                    ADD DI,4
3058  =0F27 BE4808                    MOV SI,OFFSET HASH
3059  =0F2A AC                        lods al                         ;DO USER #'S MATCH?
3060  =0F2B 263205                    XOR AL,ES:[DI]
3061  =0F2E 8AD0                      MOV dl,AL
3062  =0F30 241F                      AND AL,1FH
3063  =0F32 751E          0F52        JNZ SEARCH_H4                   ;NO
3064  =0F34 E87F00        0FB6          CALL SEARCH_H7                ;DO HASH CODES MATCH?
3065  =0F37 745A          0F93          JZ SEARCH_H5                  ;YES
3066  =                           SEARCH_H3:
3067  =0F39 43                        INC BX                          ;DCNT = DCNT + 1
3068  =0F3A E2E8          0F24        LOOP SEARCH_H2
3069  =
3070  =                           ;       UNSUCCESSFUL HASH SEARCH
3071  =
3072  =0F3C 833E8F08FF                  CMP DCNT,0FFFFH               ;WAS SEARCH FROM BEGINNING OF DIR?
3073  =0F41 750E          0F51        JNZ RET8E                       ;NO
3074  =0F43 8CD88EC0                  mov ax,ds! mov es,ax            ;restore ES for tst_log_fxs
3075  =0F47 E857F8        07A1        CALL TST_LOG_FXS                ;IS DRV REM & FX = 15,17,13,22,23,30?
3076  =0F4A 7505          0F51        JNZ RET8E                       ;NO
3077  =0F4C C6064F08FF                MOV HASHL,0FFH                  ;DISABLE HASH TABLE SEARCH
3078  =0F51 C3                    RET8E:    RET
3079  =
3080  =                           SEARCH_H4:
3081  =0F52 A01608                    MOV AL,BYTE PTR XDCNT+1         ;DOES XDCNT+1 = OFFH?
3082  =0F55 FEC0                      INC AL
3083  =0F57 7508          0F61        jnz search_h41                  ;no
3084  =0F59 26803DF5                    cmp es:b[di],0f5h             ;is hash empty entry
3085  =0F5D 75DA          0F39        jne search_h3                   ; no
3086  =0F5F EB1B          0F7C        jmps search_h42                 ; yes - save dcnt in xdcnt
3087  =                           SEARCH_H41:
3088  =0F61 FEC0                      INC AL                          ;DOES XDCNT+1 = OFFH?
3089  =0F63 75D4          0F39        JNZ SEARCH_H3                   ;NO
3090  =0F65 E84E00        0FB6        CALL SEARCH_H7                  ;DO HASH CODES MATCH?
3091  =0F68 75CF          0F39        JNZ SEARCH_H3                   ;NO
3092  =0F6A A01408                    mov al,find_xfcb
3093  =0F6D FEC0                      inc al                          ;does FIND_XFCB = OFFh?
```

```
3094
3095  =0F6F 7511          0F82          jnz search_h43                  ; no
3096  =0F71 26F60510                      test es:b[di],10h             ;is hash entry an xfcb
3097  =0F75 74C2          0F39          jz search_h3                    ; no
3098  =0F77 F6C20F                       test dl,0Fh                   ;does user field match?
3099  =0F7A 75BD          0F39          jnz search_h3                   ; no
3100  =                           SEARCH_H42:
3101  =0F7C 891E1508                     mov xdcnt,bx                  ; yes - xdcnt = dcnt
3102  =0F80 EB07          0F39          jmps search_h3
3103  =                           SEARCH_H43:
3104  =0F82 FEC0                         inc al                        ;does find_xfcb = 0FEh?
3105  =0F84 7507          0F8D          jnz search_h45                  ; no
3106  =0F86 F6C20F                       test dl,0Fh                   ;does user field match?
3107  =0F89 75AE          0F39          jnz search_h3                   ; no
3108  =0F8B EB06          0F93          jmps search_h5                  ; yes - save dcnt
3109  =                           SEARCH_H45:
3110  =0F8D 26F6051F                     test es:b[di],01Fh            ;IS HASH ENTRY FOR USER 0?
3111  =0F91 75A6          0F39          jnz SEARCH_H3                 ;NO
3112  =                           SEARCH_H5:                          ;save dcnt
3113  =0F93 8B168F08                     MOV DX,DCNT                   ;HASH TABLE MATCH
3114  =0F97 4B                          DEC BX
3115  =0F98 891E8F08                     MOV DCNT,BX
3116  =0F9C 8AC3                         MOV AL,BL
3117  =0F9E 2403                         AND AL,3
3118  =0FA0 3C03                         CMP AL,3                      ;DOES DCNT & 3 = 3?
3119  =0FA2 742E          0FD2          JZ RET8F                      ;YES
3120  =0FA4 80E3FC                      AND BL,0FCH
3121  =0FA7 80E2FC                      AND DL,0FCH                   ;DOES PREVIOUS DCNT = _
3122  =0FAA 3BDA                        CMP BX,DX                     ;NEW DCNT?
3123  =0FAC 7424          0FD2          JZ RET8F                      ; yes
3124  =0FAE 800E10080F                   or rd_dir_flag,0fh            ; no - locate new directory record
3125  =0FB3 32C0                        XOR AL,AL                     ;SET ZERO FLAG
3126  =0FB5 C3                          RET
3127  =
3128  =                           SEARCH_H7:
3129  =0FB6 A04F08                       MOV AL,HASHL                  ;ALL ENTRIES MATCH IF
3130  =0FB9 0AC0                         OR AL,AL                      ;does HASHL = 0?
3131  =0FBB 7415          0FD2          JZ RET8F                      ; yes
3132  =0FBD B4E0                         mov ah,0E0h                   ;if hashl = 3 then mask = 0F0h
3133  =0FBF 3C037402      0FC5          cmp al,3l je search_h8         ;else mask = 0D0h
3134  =0FC3 B4C0                           mov ah,0C0h                 ; - hashl = 2 - dont test s field
3135  =                           search_h8:
3136  =0FC5 8404                         test dl,ah                    ;do mask bits match
3137  =0FC7 7509          0FD2          JNZ RET8F                     ;no
3138  =0FC9 32E4                         XOR AH,AH
3139  =0FCB 91                           XCHG AX,CX                    ;CX = hash length
3140  =0FCC 57                           PUSH DI                       ;COMPARE HASH ENTRIES
3141  =0FCD 47                           INC DI
3142  =0FCE F3A6                         REPE CMPS AL,AL               ;Z FLAG SET IF MATCH
3143  =0FD0 91                           XCHG AX,CX
3144  =0FD1 5F                           POP DI
3145  =0FD2 C3                    RET8F: RET
3146  =
```

```
3147
3148  =                       FIX_HASH:
3149  =                       ;--------
3150  =0FD3 833E860800                CMP HASH_SEG,0
3151  =0FD8 74F8          0FD2        JZ RET8F
3152  =0FDA FF364808                  PUSH WORD PTR HASH              ;SAVE CURRENT HASH
3153  =0FDE FF364D08                  PUSH WORD PTR HASH+2
3154  =0FE2 E840F7        0725        CALL GET_DPTRA
3155  =0FE5 E883FE        0E6B        CALL GET_HASH                   ;COMPUTE HASH FROM DIR ENTRY
3156  =0FE8 A18F08                    MOV AX,DCNT                     ;MOVE TO HASH_TABLE(DCNT)
3157  =0FEB D1E0D1E0                  shl ax,1! shl ax,1
3158  =0FEF 8BF8                      MOV DI,AX
3159  =0FF1 8E06B608                  MOV ES,HASH_SEG
3160  =0FF5 BE4808                    MOV SI,OFFSET HASH
3161  =0FF8 B90200                    mov cx,2
3162  =0FFB F3A5                      REP MOVS ax,ax
3163  =0FFD 8CD8                      MOV AX,DS
3164  =0FFF 8EC0                      MOV ES,AX
3165  =1001 8F064D08                  POP WORD PTR HASH+2             ;RESTORE CURRENT HASH
3166  =1005 8F064808                  POP WORD PTR HASH
3167  =1009 C3                        RET
3168  =
3169  =                       if BMPM
3170  =
3171  =                       swap:           ;swap sdcnt with sdcnt0
3172  =                       ;----
3173  =100A A10909                    mov ax,sdcnt
3174  =100D 87060B09                  xchg ax,sdcnt0
3175  =1011 A30909                    mov sdcnt,ax
3176  =1014 C3                        ret
3177  =                       endif
3178  =
3179  =                       save_dcnt_pos1:
3180  =                       ;--------------
3181  =1015 803E1608FF                cmp byte ptr xdcnt+1,0ffh
3182  =101A 7586          0FD2        jne ret8f
3183  =
3184  =                       save_dcnt_pos:
3185  =                       ;-------------
3186  =101C A18F08                    mov ax,dcnt
3187  =101F A31508                    mov xdcnt,ax
3188  =1022 C3                        ret
3189  =
3190  =                       searchi:        ;search initialization
3191  =                       ;-------
3192  =1023 BB3C0B                    mov bx,offset info_fcb
3193  =1026 891E8308                  mov searcha,bx                  ;searcha = .info_fcb
3194  =
3195  =                       searchi1:
3196  =102A C6060F09FF                mov dirloc,0ffh                 ;changed if actually found
3197  =102F 880E8A08                  mov searchl,cl                  ;searchl = cl
3198  =1033 E802FE        0E38        CALL SET_HASH
3199  =1036 C3                        ret
```

```
3200
3201  =
3202  =                         search_1_name:  ;search for first fcb matching name
3203  =                         ;--------------
3204  =1037 33C0                        xor ax,ax
3205  =1039 A2480B                      MOV info_fcb+extnum,al
3206  =103C A24A0B                      MOV info_fcb+modnum,al
3207  =
3208  =                         search_name:    ;search matching name
3209  =                         ;------------
3210  =103F B10F                        mov cl,namlen
3211  =1041 E802          1045          jmps search
3212  =
3213  =                         search_ext:
3214  =                         ;----------
3215  =1043 B10C                        mov cl,extnum
3216  =
3217  =                         search:         ;search for directory element at .info_fcb
3218  =                         ;------
3219  =                         ;       entry:  CL = search length
3220  =
3221  =1045 E8DBFF        1023         call searchi
3222  =                         search1:                                ;entry point used by rename
3223  =1048 E831F7        077C         call setenddir                  ;dcnt = enddir
3224  =                                                                 ;to start at the beginning
3225  =                                                                 ;(drop through to searchn)
3226  =                         ;
3227  =                         searchn:
3228  =                         ;-------
3229  =                         ;       search for the next directory element, assuming
3230  =                         ;       a previous call on search which sets searcha and searchh1
3231  =                         ;       exit:   Z flag = 0 for successful searches
3232  =
3233  =                         if BMPM
3234  =104B 32C0                        xor al,al
3235  =104D 86062309                    xchg user_zero_pass,al
3236  =1051 84C0                        test al,al
3237  =1053 7403          1058          jz searchn1                     ;if user_zero_pass then do;
3238  =                                                                 ;  user_zero_pass = false;
3239  =1055 E882FF        100A            call swap                     ;  call swap;
3240  =                                                                 ;  end;
3241  =                         searchn1:
3242  =
3243  =                         endif
3244  =                         if BCPM
3245  =                                 mov user_zero_pass,false
3246  =                         endif
3247  =
3248  =1058 E888FE        0EE3         CALL SEARCH_HASH
3249  =105B 754C          10A9         JNZ searchFIN
3250  =
3251  =105D B100                        mov cl,false
3252  =105F E885FB        0BE7          call read_dir                  ;read next dir element
```

```
3253
3254  =1062 E810F7        0775          call endofdir
3255  =1065 7442          10A9          jz searchfin
3256  =                                                    ;skip to end if so
3257  =                                                    ;not end of directory,
3258  =                                                    ;scan for match
3259  =1067 8B158308                    mov dx,searcha          ;dx=beginning of user fcb
3260  =106B 8BF2                        mov si,dx
3261  =106D AC                          lods al                 ;first character
3262  =106E 3CE5                        cmp al,empty            ;keep scanning if empty
3263  =1070 7407          1079          jz searchnext
3264  =                                                    ;not empty, may be end of
3265  =                                                    ;logical directory
3266  =1072 52                            push dx               ;save search address
3267  =1073 E817F7        078D            call compcdr            ;past logical end?
3268  =1076 5A                            pop dx                ;recall address
3269  =1077 7330          10A9            jnb searchfin
3270  =                                                    ;artificial stop
3271  =                           searchnext:
3272  =1079 E8A9F6        0725          call get_dptra          ;bx = buffa+dptr
3273  =107C 8A0E8A08                    mov cl,searchl          ;length of search to cl
3274  =1080 32ED                        xor ch,ch               ;ch counts up, cl counts down
3275  =
3276  =1082 803FE5                      cmp b[bx],empty         ;is dir fcb empty
3277  =1085 7503          108A          jnz $+5                 ;no
3278  =1087 E888FF        1015            call save_dcnt_pos1    ;yes - save dcnt
3279  =108A C6060B0800                  mov save_xfcb,false
3280  =108F 8A07                        mov al,[bx]             ;is fcb an xfcb ?
3281  =1091 24EF                        and al,11101111b
3282  =1093 3A07                        cmp al,[bx]
3283  =1095 7421          1088          je searchloop           ;no
3284  =1097 8BF2                        mov si,dx
3285  =1099 3A04                        cmp al,[si]             ;does user # match ?
3286  =109B 751B          1088          jnz searchloop          ;no
3287  =109D A01408                      MOV AL,FIND_XFCB
3288  =10A0 0AC0                        OR AL,AL                ;IS FIND_XFCB "= 0
3289  =10A2 74A7          1048          jz searchn              ; no
3290  =10A4 A20B08                      MOV SAVE_XFCB,AL        ;
3291  =10A7 EB66          110F          jmps searchok
3292  =
3293  =                           searchfin:                    ;end of directory, or empty name
3294  =10A9 E869FF        1015          call save_dcnt_pos1
3295  =10AC E8CDF6        077C          call setenddir          ;may be artifical end
3296  =                           lret_eq_ff:
3297  =10AF B0FF                        mov al,255
3298  =10B1 8AE8                        mov ch,al
3299  =10B3 FEC5                        inc ch                  ;zero flag set on unsuccessful
3300  =10B5 E9F6F3        04AE          jmp set_lret            ;searches
3301  =
3302  =                           searchloop:
3303  =10B8 0AC9                        or cl,cl
3304  =10BA 7458          1117          jz endsearch
3305  =10BC 8BF2                        mov si,dx
```

```
3306
3307  =108E AC                          lods al                         ;fcb character
3308  =108F 247F                        and al,07fh
3309  =10C1 3C3F                        cmp al,'?'
3310  =10C3 744A          110F          jz searchok                     ;? matches all
3311  =                                                                 ;scan next character if not
3312  =                                                                 ;chksum byte
3313  =10C5 80FD0D                      cmp ch,cnksum
3314  =10C8 7445          110F          jz searchok
3315  =                                                                 ;not the ubytes field,
3316  =                                                                 ;extent field
3317  =10CA 80FD0C                      cmp ch,extnum                   ;may be extent field
3318  =10CD 740F          10DE          jz chk_ext                      ;skip to search extent
3319  =
3320  =10CF 80FD0E                        cmp ch,modnum                 ;is field module # ?
3321  =10D2 7502          10D6            jnz not_modnum                ;no
3322  =10D4 243F                            and al,3fh                  ;yes - mask off high order bits
3323  =                                 not_modnum:
3324  =10D6 2A07                          sub al,[bx]
3325  =10D8 247F                          and al,7fh                    ;mask-out flags/extent modulus
3326  =10DA 755C          1138            jnz searchnm
3327  =10DC EB31          110F            jmps searchok                 ;matched character
3328  =
3329  =                                 chk_ext:                        ;AL = fcb character
3330  =                                                                 ;attempt an extent # match
3331  =                                 if BMPM
3332  =10DE 803E0A09FF                    cmp byte ptr sdcnt+1,true     ;if sdcnt+1 = ff then do;
3333  =10E3 7508          10ED            jne dont_save                 ;  sdcnt = dcnt
3334  =10E5 8B368F08                        mov si,dcnt
3335  =10E9 89360909                        mov sdcnt,si                ;  end;
3336  =                                 dont_save:
3337  =
3338  =                                 endif
3339  =
3340  =10ED 51                            push cx                       ;save counters
3341  =10EE 8A0F                          mov cl,[bx]                   ;directory character to c
3342  =10F0 E8E9F4        05DC            call compext                  ;compare user/dir char
3343  =10F3 59                            pop cx                        ;recall counters
3344  =10F4 7532          1128            jnz searchn_jmp               ;skip if no match
3345  =10F6 F6062309FF                    test user_zero_pass,true
3346  =10FB 740D          110A            jz not_user0                  ;if user0_pass and
3347  =10FD 4343                            inc bx! inc bx
3348  =10FF 803F00                          cmp b[bx],0                   ; matching extent & modnum = 0
3349  =1102 7524          1128            jne searchn_jmp
3350  =1104 E815FF        101C            call save_dcnt_pos            ;save directory position
3351  =1107 E941FF        1048            jmp searchn                   ; of matching fcb
3352  =                                 not_user0:
3353  =
3354  =                                   ;disable search under user zero if any fcb is found under
3355  =                                   ;the current user number
3356  =
3357  =110A C606120800                    mov search_user0,false
3358  =                                 searchok:                               ;current character matches
```

```
3359
3360  =110F 4243                    inc dx! inc bx
3361  =1111 FEC5FEC9                inc ch! dec cl
3362  =1115 EBA1          1088      jmps searchloop
3363  =
3364  =                         endsearch:              ;entire name matches, return dir position
3365  =1117 803E0B08FF              CMP SAVE_XFCB,0FFH              ;is save_xfcb true
3366  =111C 750D          1128      JNZ ENDSEARCH1                  ;NO - 0 OR 0FEH
3367  =111E 803E1608FE                cmp byte ptr xdcnt+1,0feh     ;if xdcnt+1 = fe then
3368  =1123 7503          1128        jne searchn_jmp             ;  call save_dcnt_pos
3369  =1125 E8F4FE        101C          call save_dcnt_pos
3370  =                         searchn_jmp:
3371  =1128 E920FF        1048        jmp searchn
3372  =                         endsearch1:
3373  =112B 32C0                    xor al,al
3374  =112D A20F09                  mov dirloc,al
3375  =1130 A20D08                  mov lret,al
3376  =1133 8AE8                    mov ch,al
3377  =1135 FEC5                    inc ch                          ;zero flag reset on successful
3378  =1137 C3                ret9: ret                             ;searches
3379  =
3380  =                         searchnm:                           ;search no match routine
3381  =1138 0A2F                    or ch,[bx]                      ;is fcb(0) and dir fcb(0) = 0 ?
3382  =113A 75EC          1128      jnz searchn_jmp                 ; no
3383  =113C F6061208FF              test search_user0,true          ;is search_user0 true
3384  =1141 74E5          1128      jz searchn_jmp
3385  =1143 C6062309FF              mov user_zero_pass,true
3386  =
3387  =                         if BMPM
3388  =1148 E8BFFE        100A      call swap
3389  =                         endif
3390  =
3391  =1148 EBC2          110F      jmps searchok
3392  =
3393  =                         init_xfcb_search:
3394  =                         ;----------------
3395  =114D B0FF                    mov al,0ffh                     ;find_xfcb = ffh
3396  =
3397  =                         init_xfcb_search1:
3398  =                         ;-----------------
3399  =                         ;       entry:  AL = value to set find_xfcb
3400  =
3401  =114F A21408                  mov find_xfcb,al
3402  =1152 B0FE                    mov al,0feh
3403  =1154 A21608                  mov byte ptr xdcnt+1,al         ;xdcnt+1 = feh
3404  =1157 C3                RET9A: ret
3405  =
3406  =                         does_xfcb_exist:
3407  =                         ;---------------
3408  =1158 803E1608FE              cmp byte ptr xdcnt+1,0feh       ;if xdcnt+1 = feh then return
3409  =115D 7408          1137      je ret9
3410  =115F E821F6        0783      call set_dcnt                   ;dcnt = xdcnt - 1
3411  =1162 32C0                    xor al,al
```

```
3412
3413  =1164 E8E8FF      114F        call init_xfcb_search1
3414  =1167 881E8308                mov bx,searcha                ;of 10h into fcb(0)
3415  =116B 800F10                  or b[bx],10h
3416  =116E 810C                    mov cl,extnum
3417  =1170 E887FE      102A        call searchi1                 ;search for xfcb
3418  =1173 E9D5FE      1048        jmp searchn                   ;z flag set if unsuccessful
3419  =
3420  =                       XDCNT_EQ_DCNT:                      ;SAVE FCB'S DCNT IN XDCNT
3421  =                       ;-------------
3422  =1176 A18F08                  MOV AX,DCNT
3423  =1179 A31508                  MOV XDCNT,AX
3424  =117C C3                      RET
3425  =
3426  =                       RESTORE_DIR_FCB:                    ;REPOSITION SEARCH TO FCB
3427  =                       ;---------------
3428  =117D E803F6      0783        CALL SET_DCNT                 ;WHOSE DCNT IS SAVED IN XDCNT
3429  =1180 B10F                    MOV CL,NAMLEN
3430  =1182 E89EFE      1023        CALL SEARCHI
3431  =1185 E9C3FE      1048        JMP SEARCHN
3432  =
3433  =                       delet:          ;delete the currently addressed file
3434  =                       ;-----
3435  =1188 E8F4F3      057F        CALL GET_ATTS                 ;F5' = DELETE XFCB'S ONLY
3436  =                       deletex:
3437  =118B B0FE                    MOV AL,0FEH                   ;      RETAIN FILE'S LOCK
3438  =118D E88FFF      114F        CALL INIT_XFCB_SEARCH1        ;HAVE SEARCH RETURN FCB'S & XFCB'S
3439  =
3440  =                       ; DELETE PASS 1 - CHECK R/O ATTRIBUTES AND XFCB PASSWORDS
3441  =
3442  =1190 E8B0FE      1043        CALL search_ext
3443  =1193 74C2        1157        JZ RET9A
3444  =                       DELETE00:
3445  =1195 E88DF5      0725        CALL GET_DPTRA
3446  =1198 8A07                    MOV AL,[BX]
3447  =119A 2410                    AND AL,10H                    ;IS DIR FCB AN XFCB?
3448  =119C 751E        11BC        JNZ DELETE01                  ;YES
3449  =
3450  =                       if BMPM
3451  =119E E8250C      1DC6          CALL TST_OLIST
3452  =                       endif
3453  =
3454  =11A1 F606DE0880              TEST ATTRIBUTES,80H           ;DELETE XFCB'S ONLY?
3455  =11A6 7503        11AB        JNZ $+5                       ;NO
3456  =11A8 E88DF5      0738          CALL CKRODIR                ;CHECK FILE R/O ATTRIBUTE
3457  =11AB E83009      1ADE        CALL GET_DIR_MODE             ;HAVE PASSWORDS BEEN ENABLED BY
3458  =11AE D0C0                    ROL AL,1                      ;DIRECTORY LABEL?
3459  =11B0 7218        11CD        JC DELETE02                   ;YES
3460  =11B2 BB3C0B                  MOV BX,offset info_fcb
3461  =11B5 E861FC      0E19        CALL CHK_WILD                 ;IS THIS A WILD CARD DELETE?
3462  =11B8 7413        11CD        JZ DELETE02                   ;YES
3463  =11BA EB1E        11DA        JMPS DELETE11                 ;NO NEED FOR TWO PASSES
3464  =                       DELETE01:
```

```
3465
3466  =11BC E81F09       1ADE          CALL GET_DIR_MODE              ;HAVE PASSWORDS BEEN ENABLED BY
3467  =11BF D0C0                       ROL AL,1                       ;DIRECTORY LABEL?
3468  =11C1 730A         11CD          JNC delete02                   ;NO
3469  =11C3 E8440A       1C0A            CALL CHK_XFCB_PASSWORD       ;CHECK XFCB PASSWORD
3470  =11C6 7405         11CD            jz delete02
3471  =11C8 E86709       1832              call chk_pw_error
3472  =11CB EB6E         1138              jmps deletex
3473  =                            DELETE02:
3474  =11CD E878FE       1048          CALL SEARCHN
3475  =11D0 75C3         1195          jnz DELETE00
3476  =
3477  =                            ; DELETE PASS 2 - DELETE ALL MATCHING FCB'S AND/OR XFCB'S
3478  =
3479  =11D2 E86EFE       1043          CALL search_ext
3480  =                            DELETE10:
3481  =11D5 7503         11DA          JNZ DELETE11
3482  =11D7 E939FC       0E13            JMP CPYDIRLOC                ;DELETE FINISHED
3483  =                            DELETE11:
3484  =11DA E848F5       0725          CALL GET_DPTRA
3485  =11DD 8A07                       MOV AL,[BX]
3486  =11DF 2410                       AND AL,10H                     ;IS DIR FCB AN XFCB?
3487  =11E1 750C         11EF          JNZ DELETE12                   ;YES
3488  =
3489  =                            if BMPM
3490  =11E3 53                           PUSH BX
3491  =11E4 E8E608       1DCD            CALL CHK_DLIST
3492  =11E7 5B                           POP BX
3493  =                            endif
3494  =
3495  =11E8 F606DE0880                   TEST ATTRIBUTES,80H            ;XFCB ONLY DELETE?
3496  =11ED 7503         11F2            JNZ DELETE13                   ;YES
3497  =
3498  =                            DELETE12:
3499  =11EF C607E5                       MOV BCBX],0E5H                 ;DELETE DIR XFCB
3500  =                            delete13:
3501  =11F2 9C                           pushf
3502  =11F3 E8400A       1C43            call get_dtba8
3503  =11F6 0AC07502     11FC            or al,all jnz delete14
3504  =11FA 8807                           mov [bx],al
3505  =                            delete14:
3506  =11FC E80EFA       0C0D          CALL WRDIR                     ;UPDATE DIRECTORY RECORD
3507  =11FF 9D                           popf
3508  =1200 7505         1207          JNZ delete15
3509  =1202 B100                         MOV CL,0
3510  =1204 E838FB       0D42            CALL scndmab                 ;RETURN BLOCKS IF FCB
3511  =                            delete15:
3512  =1207 E8C9FD       0FD3          CALL FIX_HASH                  ;UPDATE DIRECTORY HASH TABLE
3513  =120A E83EFE       1048          CALL SEARCHN
3514  =120D EBC6         11D5          jmps DELETE10
3515  =
3516  =                            getblock:
3517  =                            ;--------
```

```
3518
3519  =                     ;       given allocation vector position cx, find the zero bit
3520  =                     ;       closest to this position by searching right 1st then left.
3521  =                     ;       if found, set the bit to 1 and return the bit position
3522  =                     ;       in bx.   if not found (i.e., we pass 0 on the left, or
3523  =                     ;       maxall on the right), return 0000 in bx
3524  =                     ;       entry:  CX = starting block number to search from
3525  =                     ;       exit:   BX = available block number, 0000 if not found
3526  =
3527  =120F 8BD1                    mov dx,cx                       ;copy start pos. to dx
3528  =                         rightst:
3529  =1211 3B168D08                mov dx,maxall                   ;value of maximum allocation#
3530  =1215 732A          1241      jae rblok0                      ;return block 0000 if so
3531  =1217 42                        inc dx
3532  =1218 5152                    push cx! push dx                ;left, right pushed
3533  =121A 8BCA                    mov cx,dx                       ;ready right for call
3534  =121C E893FA        0CB2      call getallocbit
3535  =121F D0D8                    rcr al,1
3536  =1221 7314          1237      jae rblok                       ;return block number if zero
3537  =                                                             ;bit is one, so try the right
3538  =1223 5A59                    pop dx! pop cx                  ;left, right restored
3539  =                         lefttst:
3540  =1225 0BC9                    or cx,cx
3541  =1227 74E8          1211      jz righttst                     ;skip if left=0000
3542  =                                                             ;left not at position zero,
3543  =1229 49                        dec cx                        ;bit zero ?
3544  =122A 5251                    push dx! push cx                ;left,right pushed
3545  =122C E883FA        0CB2      call getallocbit
3546  =122F D0D8                    rcr al,1
3547  =1231 7304          1237      jae rblok                       ;return block number if zero
3548  =1233 595A                      pop cx! pop dx                ;restore left and right ptr
3549  =1235 EBDA          1211        jmps righttst                 ;for another attempt
3550  =                         rblok:
3551  =1237 D0D0                    rcl al,1
3552  =1239 FEC0                    inc al                          ;bit back into position
3553  =                                                             ;and set to 1
3554  =                                                             ;dl contains the number of
3555  =                                                             ;shifts required to reposition
3556  =123B E895FA        0CD3      call rotr                       ;move bit back to position
3557  =                                                             ;and store
3558  =123E 5B5A                    pop bx! pop dx                  ;bx returned value, dx discarded
3559  =1240 C3                      ret
3560  =                         rblok0:
3561  =1241 0BC9                    or cx,cx
3562  =1243 75E0          1225      jnz lefttst
3563  =1245 8BD9                    mov bx,cx                       ;cannot find an available
3564  =1247 C3                      ret                             ;bit, return 0000
3565  =
3566  =                         copy_dir2:
3567  =                         ;---------
3568  =1248 52                      push dx                         ;save length for later
3569  =1249 B500                    mov ch,0                        ;double index to cx
3570  =124B BA3C0B                  mov dx,offset info_fcb          ;bx = source for data
```

```
3571
3572  =124E 0301                    add dx,cx                       ;dx=.fcb(cl), source for copy
3573  =1250 E802F4      0725        call get_dptra                  ;bx=.buff(dptr), destination
3574  =1253 59                      pop cx                          ;dx=source, bx=dest, c=length
3575  =1254 C3                      ret
3576  =
3577  =                         copy_dir:
3578  =                         ;--------
3579  =                         ;       copy fcb information starting
3580  =                         ;       into the currently addressed directory entry
3581  =
3582  =1255 B680                    mov dh,80h                      ;dh = attribute mask
3583  =                         copy_dir0:
3584  =                         ;---------
3585  =                         ;       entry:  DL = # of characters to copy
3586  =                         ;               DH = attribute mask
3587  =
3588  =1257 E8EEFF      1248        call copy_dir2                  ;init regs bx, cx, dx for copy
3589  =125A FEC1                    inc cl
3590  =                         copy_dir1:
3591  =125C FEC9                    dec cl
3592  =125E 7503        1263        JNZ $+5
3593  =1260 E9AAF9      0C0D          JMP wrdir                     ;copy operation complete
3594  =1263 8A27                    mov ah,[bx]                     ;pick up dir fcb byte
3595  =1265 22E5                    and ah,ch                       ;mask with attribute mask (0 | 80h)
3596  =1267 8BF2                    mov si,dx
3597  =1269 AC                      lods al                         ;pick up fcb byte
3598  =126A 247F                    and al,7fh                      ;clear attribute bit
3599  =126C 0AC4                    or al,ah                        ;or in dir fcb att & att mask
3600  =126E 8807                    mov [bx],al                     ;store in dir fcb byte
3601  =1270 43                      inc bx                          ;advance fcb ptr
3602  =1271 42                      inc dx                          ;advance dir fcb ptr
3603  =1272 EBE8        125C        jmps copy_dir1
3604  =
3605  =                         copy_user_no:
3606  =                         ;------------
3607  =1274 A03C0B                  mov al,info_fcb                 ;fcb(16) = fcb(0)
3608  =1277 BB4C0B                  mov bx,offset info_fcb+dskmap
3609  =127A 8807                    mov [bx],al                     ;bx = .fcb(16)
3610  =127C C3                      ret
3611  =
3612  =                         open:           ;search for the directory entry, copy to fcb
3613  =                         ;----
3614  =127D E88FFD      103F        call search_name
3615  =                         open1:
3616  =1280 7438        12BA        jz ret11                        ;return with lret=255 if end
3617  =                                                             ;not end of directory,copy fcb
3618  =                         opencopy:
3619  =                         ;--------                           ;(referenced below to copy fcb)
3620  =1282 E8D8F4      075D        call setfwf                     ;al = modnum | fwfmsk
3621  =1285 53                      push bx                         ;bx = .fcb(modnum)
3622  =1286 4B4B                    dec bx! dec bx
3623  =1288 8A27                    mov ah,[bx]                     ;ah = extnum
```

```
3624
3625  =128A 50                          push ax                         ;save extnum & modnum
3626  =128B E897F4        0725          call get_dptra
3627  =128E 8BD3                        mov dx,bx                       ;dx = .buff(dptr)
3628  =1290 BB3C0B                      mov bx,offset info_fcb          ;bx=.fcb(0)
3629  =1293 8120                        mov cl,nxtrec                   ;length of move operation
3630  =1295 E84AF2        04E2          call move                       ;from .buff(dptr) to .fcb(0)
3631  =                                                                 ;note that entire fcb is
3632  =                                                                 ;copied, including indicators
3633  =1298 E8FFF2        059A          call get_dir_ext
3634  =129B 8AC8                        mov cl,al                       ;cl = dir_ext
3635  =129D 585B                        pop ax! pop bx
3636  =129F 8807                        mov [bx],al                     ;restore modnum
3637  =12A1 4B4B                        dec bx! dec bx
3638  =12A3 8827                        mov [bx],ah                     ;restore extnum number
3639  =
3640  =                           ;     bx = .user extent#, cl = dir extent#
3641  =                           ;     if user ext < dir ext then user := 128 records
3642  =                           ;     if user ext = dir ext then user := dir records
3643  =                           ;     if user ext > dir ext then user := 0 records
3644  =
3645  =                           set_rc:
3646  =                           ;-------
3647  =                           ;     entry:  BX = .user extent #
3648  =                           ;             CL = dir extent #
3649  =
3650  =12A5 32ED                        xor  ch,ch
3651  =12A7 BE480B                      mov si,offset info_fcb+reccnt
3652  =12AA 8A07                        mov al,[bx]                     ;al = current extent
3653  =12AC 2AC1                        sub al,cl                       ;compare fcb ext to dir ext
3654  =12AE 740B          12BB          jz set_rc2                      ;fcb ext = dir ext
3655  =                                                                 ;  actual_rc = 0, fcb(rc) = fcb(rc)
3656  =12B0 8AC5                          mov al,ch
3657  =12B2 7304          12B8            jae set_rc1                   ;fcb ext > dir ext
3658  =                                                                 ;  actual_rc = 0, fcb(rc) = 0
3659  =
3660  =12B4 B080                            mov al,128                  ;fcb ext < dir ext
3661  =12B6 0A04                            or al,[si]                  ;  fcb(rc) = 128 | fcb(rc)
3662  =                           set_rc1:
3663  =12B8 8804                          mov [si],al
3664  =12BA C3                    ret11:  ret
3665  =
3666  =                           set_rc2:
3667  =12BB 3804                        cmp b[si],al                    ;is fcb(rc) = 0?
3668  =12BD 75FB          12BA          jnz ret11                       ;no
3669  =                           set_rc3:
3670  =12BF 32C0                        xor al,al                       ;AL = 0
3671  =12C1 8804                        mov b[si],al                    ;required by func 99
3672  =12C3 38061109                    cmp dminx,al                    ;do blocks exist in fcb?
3673  =12C7 74F1          12BA          jz ret11                        ;no
3674  =12C9 C60480                      mov b[si],80h                   ;fcb(rc) = 80h
3675  =12CC C3                          ret
3676  =
```

```
3677
3678  =                     restore_rc:
3679  =                     ;----------
3680  =                     ;       if actual_rc ~= 0 then fcb(rc) = actual_rc
3681  =                     ;       entry:  BX = .fcb(extnum)
3682  =
3683  =12CD A04B0B                 mov al,info_fcb+reccnt
3684  =12D0 3C81                   cmp al,81h
3685  =12D2 7205          12D9     jb restore_rc1
3686  =12D4 247F                     and al,7fh
3687  =12D6 A24B0B                   mov info_fcb+reccnt,al
3688  =                     restore_rc1:
3689  =12D9 C3                     ret
3690  =
3691  =                     mergezero:
3692  =                     ;---------
3693  =                     ;       if fcb1(i) = 0  then fcb1(i) := fcb2(i)
3694  =                     ;       entry:  BX = .fcb1(i)
3695  =                     ;               DX = .fcb2(i)
3696  =
3697  =12DA 833F00                 cmp w[bx],0
3698  =12DD 7505          12E4     jnz ret12                       ;rnz
3699  =12DF 8BF2                   mov si,dx
3700  =12E1 AD                     lods ax
3701  =12E2 8907                   mov [bx],ax
3702  =12E4 C3              ret12: ret
3703  =
3704  =                     close_fcb:
3705  =                     ;---------
3706  =12E5 F606C40880             test byte ptr chksiz+1,080h     ;is drive permanent
3707  =12EA 7505          12F1     jnz close_fcb1                  ; yes
3708  =12EC C6061008F0               mov rd_dir_flag,0f0h          ; no - must read directory to
3709  =                     close_fcb1:                             ; verify media has not changed
3710  =12F1 E848FD        103F     call search_name                ;locate file
3711  =12F4 74EE          12E4     jz ret12                        ;return if not found
3712  =                                                            ;merge the disk map at info_fcb
3713  =12F6 E82CF4        0725     call get_dptra
3714  =12F9 83C310                 add bx,dskmap
3715  =12FC 8BD3                   mov dx,bx                       ;DX = .buff(dptr+16)
3716  =12FE BB4C0B                 mov bx,offset info_fcb+dskmap   ;BX = .fcb(16)
3717  =1301 B110                   mov cl,(fcblen-dskmap)          ;length of single byte dm
3718  =                     merge0:
3719  =1303 803E120900             cmp single,0
3720  =1308 741B          1325     jz merged                       ;skip to double
3721  =
3722  =                     ;       this is a single byte map
3723  =                     ;       if fcb(i) = 0 then fcb(i) = buff(i)
3724  =                     ;       if buff(i) = 0 then buff(i) = fcb(i)
3725  =                     ;       if fcb(i) <> buff(i) then error
3726  =
3727  =130A 8A07                     mov al,[bx]
3728  =130C 0AC0                     or al,al
3729  =130E 8BF2                     mov si,dx
```

```
3730
3731  =1310 AC                          lods al
3732  =1311 7502        1315            jnz fcbnzero
3733  =                                                   ;fcb(i) = 0
3734  =1313 8807                          mov [bx],al     ;fcb(i) = buff(i)
3735  =                           fcbnzero:
3736  =1315 0AC0                        or al,al
3737  =1317 7506        131F            jnz buffnzero
3738  =                                                   ;buff(i) = 0
3739  =1319 8A07                          mov al,[bx]
3740  =131B 8BFA                          mov di,dx
3741  =131D FC                            cld
3742  =131E AA                            stos al         ;buff(i)=fcb(i)
3743  =                           buffnzero:
3744  =131F 3A07                        cmp al,[bx]
3745  =1321 7418        1338            jz dmset        ;fcb(i) = buff(i)?
3746  =1323 EB6E        1393            jmps mergerr    ;if not merge ok
3747  =                           merged:
3748  =                                                   ;this is a double byte merge
3749  =1325 E882FF      12DA          call mergezero    ;buff = fcb if buff 0000
3750  =1328 87DA                      xchg bx,dx
3751  =132A E8ADFF      12DA          call mergezero
3752  =132D 87DA                      xchg bx,dx        ;fcb = buff if fcb 0000
3753  =                                                 ;they should be identical
3754  =                                                 ;at this point
3755  =132F 8BF2                      mov si,dx
3756  =1331 8B04                      mov ax,[si]
3757  =1333 3B07                      cmp ax,[bx]
3758  =1335 755C        1393          jnz mergerr
3759  =1337 42                        inc dx
3760  =1338 43                        inc bx
3761  =                                                 ;merge ok for this pair
3762  =1339 FEC9                      dec cl            ;extra count for double byte
3763  =                           dmset:
3764  =133B 42                        inc dx
3765  =133C 43                        inc bx
3766  =133D FEC9                      dec cl
3767  =133F 75C2        1303          jnz merge0        ;for more
3768  =                                                 ;end of disk map merge,
3769  =                                                 ;check record count
3770  =                                                 ;dx = .buff(dptr)+32,
3771  =                                                 ;bx = .fcb(32)
3772  =1341 8BDA                      mov bx,dx
3773  =1343 83EB14                    sub bx,(fcblen-extnum)
3774  =1346 53                        push bx
3775  =1347 E850F2      059A          call get_dir_ext
3776  =134A 5E                        pop si            ;SI = .fcb(extnum),
3777  =                                                 ;BX = .buff(dptr+extnum)
3778  =134B 8A0C                      mov cl,[si]       ;CL = fcb extent number
3779  =134D 8A2F                      mov ch,[bx]       ;CH = buff extent number
3780  =
3781  =134F 8804                      mov [si],al       ;fcb(ext) = dir(ext)
3782  =1351 8807                      mov [bx],al       ;buff(dptr+ext) = dir(ext)
```

```
3783
3784  =1353 83C603                    add si,(reccnt-extnum)          ;SI = fcb(rc)
3785  =1356 83C303                    add bx,(reccnt-extnum)          ;BX = buff(rc)
3786  =
3787  =1359 3AC17511      136E        cmp al,cl! jne mrg_rc1          ;fcb(ext)~=dir(ext)=buff(ext)
3788  =135D 3AC5750F      1370          cmp al,ch! jne mrg_rc2        ;fcb(ext)=dir(ext)~=buff(ext)
3789  =
3790  =1361 8A04                       mov al,[si]                   ;fcb(ext)=dir(ext)=buff(ext)
3791  =
3792  =                           ;     under MP/M reccnt may have been extended
3793  =                           ;     by another process if the file was opened in
3794  =                           ;     unlocked mode - since extents match, set
3795  =                           ;     both reccnt fields to the higher value
3796  =
3797  =1363 3A077207      136E          cmp al,[bx]! jb mrg_rc1
3798  =1367 0AC07505      1370            or al,al! jne mrg_rc2
3799  =136B E851FF        12BF            call set_rc3
3800  =                           mrg_rc1:
3801  =136E 87DE                        xchg bx,si                    ;fcb(rc) = buff(rc)
3802  =                           mrg_rc2:
3803  =1370 8A048807                    mov al,[si]! mov [bx],al      ;buff(rc) = fcb(rc)
3804  =
3805  =                           if BMPM
3806  =1374 F606FB08FF                  test dont_close,true
3807  =1379 7401          137C          jz $+3
3808  =137B C3                            ret
3809  =                           endif
3810  =
3811  =137C E8A6F3        0725         call get_dptra                ;set t3' off indicating file update
3812  =137F 83C308                      add bx,11
3813  =1382 8A07                        mov al,[bx]
3814  =1384 247F                        and al,7fh
3815  =1386 8807                        mov [bx],al
3816  =1388 E8D2F3        075D          call setfwf
3817  =138B B101                        mov cl,1
3818  =138D E887F9        0D47          call scndmb                   ;set 2nd ALV
3819  =1390 E97AF8        0C0D          jmp wrdir                     ;ok to "wrdir" here -
3820  =                                                              ;1.4 compatible
3821  =                                                              ; ret
3822  =                           mergerr:
3823  =1393 E8C7F3        075D          call setfwf                   ;elements did not merge ok
3824  =1396 C7470200FFFF                mov word ptr 2[bx],0ffffh     ;flag fcb as invalid
3825  =139B E911FD        10AF          jmp lret_eq_ff                ;=255 non zero flag set
3826  =
3827  =                           close:          ;locate the directory element and re-write it
3828  =                           ;-----
3829  =139E 33C0                        xor ax,ax
3830  =13A0 A20D08                      mov lret,al
3831  =
3832  =                           if BMPM
3833  =13A3 A2FB08                      mov dont_close,al
3834  =                           endif
3835  =
```

```
3836
3837  =13A6 E86DF3      0716          call nowrite
3838  =13A9 7554        13FF          jnz ret13                   ;skip close if r/o disk
3839  =                               ;check file write flag -
3840  =                               ;0 indicates written
3841  =13AB A04A0B                    mov al,info_fcb+modnum      ;fcb(modnum) in AL
3842  =13AE 2480                      and al,fwfmsk
3843  =13B0 754D        13FF          jnz ret13                   ;return if bit remains set
3844  =                       close1:
3845  =13B2 E880F3      0765          call chk_inv_fcb
3846  =13B5 74DC        1393          jz mergerr
3847  =
3848  =                       if BMPM
3849  =13B7 C6061108FF                mov comp_fcb_cks,true
3850  =                       endif
3851  =
3852  =13BC E8D8F1      059A          call get_dir_ext            ;al = dir extent
3853  =13BF 8AC8                      mov cl,al
3854  =13C1 8A2F                      mov ch,[bx]                 ;ch = fcb(ext)
3855  =13C3 51                        push cx                     ;save fcb(ext) & dir extent
3856  =13C4 8807                      mov [bx],al
3857  =13C6 E804FF      12CD          call restore_rc
3858  =13C9 3ACD                      cmp cl,ch
3859  =13CB 7303        13D0          jae $+5
3860  =13CD E8D5FE      12A5            call set_rc
3861  =13D0 E812FF      12E5          call close_fcb
3862  =13D3 BB480B                    mov bx,offset info_fcb+extnum
3863  =13D6 59                        pop cx
3864  =13D7 8A0F                      mov cl,[bx]
3865  =13D9 882F                      mov [bx],ch                 ;reset extent
3866  =13DB E9C7FE      12A5          jmp set_rc                  ;reset record count
3867  =
3868  =                       make:
3869  =                       ;----
3870  =                       ;       create a new file by creating a directory entry
3871  =                       ;       then opening the file
3872  =
3873  =13DE 833E1508FF                cmp xdcnt,0ffffh
3874  =13E3 7403        13E8          je $+5
3875  =13E5 E898F3      0783            call set_dcnt
3876  =13E8 FF363C0B                  push word ptr info_fcb
3877  =13EC C6063C0BE5                mov info_fcb,empty          ;info_fcb = empty
3878  =13F1 B101                      mov cl,1
3879  =
3880  =13F3 E82DFC      1023          call searchi
3881  =13F6 E852FC      1048          call searchn
3882  =                               ;zero flag set if no space
3883  =13F9 8F063C0B                  pop word ptr info_fcb       ;restore info_fcb
3884  =13FD 7501        1400          jnz $+3                     ;return with error condition
3885  =13FF C3                ret13:    ret                       ;255 if not found
3886  =
3887  =1400 F6061308FF                test make_xfcb,0ffh         ;return early if making an xfcb
3888  =1405 75F8        13FF          jnz ret13
```

```
3889
3890  =
3891  =1407 BB490B                 mov bx,offset info_fcb+chksum
3892  =140A C60700                 mov b[bx],0                    ;clear s1 byte
3893  =140D 43                     inc bx
3894  =140E 8A07                   mov al,[bx]                    ;clear and save file write flag
3895  =1410 5053                   push ax! push bx
3896  =1412 80273F                 and b[bx],3fh
3897  =1415 43                     inc bx
3898  =                                                           ;clear the remainder of fcb
3899  =1416 B91100                 mov cx,fcblen-namlen           ;number of bytes to fill
3900  =1419 B001                   mov al,1                       ;clear al for fill
3901  =                        make0:
3902  =141B C60700                 mov b[bx],0
3903  =141E 43                     inc bx
3904  =141F E2FA          141B     loop make0
3905  =1421 FEC8750A      142F     dec al! jnz make1
3906  =1425 E81D08        1C45       call get_dtba
3907  =1428 0AC0                     or al,al
3908  =142A B90A00                   mov cx,10
3909  =142D 74EC          141B       jz make0
3910  =                        make1:
3911  =142F E866F3        0798     call setcdr                    ;may have extended dir
3912  =                                                           ;now copy entry to dir
3913  =1432 B100                   mov cl,0
3914  =1434 BA2000                 mov dx,fcblen                  ;dh = 0
3915  =1437 E81DFE        1257     call copy_dir0
3916  =
3917  =143A 5B58                   pop bx! pop ax                 ;restore file write flag
3918  =143C 8807                   mov [bx],al
3919  =
3920  =143E C606DA0800             mov actual_rc,0                ;actual_rc = 0
3921  =
3922  =1443 E880FB        0FD3     CALL FIX_HASH
3923  =
3924  =                                                           ;set file write flag to "1"
3925  =1446 E914F3        075D     jmp setfwf
3926  =                            ;ret
3927  =
3928  =                        openreel:
3929  =                        ;--------
3930  =                        ;       close the current extent, and open the next one
3931  =                        ;       if possible.  rmf is true if in read mode.
3932  =                        ;       lret is set to 0 if successful
3933  =
3934  =1449 A04A0B                 mov al,info_fcb+modnum
3935  =144C A2DC08                 mov save_mod,al                ;save current module #
3936  =144F BB480B                 mov bx,offset info_fcb+extnum
3937  =1452 8A07                   mov al,[bx]
3938  =1454 8AC8                   mov cl,al
3939  =1456 FEC1                   inc cl                         ;increment ext #
3940  =1458 E881F1        05DC     call compext                   ;did we cross a dir fcb boundary ?
3941  =145B 7503          1460     jnz $+5
```

```
3942
3943  =145D E97400        14D4          jmp openr3                    ;no
3944  =1460 5351                        push bx! push cx
3945  =1462 E839FF        139E          call close                    ;close current extent
3946  =                                                               ;lret remains at enddir
3947  =1465 595B                        pop cx! pop bx                ;if we cannot open next ext
3948  =1467 803E0D08FF                  cmp lret,0ffh
3949  =146C 7501          146F          jne $+3
3950  =146E C3                           ret
3951  =146F 801F                        mov al,maxext
3952  =1471 22C1                        and al,cl
3953  =1473 8807                        mov [bx],al                   ;update fcb extent field
3954  =1475 7508          1482          jnz openr0                    ;branch if in same module
3955  =
3956  =                           ;               extent number overflow, go to next module
3957  =
3958  =1477 83C302                      add bx,(modnum-extnum)        ;bx=.fcb(modnum)
3959  =147A FE07                        inc b[bx]                     ;fcb(modnum)=++1
3960  =                                                               ;module number incremented,
3961  =                                                               ;check for overflow
3962  =147C 8A07                        mov al,[bx]
3963  =147E 243F                        and al,maxmod                 ;mask high order bits
3964  =1480 7439          1488          jz openerr                    ;cannot overflow to 0
3965  =                                                               ;otherwise, ok to continue
3966  =                                                               ;with new module
3967  =                           openr0:
3968  =1482 C7061508FFFF                mov xdcnt,0ffffh              ;reset xdcnt for make
3969  =
3970  =                           if BMPM
3971  =1488 C6060A09FF                  mov byte ptr sdcnt+1,0ffh
3972  =                           endif
3973  =
3974  =148D E8AFFB        103F          call search_name              ;next extent found?
3975  =1490 7516          14A8          jnz openr1
3976  =1492 A0DD09                       mov al,rmf                    ;end of file encountered
3977  =1495 FEC0                         inc al                        ;0ffh becomes 00 if read
3978  =1497 7422          14BB          jz openerr                    ;sets lret = 1
3979  =1499 E842FF        13DE          call make                     ;try to extend current file
3980  =                                                               ;cannot be end of directory
3981  =149C 7410          14BB          jz openerr                    ;with lret = 1
3982  =
3983  =                           if BMPM
3984  =149E E8960D        2237            call fix_olist_item
3985  =14A1 C6061108FF                    mov comp_fcb_cks,true
3986  =                           endif
3987  =
3988  =14A6 EB08          14B0          jmps openr2
3989  =                           openr1:
3990  =
3991  =                           if BMPM
3992  =14A8 C6061108FF                  mov comp_fcb_cks,true         ;set fcb checksum flag
3993  =                           endif                               ;not end of file, open
3994  =
```

```
3995
3996  =14AD E8D2FD       1282          call opencopy
3997  =
3998  =                                if BCPM
3999  =                                      call set_lsn
4000  =                                endif
4001  =
4002  =                                openr2:
4003  =14B0 E83AF1       05ED          call getfcb                    ;set parameters
4004  =14B3 32C0                       xor al,al
4005  =14B5 A21509                     mov vrecord,al
4006  =14B8 E9F3EF       04AE          jmp set_lret                   ;lret = 0
4007  =                                ;ret
4008  =
4009  =                                openerr:
4010  =                                ;-------
4011  =14BB BB480B                     mov bx,offset info_fcb+extnum
4012  =14BE A0DC08                     mov al,save_mod
4013  =14C1 884702                     mov 2[bx],al
4014  =14C4 8A07                       mov al,[bx]
4015  =14C6 FEC8                       dec al
4016  =14C8 241F                       and al,1fh
4017  =14CA 8807                       mov [bx],al
4018  =14CC C6061108FF                 mov comp_fcb_cks,0ffh
4019  =                                                               ;cannot move to next extent
4020  =                                                               ;of this file
4021  =14D1 E9D8EF       04AC          jmp set_lret1                  ;lret = 1
4022  =                                ;ret
4023  =                                openr3:
4024  =14D4 880F                       mov [bx],cl                    ;increment extent field
4025  =14D6 E8C1F0       059A          call get_dir_ext
4026  =14D9 8AC8                       mov cl,al
4027  =14DB F6069E0880                 test high_ext,80h              ;is open mode unlocked?
4028  =14E0 7512         14F4          jnz openr4                     ;yes
4029  =14E2 3A07                       cmp al,[bx]                    ;is dir ext < fcb(ext)?
4030  =14E4 730E         14F4          jae openr4                     ;no
4031  =14E6 FE0F                         dec b[bx]                    ;decrement extent
4032  =14E8 803E0D09FF                   cmp rmf,0ffh                 ;is this a read operation?
4033  =14ED 7503         14F2          jne $+5                        ;no
4034  =14EF E9BAEF       04AC            jmp set_lret1                ;yes - reading unwritten data
4035  =14F2 FE07                         inc b[bx]                    ;restore extent
4036  =                                openr4:
4037  =14F4 E8D6FD       12CD          call restore_rc
4038  =14F7 E8A8FD       12A5          call set_rc
4039  =14FA EBB4         1480          jmps openr2
4040  =
4041  =                                ;********** end bdos file system part 2 **********
4042
```

```
4043
4044  =                         eject | include file3.bdo       ; file system part 3
4045  =
4046  =                   ;************ bdos file system part 3 ************
4047  =
4048  =                   rseek:          ;random access seek operation
4049  =                   ;-----
4050  =                   ;       fcb is assumed to address an active fcb
4051  =                   ;       (modnum has been set to 1100$0000b if previous bad seek)
4052  =                   ;       entry:  CL = true if read operation
4053  =                   ;                  = false if write operation
4054  =                   ;       exit:   Z flag set if successful
4055  =
4056  =14FC 51                    push cx                         ;save r/w flag (indexed on stack)
4057  =14FD A05D0B                mov al,info_fcb+ranrec          ;AL = fcb(ranrec(0))
4058  =1500 8AD0                  mov dl,al
4059  =1502 80E27F                and dl,7fh                      ;record number
4060  =1505 D0D0                  rcl al,1                        ;cy=lsb of extent#
4061  =1507 A05E0B                mov al,info_fcb+ranrec+1        ;AL = fcb(ranrec(1))
4062  =150A 8AE8                  mov ch,al
4063  =150C D0D5                  rcl ch,1
4064  =150E 80E51F                and ch,11111b                   ;CH = extent #
4065  =                                                           ;DL = record #
4066  =1511 24F0                  and al,11110000b
4067  =1513 0A065F0B              or al,info_fcb+ranrec+2
4068  =1517 B104                  mov cl,4
4069  =1519 D2C0                  rol al,cl
4070  =151B 8ACD                  mov cl,ch
4071  =151D 8AE8                  mov ch,al                       ;CH = mod#, CL = ext#
4072  =
4073  =151F 803E5F0B03                cmp byte ptr info_fcb+ranrec+2,3  ;r(3) field must
4074  =                                                           ;be <= 3
4075  =1524 B306                  mov bl,6                        ;zero flag, BL=6
4076  =                                                           ;produce error 6,
4077  =1526 7603          1528        jbe $+5                     ;seek past physical eod
4078  =1528 E9B500        15E1          jmp seekerr               ;random record out of range
4079  =                                                           ;otherwise, high byte = 0
4080  =152B 88165C0B              mov info_fcb+nxtrec,dl          ;record number
4081  =
4082  =                   ;       arrive here with CH=mod#, CL=ext#, SI=.fcb, rec stored
4083  =                   ;       the r/w flag is still stacked.   compare fcb values
4084  =
4085  =152F 803E7A0825            cmp fx_intrn,fxi99
4086  =1534 7456          158C    je rseek3
4087  =1536 E82CF2        0765    call chk_inv_fcb
4088  =1539 7451          158C    jz rseek3
4089  =                                                           ;check module # first
4090  =153B 8AC5                    mov al,ch                   ;b=seek mod#
4091  =                                                           ;could be overflow at eof,
4092  =                                                           ;producing module# of 90h
4093  =                                                           ;or 10h, so compare all
4094  =                                                           ;but fwf
4095  =153D 2A064A0B                sub al,info_fcb+modnum
```

```
4096
4097  =1541 243F                    and al,3fh                ;same modnum?
4098  =1543 7539          157E      jnz ranclose              ; no
4099  =1545 A0480B                    mov al,info_fcb+extnum
4100  =1548 3AC1                      cmp al,cl
4101  =154A 7503          154F        jnz $+5
4102  =154C E98400        1503          jmp seekok1           ;extents equal
4103  =154F E88AF0        05DC        call compext
4104  =1552 752A          157E        jnz ranclose
4105  =1554 51                        push cx
4106  =1555 E842F0        059A        call get_dir_ext
4107  =1558 59                        pop cx
4108  =1559 3AC1                      cmp al,cl               ;is dir_ext < new extent?
4109  =155B 7313          1570        jnb rseek2              ;no
4110  =155D F6069E0880                test high_ext,80h       ;is open mode unlocked?
4111  =1562 750C          1570        jnz rseek2              ;yes
4112  =1564 5A52                        pop dx! push dx       ;dl = read/write flag
4113  =1566 FEC2                      inc dl                  ;is this a write fx?
4114  =1568 7505          1570        jnz rseek2              ;yes
4115  =156A FEC2                        inc dl                ;reset z flag
4116  =156C 5A                          pop dx                ;discard read/write flag
4117  =156D E93CEF        04AC          jmp set_lret1         ;lret=1, reading unwritten data
4118  =                   rseek2:
4119  =1570 880E480B                    mov info_fcb+extnum,cl  ;fcb(ex) = new extent
4120  =1574 8AC8                        mov cl,al
4121  =1576 E854FD        12CD          call restore_rc
4122  =1579 E829FD        12A5          call set_rc
4123  =157C EB55          15D3          jmps seekok1
4124  =                   ranclose:
4125  =157E 51                      push cx                   ;save seek ext# and mod#
4126  =157F E81CFE        139E      call close                ;current extent closed
4127  =1582 59                      pop cx                    ;recall ext# and mod#
4128  =1583 B303                    mov bl,3                  ;cannot close error #3
4129  =1585 A00D08                  mov al,lret
4130  =1588 FEC0                    inc al
4131  =158A 7455          15E1      jz seekerr
4132  =                   rseek3:
4133  =158C C7061508FFFF            mov xdcnt,0ffffh          ;reset xdcnt for make
4134  =
4135  =                   if BMPM
4136  =1592 C6060A09FF              mov byte ptr sdcnt+1,0ffh
4137  =                   endif
4138  =
4139  =1597 860E480B                xchg info_fcb+extnum,cl   ;fcb(extnum)=ext#
4140  =159B A04A0B                  mov al,info_fcb+modnum
4141  =159E 86E8                    xchg ch,al
4142  =15A0 51                      push cx                   ;save CL=extent and CH=module
4143  =15A1 80E540                  and ch,40h                ;copy file write flag
4144  =15A4 0AC5                    or al,ch
4145  =15A6 A24A0B                  mov info_fcb+modnum,al    ;fcb(modnum)=mod#
4146  =15A9 E8D1FC        127D      call open                 ;is the file present?
4147  =15AC A00D08                  mov al,lret
4148  =15AF FEC0                    inc al
```

```
4149
4150  =15B1 751A          15CD          jnz seekok                    ;open successful?
4151  =                                                               ;cannot open file, read mode?
4152  =15B3 88EC                          mov bp,sp                   ;CL = r/w flag (=0ffh if read)
4153  =15B5 884E02                        mov cx,2[bp]                ;everyone expects this
4154  =                                                               ;item stacked
4155  =15B8 B304                          mov bl,4                    ;seek to unwritten extent #4
4156  =15BA FEC1                          inc cl                      ;becomes 00 if read operation
4157  =15BC 7418          15D9            jz badseek                  ;error if read operation
4158  =                                                               ;write, make new extent
4159  =15BE E81DFE        13DE            call make
4160  =15C1 B305                          mov bl,5                    ;cannot create new extent #5
4161  =15C3 A00D08                        mov al,lret
4162  =15C6 FEC0                          inc al
4163  =15C8 740F          15D9            jz badseek                  ;no dir space
4164  =                                                               ;file make successful
4165  =
4166  =                         if BMPM
4167  =15CA E86A0C        2237            call fix_olist_item
4168  =                         endif
4169  =
4170  =                         seekok:
4171  =15CD 59                            pop cx                      ;discard ext and mod
4172  =
4173  =                         if BMPM
4174  =15CE C6061108FF                    mov comp_fcb_cks,true
4175  =                         endif
4176  =                         if BCPM
4177  =                                   call set_lsn
4178  =                         endif
4179  =
4180  =                         seekok1:
4181  =15D3 59                            pop cx                      ;discard r/w flag
4182  =15D4 32C0                          xor al,al
4183  =15D6 E9D5EE        04AE            jmp set_lret                ;with Z flag set
4184  =
4185  =                         badseek:        ;restore fcb(ext) and fcb(mod)
4186  =15D9 58                            pop ax                      ;AL = extent, AH = module
4187  =15DA A2480B                        mov info_fcb+extnum,al
4188  =15DD 88264A0B                      mov info_fcb+modnum,ah
4189  =                         seekerr:
4190  =15E1 59                            pop cx                      ;discard r/w flag
4191  =15E2 881E0D08                      mov lret,bl                 ;lret=#
4192  =15E6 0ADB                          or bl,bl                    ;Z flag reset
4193  =15E8 C3                            ret
4194  =
4195  =                         if BMPM
4196  =
4197  =                         test_disk_fcb:
4198  =                         ;-------------
4199  =                         ;       exit:   Z flag reset if file opened in unlock mode and
4200  =                         ;                 access of dir fcb was successful
4201  =
```

```
4202
4203  =15E9 F6069E0880              test high_ext,80h               ;was file opened in unlocked mode
4204  =15EE 7428          1618      jz ret13a                       ;no
4205  =15F0 F606FB08FF              test dont_close,0ffh            ;return if dont_close false
4206  =15F5 7421          1613      jz ret13a
4207  =15F7 E888FD        1382      call close1                     ;fcb in memory
4208  =                       test_disk_fcb1:
4209  =15FA BB0D08                  mov bx,offset lret
4210  =15FD 803FFF                  cmp b[bx],0ffh
4211  =1600 7505          1608      jne not_err
4212  =1602 5E                        pop si                        ;discard return address
4213  =1603 B00B                      mov al,11                     ;unallocated block verify error
4214  =1605 E9A6EE        04AE        jmp set_lret                  ;error - close operation unsuccessful
4215  =                       not_err:
4216  =1608 C60700                  mov b[bx],0                     ;lret = 0
4217  =160B A04B08                  mov al,info_fcb+reccnt
4218  =160E A21309                  mov rcount,al
4219  =1611 32C0                    xor al,al
4220  =1613 A2FB08                  mov dont_close,al               ;dont_close = false
4221  =1616 FEC0                    inc al                          ;successful operation, reset zero flag
4222  =1618 C3                ret13a: ret
4223  =
4224  =                       endif
4225  =
4226  =                       CHECK_NPRS:
4227  =                       ;----------
4228  =                       ;       DIR_CNT CONTAINS THE NUMBER OF 128 BYTE RECORDS
4229  =                       ;       TO TRANSFER DIRECTLY.  THIS ROUTINE SET DIR_CNT
4230  =                       ;       WHEN INITIATING A SEQUENCE OF DIRECT PHYSICAL I/O
4231  =                       ;       OPERATIONS.  DIR_CNT IS DECREMENTED EACH TIME CHECK_NPRS
4232  =                       ;       IS CALLED DURING SUCH A SEQUENCE.
4233  =                       ;       exit:   ~C FLG & ~Z FLG - DIRECT PHYSICAL I/O OPERATION
4234  =                       ;               ~C FLG &  Z FLG - INDIRECT(DEBLOCK) I/O OPERATION
4235  =                       ;                C FLG          - NO PHYSICAL I/O OPERATION
4236  =
4237  =1619 8A2E2208                MOV CH,blk_off                  ;CH = VRECORD & BLKMSK
4238  =161D A01708                  MOV AL,DIR_CNT
4239  =1620 3C02                    CMP AL,2                        ;IS DIR_CNT > 1?
4240  =1622 7207          162B      JC CHECK_NPR1                   ;NO
4241  =1624 FEC8                      DEC AL                        ;DIR_CNT = DIR_CNT - 1
4242  =1626 A21708                    MOV DIR_CNT,AL                ;we are in mid-multi sector i/o
4243  =1629 F9                        STC
4244  =162A C3                        ret                           ;RETURN WITH C FLAG SET
4245  =                       CHECK_NPR1:
4246  =162B A0C808                  MOV AL,PHYMSK                   ;CL = PHYMSK
4247  =162E 8AC8                    MOV CL,AL
4248  =1630 22C5                    AND AL,CH                       ;AL = VRECORD & PHYMSK
4249  =                                                             ;ARE WE IN MID-PHYSICAL RECORD?
4250  =1632 740A          163E      JZ CHECK_NPR11                  ;NO
4251  =                       CHECK_NPR1X:
4252  =1634 0AC9                      OR CL,CL                      ;IS PHYMSK = 0?
4253  =1636 7403          163B        JZ CHECK_NPR1Y              ;YES
4254  =1638 32C0                        XOR AL,AL                   ;RETURN WITH Z FLAG SET & C FLAG RESET
```

```
4255
4256  =163A C3                          RET
4257  =                           CHECK_NPR1Y:
4258  =163B 0C01                        OR AL,1                 ;RESET C & Z FLAGS
4259  =163D C3                          RET
4260  =                           CHECK_NPR11:
4261  =163E 8AF1                        MOV DH,CL
4262  =1640 F6D6                        not DH                  ;DH = ~ PHYMSK
4263  =1642 A01808                      MOV AL,MULT_NUM
4264  =1645 3C02                        CMP AL,2                ;IS MULT_NUM < 2?
4265  =1647 72EB          1634          JC CHECK_NPR1X          ;YES
4266  =1649 F6069E0880                  TEST HIGH_EXT,80H       ;WAS FILE OPENED IN UNLOCKED MODE?
4267  =164E 75E4          1634          JNZ CHECK_NPR1X         ;YES
4268  =1650 BB1509                      MOV BX,OFFSET VRECORD
4269  =1653 8A27                        MOV AH,[BX]             ;AH = VRECORD
4270  =1655 02C4                        ADD AL,ah
4271  =1657 3C80                        CMP AL,80H
4272  =1659 7202          165D          JC CHECK_NPR2
4273  =165B B080                        MOV AL,80H
4274  =                           CHECK_NPR2:                   ;AL = MIN(VRECORD + MULT_NUM,80H) = X
4275  =165D 51                          PUSH CX                 ;SAVE VRECORD & BLKMSK, PHYMSK
4276  =165E C6077F                      MOV B[BX],7FH           ;VRECORD = 7F
4277  =1661 5350                        PUSH BX! PUSH AX
4278  =1663 8AD8                        MOV BL,AL               ;BL = X
4279  =1665 A0BB08                      MOV AL,BLKMSK
4280  =1668 8AD0                        MOV DL,AL
4281  =166A FEC2                        INC DL                  ;DL = BLKMSK + 1
4282  =166C F6D0                        not AL
4283  =166E 22E0                        AND AH,AL               ;AH = VRECORD & ~BLKMSK
4284  =1670 F6060D09FF                  TEST RMF,0FFH           ;READ FUNCTION?
4285  =1675 7409          1680          JZ CHECK_NPR21          ;NO
4286  =1677 A01309                      MOV AL,RCOUNT
4287  =167A 22C6                        AND AL,DH               ;AL = RCOUNT & ~PHYMSK
4288  =167C 3AC3                        CMP AL,BL               ;IS AL < X?
4289  =167E 7202          1682          JC CHECK_NPR23          ;YES
4290  =                           CHECK_NPR21:
4291  =1680 8AC3                        MOV AL,BL               ;AL = MIN(VRECORD + MULT_NUM,80H) = X
4292  =                           CHECK_NPR23:
4293  =1682 2AC4                        SUB AL,AH               ;AL = AL - VRECORD & ~BLKMSK
4294  =1684 3AC2                        CMP AL,DL               ;IS AL < BLKMSK + 1?
4295  =1686 7268          16F0          JC CHECK_NPR9           ;YES
4296  =1688 50                          PUSH AX                 ;AL = MAX # OF RECORDS
4297  =1689 E8B2EE        053E          CALL DMPOSITION         ;COMPUTE MAXIMUM DISK POSITION
4298  =168C 8AE8                        MOV CH,AL               ;CH = MAX DISK POS
4299  =168E A01109                      MOV AL,DMINX            ;AL = CURRENT DISK POS
4300  =1691 3AC5                        CMP AL,CH               ;IS CURR POS = MAX POS?
4301  =1693 8AD0                        MOV DL,AL
4302  =1695 743C          16D3          JZ CHECK_NPR5           ;YES
4303  =1697 8AC8                        MOV CL,AL
4304  =1699 51                          PUSH CX                 ;CL = CURR POS, CH = MAX POS
4305  =169A B500                        MOV CH,0
4306  =169C E8B6EE        0555          CALL GETDM              ;BX = BLOCK NUMBER (CURR POS)
4307  =                           CHECK_NPR4:
```

```
4308
4309  =169F 53                        PUSH BX
4310  =16A0 41                        INC CX                  ;CURR POS = CURR POS + 1
4311  =16A1 E8B1EE      0555          CALL GETDM              ;GET NEXT BLOCK(CURR POS)
4312  =16A4 5A                        POP DX
4313  =16A5 42                        INC DX
4314  =16A6 3BDA                      CMP BX,DX               ;DOES CURR BLK = LAST BLK + 1?
4315  =16A8 74F5        169F          JZ CHECK_NPR4           ;YES
4316  =16AA F6060D09FF                test rmf,0ffh           ;write function?
4317  =16AF 7517        16C8          jnz check_npr45         ; no
4318  =16B1 0BDB7513    16C8          or bx,bx! jnz check_npr45  ;does curr blk = 0?
4319  =16B5 3B160D08                  cmp dx,maxall           ;is last blk + 1 >= max alloc blk
4320  =16B9 730D        16C8          jae check_npr45         ; yes
4321  =16BB 5152                      push cx! push dx
4322  =16BD 89CA                      mov cx,dx
4323  =16BF E8F0F5      0CB2          call getallocbit
4324  =16C2 5B59                      pop bx! pop cx
4325  =16C4 D0E873D7    169F          shr al,1! jnc check_npr4  ;if blk not allocated it will be
4326  =                       check_npr45:
4327  =16C8 FEC9                      DEC CL                  ;CL = INDEX OF LAST SEQ BLK
4328  =16CA 5A                        POP DX
4329  =16CB 8AC6                      MOV AL,DH               ;AL = MAX POS
4330  =16CD 3AC1                      CMP AL,CL               ;IS AL < LAST SEQ BLK POS
4331  =16CF 7202        16D3          JC CHECK_NPR5           ;YES
4332  =16D1 8AC1                      MOV AL,CL
4333  =                       CHECK_NPR5:                     ;AL = LAST BLK POS
4334  =16D3 2AC2                    SUB AL,DL               ;AL = AL - STARTING POS
4335  =16D5 8AE8                    MOV CH,AL
4336  =16D7 FEC5                    INC CH                  ;CH = # OF CONSECUTIVE BLOCKS
4337  =16D9 A0BB08                  MOV AL,BLKMSK
4338  =16DC FEC0                    INC AL                  ;AL = BLKMSK + 1
4339  =16DE F6E5                    MUL CH                  ;AL = # OF CONSECUTIVE LOGICAL RECS
4340  =16E0 59                      POP CX                  ;CL = MAXIMUM # OF RECORDS
4341  =16E1 86C1                    XCHG AL,CL
4342  =16E3 F6060D09FF              TEST RMF,0FFH           ;READ FUNCTION?
4343  =16E8 7404        16EE        JZ CHECK_NPR8           ;NO
4344  =16EA 3AC1                      CMP AL,CL             ;IS MAX < # OF CONS. RECS?
4345  =16EC 7202        16F0          JC CHECK_NPR9         ;YES
4346  =                       CHECK_NPR8:
4347  =16EE 8AC1                    MOV AL,CL
4348  =                       CHECK_NPR9:                     ;AL = # OF CONSECUTIVE RECS
4349  =16F0 59                    POP CX
4350  =16F1 5B                    POP BX
4351  =16F2 882F                  MOV [BX],CH               ;RESTORE VRECORD
4352  =16F4 59                    POP CX
4353  =16F5 8A361808              MOV DH,MULT_NUM
4354  =16F9 2AC5                  SUB AL,CH                 ;AL = AL - VRECORD & blkmsk
4355  =16FB 3AC6                  CMP AL,DH                 ;IS AL < MULT_NUM
4356  =16FD 7202        1701      JC CHECK_NPR10            ;YES
4357  =16FF 8AC6                    MOV AL,DH
4358  =                       CHECK_NPR10:                    ;AL = # OF CONSECUTIVE RECS
4359  =1701 F6D1                  not CL
4360  =1703 22C1                  AND AL,CL                 ;IF DIR_CNT = 0 THEN
```

```
4361
4362  =1705 741A        1721          JZ RET13B                       ;RETURN WITH Z FLAG SET, C FLAG RESET
4363  =1707 A21708                    MOV DIR_CNT,AL                  ;DIR_CNT = AL & ~PHYMSK
4364  =170A F6060D09FF                TEST RMF,0FFH                   ;READ FUNCTION?
4365  =170F 7405        1716          JZ CHECK_NPR10X                 ;NO
4366  =1711 50                        PUSH AX
4367  =1712 E82914      283E          CALL flushx                     ;FLUSH BLOCKING/DEBLOCKING
4368  =1715 58                        POP AX                          ;BUFFERS FOR DRIVE
4369  =                         CHECK_NPR10X:
4370  =1716 8A0EC708                  MOV CL,PHYSHF
4371  =171A D2E8                      SHR AL,CL                       ;AL = # OF CONSECUTIVE PHYSICAL RECS
4372  =171C A22008                    mov mult_sec,al                 ;save multisector count for r/w
4373  =171F 0C01                      OR AL,1                         ;RETURN WITH C AND Z FLAGS RESET
4374  =1721 C3                  RET13B: RET
4375  =
4376  =                         diskread:       ;enter here for sequential and random disk reads
4377  =                         ;--------
4378  =1722 E845F0      076A          call tst_inv_fcb                ;verify fcb has not been flagged
4379  =1725 B0FF                      mov al,true                     ;as invalid
4380  =1727 A20D09                    mov rmf,al                      ;read mode flag =
4381  =                                                               ;true (open$reel)
4382  =                                                               ;read the next record from
4383  =                                                               ;the current fcb
4384  =
4385  =                         if BMPM
4386  =172A A2F808                    mov dont_close,al
4387  =                         endif
4388  =
4389  =172D E88DEE      05E9          call getfcb                     ;sets parameters for the read
4390  =                         disk_read0:
4391  =1730 A01509                    mov al,vrecord
4392  =1733 3A061309                  cmp al,rcount                   ;vrecord-rcount
4393  =                                                               ;skip if rcount > vrecord
4394  =1737 7216        174F          jb recordok
4395  =
4396  =                         if BMPM
4397  =                         ;       if fcb was opened in unlocked mode, check the directory fcb
4398  =                         ;       to make sure another process has not allocated a block to
4399  =                         ;       the fcb since this process opened the fcb
4400  =
4401  =1739 E8ADFE      15E9            call test_disk_fcb              ;jump back to diskread0 if dont_close
4402  =173C 75F2        1730            jnz diskread0                   ;true, file opened in unlocked mode,
4403  =                                                                 ;and access of dir fcb successful
4404  =                                                                 ;dont_close set to false
4405  =173E A01509                      mov al,vrecord
4406  =                         endif
4407  =                                                               ;not enough records in extent
4408  =                                                               ;record count must be 128
4409  =                                                               ;to continue
4410  =1741 3C80                        cmp al,128                    ;vrecord = 128?
4411  =1743 7532        1777            jnz diskeof                   ;skip if vrecord<>128
4412  =1745 E801F0      1449            call openreel                 ;go to next extent if so
4413  =                                                               ;now check for open ok
```

```
4414
4415  =1748 803E0D0800          1777          cmp lret,0                ;stop at eof
4416  =174D 7528                              jnz diskeof
4417  =                                 recordok:                       ;fcb addresses a record to read
4418  =174F E819EE              056B          call index                ;error 2 if reading
4419  =                                                                 ;unwritten data
4420  =                                                                 ;(returns 1 to be compatible
4421  =                                                                 ;with 1.4)
4422  =                                                                 ;arecord=0000?
4423  =                                 if BMPM
4424  =1752 7507                1758          jnz recordok1
4425  =1754 E892FE              15E9            call test_disk_fcb      ;jump back to diskread0 if dont_close
4426  =1757 7507                1730            jnz diskread0           ;true, file opened in unlocked mode,
4427  =                                                                 ;and access of dir fcb successful
4428  =                                 endif
4429  =1759 741C                1777          jz diskeof
4430  =                                 recordok1:                      ;record has been allocated, read it
4431  =
4432  =                                 if BMPM
4433  =175B E8EF08              204D          call test_lock            ;check if record locked by another
4434  =                                 endif
4435  =175E E862ED              0513          call atran                ;arecord now a disk address
4436  =1761 E885FE              1619          CALL CHECK_NPRS
4437  =1764 720E                1774          JC RECORDOK2
4438  =1766 7503                176B            JNZ $+5
4439  =1768 E9F3F2              0A5E              JMP READ_DEBLOCK
4440  =176B E8ACF4              0C1A            CALL SETDATA
4441  =176E E883ED              04F4            call seek               ;to proper track,sector
4442  =1771 E87DF0              07F1            call rdbuff             ;to dma address
4443  =                                 RECORDOK2:
4444  =1774 E9A2EE              0619          jmp setfcb                ;replace parameter
4445  =
4446  =                                 diskeof:
4447  =1777 E932ED              04AC          jmp set_lret1             ;lret = 1
4448  =                                 ;ret
4449  =
4450  =                                 diskwrite:      ;enter here for sequential and random disk writes
4451  =                                 ;---------
4452  =177A C6060D0900                  mov rmf,false                   ;read mode flag
4453  =                                                                 ;write record to currently
4454  =                                                                 ;selected file
4455  =177F E8C1EF              0743          call checkwrite           ;in case write protected
4456  =
4457  =1782 A04A0B                     mov al,info_fcb+modnum
4458  =1785 D0D0F6D0                    rcl al,11 not al
4459  =1789 84069F08                    test xfcb_read_only,al          ;read/only error if xfcb_read_only true
4460  =178D B403                        mov ah,3                        ;xfcb_read_only true if file is
4461  =178F 7403                1794    jz $+5                          ;password protected in write mode
4462  =                                                                 ;and password was not supplied
4463  =                                                                 ;when the file was opened.
4464  =                                 jmp_set_aret:
4465  =1791 E920ED              0484          jmp set_aret
4466  =
```

```
4467
4468  =1794 F6069E0840              test high_ext,40h           ;read/only error if fcb opened
4469  =1799 75F6          1791      jnz jmp_set_aret            ;in read/only mode
4470  =
4471  =179B BB3C08                  mov bx,offset info_fcb      ;bx = .fcb(0)
4472  =179E E89AEF        0738      call ckrofile               ;may be a read-only file
4473  =
4474  =17A1 E8C6EF        0769      call tst_inv_fcb            ;test for valid fcb
4475  =17A4 E81D05        1CC4      call update_stamp
4476  =17A7 E843EE        05ED      call getfcb                 ;to set local parameters
4477  =17AA A01509                  mov al,vrecord
4478  =17AD 3C80                    cmp al,lstrec+1             ;vrecord-128
4479  =                                                         ;skip if vrecord > lstrec
4480  =                                                         ;vrecord = 128, cannot open
4481  =                                                         ;next extent
4482  =17AF 7208          17BC      jb dskwr0
4483  =17B1 E895FC        1449      call openreel               ;try to open next extent
4484  =17B4 F6060D08FF              test lret,true
4485  =17B9 7401          17BC      jz dskwr0
4486  =17BB C3                        ret
4487  =                       dskwr0:
4488  =
4489  =                       if BMPM
4490  =17BC C606FB08FF              mov dont_close,true
4491  =                                                         ;if file is opened unlocked,
4492  =17C1 E88908        204D      call test_lock              ; verify record is not locked
4493  =                                                         ; by another process
4494  =                       disk_write1:
4495  =
4496  =                       endif
4497  =17C4 E8A4ED        0568      call index
4498  =17C7 7404          17CD      JZ DISK_WRITE2
4499  =17C9 B100                      mov cl,0                  ;marked as normal write
4500  =                                                         ;operation for wrbuff
4501  =17CB EB51          181E        jmps dskwr1
4502  =                       disk_write2:
4503  =
4504  =                       if BMPM
4505  =17CD E819FE        15E9      call test_disk_fcb          ;jump back to disk_write1 if
4506  =                                                         ;dont_close is true, file was opened
4507  =                                                         ;in unlocked mode, and access of
4508  =                                                         ;dir fcb was successful
4509  =
4510  =17D0 75F2          17C4      jnz disk_write1
4511  =                       endif
4512  =
4513  =                       ;       not allocated:
4514  =                       ;       the argument to getblock is the starting
4515  =                       ;       position for the disk search, and should be
4516  =                       ;       the last allocated block for this file, or
4517  =                       ;       the value 0 if no space has been allocated
4518  =
4519  =17D2 E869ED        053E      call dmposition
```

```
4520
4521  =17D5 A21109                    mov dminx,al                    ;save for later
4522  =17D8 33C9                      xor cx,cx                       ;may use block zero
4523  =17DA 0AC0                      or al,al
4524  =17DC 7408          17E6        jz nopblock                     ;skip if no previous block
4525  =                               ;previous block exists at a
4526  =17DE 8AC8                        mov cl,al
4527  =17E0 49                          dec cx
4528  =17E1 E871ED        0555          call getdm                    ;previous block # to bx
4529  =17E4 8BCB                        mov cx,bx                     ;cx=prev block#
4530  =                         nopblock:
4531  =                                                               ;cx = 0, or previous block #
4532  =17E6 E826FA        120F        call getblock                   ;block # to bx
4533  =                                                               ;arrive with block# or zero
4534  =17E9 0BDB                      or bx,bx
4535  =17EB 7505          17F2        jnz blockok
4536  =                                                               ;cannot find block to allocate
4537  =17ED B002                        mov al,2
4538  =17EF E98CEC        04AE          jmp set_lret                  ;lret=2
4539  =                         blockok:
4540  =
4541  =                         if BMPM
4542  =17F2 C6051108FF                mov comp_fcb_cks,true           ;must recompute fcb checksum
4543  =                         endif
4544  =
4545  =17F7 891E1709                  mov arecord,bx                  ;bx contains allocated block #
4546  =
4547  =17FB 8BD3                      mov dx,bx                       ;place unallocated block in
4548  =17FD E8CCF1        09CC        call ua_setup                   ; list for pre read check
4549  =
4550  =                                                               ;block number in dx preserved
4551  =1800 BB4C0B                    mov bx,offset info_fcb+dskmap   ;bx=.fcb(diskmap)
4552  =1803 803E120900                cmp single,0                    ;set flags for single byte dm
4553  =1808 A01109                    mov al,dminx                    ;recall dm index
4554  =180B B400                      mov ah,0
4555  =180D 7406          1815        jz allocwd                      ;skip if allocating word
4556  =                               ;allocating a byte value
4557  =180F 03D8                        add bx,ax
4558  =1811 8817                        mov [bx],dl                   ;single byte alloc
4559  =1813 EB07          181C          jmps dskwru                   ;to continue
4560  =                         allocwd:
4561  =1815 03D8                      add bx,ax                       ;allocate a word value
4562  =1817 03D8                      add bx,ax                       ;bx=.fcb(dminx*2)
4563  =1819 8917                      mov [bx],dx                     ;double wd
4564  =181B 43                        inc bx
4565  =                         dskwru:                 ;disk write to previously unallocated block
4566  =181C B102                      mov cl,2                        ;marked as unallocated write
4567  =                         dskwr1:                 ;continue the write operation of no allocation error
4568  =                                                 ;c = 0 if normal write, 2 if to prev unalloc block
4569  =181E 880E0E09                  mov wflag,cl                    ;save the write flag
4570  =1822 E8EEEC        0513        call atran                      ;arecord set
4571  =                         diskwr10:
4572  =1825 803E7A081B                cmp fx_intrn,fxi40              ;check random with 0 fill
```

```
4573
4574  =182A 7570          189C        jnz dskw11
4575  =
4576  =                         ;     this is random write with zero fill
4577  =
4578  =182C 803E0E0902               cmp wflag,2                 ;check for new allocation
4579  =1831 7569          189C        jnz dskw11                  ;old allocation
4580  =
4581  =                         ;     new alloction for random with zero fill
4582  =
4583  =1833 C6060E0900               MOV WFLAG,0                 ;RESET WRITE FLAG
4584  =1838 FF361709                 PUSH arecord                ;SAVE ARECORD
4585  =183C A0C808                   MOV AL,PHYMSK
4586  =183F FEC0                     INC AL
4587  =1841 32E4                     XOR AH,AH
4588  =1843 50                       PUSH AX                     ;AX = PHYMSK + 1
4589  =1844 86C4                     xchg al,ah
4590  =1846 D1E8                     shr ax,1                    ;times 128
4591  =1848 8BC8                     mov cx,ax                   ;CX = # OF BYTES IN PHY RECORD
4592  =
4593  =184A 8B3EB208                 MOV di,DIR_BCBA
4594  =184E 83EF0C                   sub di,12
4595  =                         FILL00:
4596  =1851 8B7D0C                   mov di,12[di]
4597  =1854 837D0C00                 cmp word ptr 12[di],0       ;check if last bcb
4598  =1858 75F7          1851       JNZ FILL00                  ;BX = OLDEST DIRECTORY BCB
4599  =
4600  =185A C605FF                   MOV b[di],0ffh              ;BCB(0) = OFFH, DISCARD BCB
4601  =185D 8B7D0A                   mov di,10[di]
4602  =1860 893E2009                 mov cur_dma,di
4603  =1864 8C1E1E09                 mov cur_dma_seg,ds
4604  =1868 32C0                     XOR AL,AL                         ;ZERO PHYSICAL RECORD BUFFER
4605  =186A F3AA                     REP STOS AL
4606  =
4607  =186C 8A0EBB08                 mov cl,blkmsk               ;set high water mark for
4608  =1870 E895F1        0A08       call ua_preread             ; pre read check
4609  =1873 A11A09                   MOV ax,ARECORD1
4610  =1876 B102                     MOV CL,2                    ;SET WRITE FLAG
4611  =                         FILL2:
4612  =1878 A31709                   MOV ARECORD,ax
4613  =187B 51                       PUSH CX
4614  =187C E8D4EF        0853       call discard_data
4615  =187F E872EC        04F4       CALL SEEK
4616  =1882 59                       POP CX
4617  =1883 E862EF        07E8       CALL WRBUFF
4618  =1886 A11709                   MOV ax,ARECORD              ;ARECORD = ARECORD + PHYMSK + 1
4619  =1889 58                       POP bx
4620  =188A 53                       PUSH bx
4621  =188B 03C3                     ADD ax,bx
4622  =188D 8A1EBB08                 MOV bl,BLKMSK               ;END OF BLOCK?
4623  =1891 22D8                     AND bl,al
4624  =1893 B100                     MOV CL,0
4625  =1895 75E1          1878       JNZ FILL2                   ;NO
```

```
4626
4627  =1897 58                          POP BX
4628  =1898 8F061709                    POP ARECORD                    ;RESTORE ARECORD
4629  =                           dskwr1:
4630  =189C E87AFD          1619        CALL CHECK_NPRS
4631  =189F 7229            18CA        JC DONT_WRITE
4632  =18A1 7507            18AA          JNZ WRITE
4633  =18A3 B402                            MOV AH,2
4634  =18A5 E8D1F1          0A79            CALL DEBLOCK_DTA
4635  =18A8 EB20            18CA            JMPS DONT_WRITE
4636  =                           WRITE:
4637  =18AA E860F3          0C1A            CALL SETDATA
4638  =18AD E844EC          04F4            call seek                  ;to proper file position
4639  =18B0 E8A0EF          0853            call discard_data
4640  =18B3 8A0E2208                        mov cl,blk_off             ;set high water mark for
4641  =18B7 E84EF1          0A08            call ua_preread            ; preread check
4642  =18BA 8A0E0E09                        MOV CL,WFLAG               ;(cl=2 if new block,0 if not)
4643  =                                     ;set wflag to zero if blk_off
4644  =18BE 803E220800                      cmp blk_off,0              ;does not specify 1st record of
4645  =18C3 7402            18C7            JZ WRITE0                  ;new block
4646  =18C5 B100                              MOV CL,0
4647  =                           WRITE0:
4648  =18C7 E81EEF          07E8            call wrbuff                ;written to disk
4649  =                                     ;increment record count
4650  =                                     ;if rcount <= vrecord
4651  =                           DONT_WRITE:
4652  =18CA A01509                          mov al,vrecord
4653  =18CD BB1309                          mov bx,offset rcount
4654  =18D0 3A07                            cmp al,[bx]                ;vrecord-rcount
4655  =18D2 7220            18F4            jb dskwr2
4656  =                                     ;rcount <= vrecord
4657  =18D4 8807                              mov [bx],al
4658  =18D6 FE07                              inc b[bx]                  ;rcount = vrecord+1
4659  =
4660  =                                 if BMPM
4661  =                                 ;****   PATCH   04/08/83
4662  =                                 ;disabling extention of reccnt to block size for unlocked files
4663  =                                 ;****   OLD CODE
4664  =                                 ;         test high_ext,80h
4665  =                                 ;****   NEW CODE
4666  =18D8 F6069E0800                        test high_ext,00h
4667  =                                 ;****   END OF PATCH
4668  =18DD 7410            18EF              jz write1
4669  =
4670  =                                 ;       for unlocked file
4671  =                                 ;         rcount = vrecord & (~ blkmsk) + blkmsk + 1
4672  =
4673  =18DF 8A2E8B08                          mov ch,blkmsk
4674  =18E3 8AC8                              mov cl,ch
4675  =18E5 F6D1                              not cl
4676  =18E7 FEC5                              inc ch
4677  =18E9 22C1                              and al,cl
4678  =18EB 02C5                              add al,ch
```

```
4679
4680  =18ED 8807                        mov [bx],al
4681  =                           write1:
4682  =
4683  =                           endif
4684  =
4685  =18EF C6060E0902                  mov wflag,2                 ;mark as record count
4686  =                                                             ;incremented
4687  =                           dskwr2:
4688  =18F4 803E0E0902                  cmp wflag,2                 ;wflag=2 if new block or new record #
4689  =18F9 7517          1912          jnz noupdate
4690  =18FB 80264A087F                    and info_fcb+modnum,not fwfmsk
4691  =
4692  =                           if BMPM
4693  =1900 F6069E0880                    test high_ext,80h           ;is file open mode unlocked?
4694  =1905 7408          1912            jz dskwr2a
4695  =1907 8A07                            mov al,[bx]
4696  =1909 A2480B                          mov info_fcb+reccnt,al    ;restore fcb(rcount)
4697  =190C E88FFA        139E              call close
4698  =190F E8E8FC        15FA              call test_disk_fcb1
4699  =                           dskwr2a:
4700  =
4701  =                           endif
4702  =
4703  =                           noupdate:
4704  =1912 E835EE        0748          call getmodnum              ;set file write flag if reset
4705  =1915 2440                        and al,40h
4706  =1917 7508          1921          jnz dskwr3
4707  =1919 800F40                        or b[bx],40h
4708  =191C 80264A087F                    and info_fcb+modnum,not fwfmsk  ;reset fcb write flag to ensure
4709  =                                                             ;t3' gets reset by the close function
4710  =                           dskwr3:
4711  =1921 E9F5EC        0619          jmp setfcb                  ;replace parameters
4712  =                                                             ;ret
4713  =
4714  =                           compute_rr:     ;compute random record position for getfilesize/setrandom
4715  =                           ;----------
4716  =                           ;       entry:  BX = offset of fcb
4717  =                           ;               DX = index into fcb to pick record no. from
4718  =                           ;       exit:   AL,CX = random record number
4719  =                           ;               BX = offset of fcb
4720  =
4721  =1924 87DA                        xchg bx,dx
4722  =1926 03DA                        add bx,dx
4723  =                                                             ;dx=.buf(dptr) or .fcb(0),
4724  =                                                             ;bx=.f(nxtrec/reccnt)
4725  =1928 8A0F                        mov cl,[bx]                 ;pick up record no.
4726  =192A 32ED                        xor ch,ch                   ;cx = 0000 0000 ?rrr rrrr
4727  =192C 8BDA                        mov bx,dx
4728  =192E 8A670C                      mov ah,extnum[bx]
4729  =1931 D1E8                        shr ax,1
4730  =1933 25800F                      and ax,0f80h
4731  =1936 03C8                        add cx,ax                   ;cx = 000? eeee errrr rrrr
```

```
4732
4733  =1938 8A470E                  mov al,modnum[bx]               ;al = xxmm mmmm
4734  =193B 243F                    and al,3fh
4735  =193D B410                    mov ah,16
4736  =193F F6E4                    mul ah                          ;ax = 0000 00mm mmm0 0000
4737  =1941 02E8                    add ch,al
4738  =                                                             ;cx = mmmm eeee errr rrrr
4739  =1943 B000                    mov al,0
4740  =1945 12C4                    adc al,ah                       ;al = 0000 000m
4741  =1947 C3              ret19:  ret
4742  =
4743  =                     compare_rr:     ;compare dir to fcb random records
4744  =                     ;----------
4745  =                     ;       entry:  AL,CX = directory random record #
4746  =                     ;       exit:   C flag set
4747  =                     ;               BX = .fcb(rndrec)
4748  =
4749  =1948 BB5D0B                  mov bx,offset info_fcb+ranrec
4750  =194B 3A470275F7      1947    cmp al,2[bx]! jne ret19
4751  =1950 380F                    cmp cx,[bx]                     ;carry if .fcb(ranrec) >
4752  =1952 C3                      ret                             ;directory
4753  =
4754  =                     selectdisk:
4755  =                     ;----------
4756  =                     ;       select the disk drive given in AL, and fill
4757  =                     ;       the base addresses curtrka - alloca, then fill
4758  =                     ;       the values of the disk parameter block
4759  =                     ;       entry:  AL = disk to select
4760  =                     ;       exit:   C flag set if no error
4761  =
4762  =1953 8AC8                    mov cl,al                       ;current disk# to cl
4763  =                                                             ;lsb of dl = 0 if not yet
4764  =1955 B009                    mov al,io_seldsk                ;logged in
4765  =1957 E8CFEA          0429    call xiosif                     ;bx filled by call
4766  =                                                             ;bx = 0000 if error,
4767  =195A 0B0B                    or bx,bx                        ;otherwise disk headers
4768  =195C 743C            199A    jz ret4                         ;rz - carry flag reset
4769  =195E 4343                    inc bx! inc bx
4770  =1960 891EA608                mov cdrmaxa,bx
4771  =1964 4343                    inc bx! inc bx
4772  =1966 891EA808                mov drvlbla,bx                  ;media flag = drv tbl + 1
4773  =196A 83C304                  add bx,4                        ;lsn_add = drvlbl + 2
4774  =196D 8BF3                    mov si,bx
4775  =196F BFAC08                  MOV di,OFFSET DPBADDR           ;dx= source for move, bx=dest
4776  =1972 B90C00                  mov cx,addlist                  ;addlist filled
4777  =1975 F3A4                    rep movsb                       ;now fill the disk
4778  =1977 8B36AC08                mov si,dpbaddr                  ;parameter block
4779  =197B BF8808                  mov di,offset sectpt            ;bx is destination
4780  =197E B91100                  mov cx,dpblist
4781  =1981 F3A4                    rep movsb                       ;data filled
4782  =1983 8A0EC708                mov cl,physhf                   ;convert # sectors per track
4783  =1987 D3268808                shl sectpt,cl                   ; from physical to logical
4784  =                                                             ;set single/double map mode
```

```
4785
4786  =1988 A0BE08                    mov al,byte ptr maxall+1        ;largest allocation number
4787  =198E 0AC0                      or al,al
4788  =1990 7402          1994        jz retselect
4789  =                                                               ;if high order of maxall not
4790  =1992 B001                        mov al,1                      ;zero then use double disk map
4791  =                           retselect:
4792  =1994 FEC8                      dec al
4793  =1996 A21209                    mov single,al                   ;true if single disk map mode
4794  =1999 F9                        stc                             ;select disk function ok
4795  =                           ret4:
4796  =199A C3                        ret
4797  =
4798  =                           disk_select:
4799  =                           ;-----------
4800  =                           ;       entry:  AL = disk to select
4801  =                           ;       exit:   Z flag set if drive logged in
4802  =
4803  =199B A21609                    mov adrive,al
4804  =                           disk_select1:
4805  =199E A2A008                    mov curdsk,al
4806  =19A1 8B150808                  mov dx,dlog
4807  =19A5 E872ED        071A        call test_vector
4808  =19A8 52                        push dx                         ;dl = 1 if drive logged in
4809  =                                                               ;save it for test below,
4810  =                                                               ;send to seldsk
4811  =19A9 E8A7FF        1953        call selectdisk
4812  =19AC 5B                        pop bx                          ;recall dlog vector
4813  =19AD 7319          19C8        jnc select0                     ;carry set if select ok
4814  =19AF FECB                      dec bl                          ;is the disk logged in?
4815  =19B1 C3                        ret
4816  =
4817  =                           tmp_select:
4818  =                           ;----------
4819  =                           ;       entry:  DL = drive to select (0-f)
4820  =
4821  =19B2 88167F08                  mov seldsk,dl
4822  =
4823  =                           curselect:
4824  =                           ;---------
4825  =19B6 A07F08                    mov al,seldsk
4826  =19B9 3A06A008                  cmp al,curdsk
4827  =19BD 7505          19C4        jne select                      ;don't select if seldsk = curdsk
4828  =19BF FEC07405      19C8          inc all jz select0
4829  =19C3 C3                    ret19a:   ret
4830  =
4831  =                           SELECT:         ;select disk in info_fcb for subsequent input or output ops
4832  =                           ;------
4833  =19C4 3C107203      19C8        cmp al,16 ! jb sel0
4834  =                           select0:
4835  =19C8 E9A8EA        0473              jmp selerror
4836  =                           sel0:
4837  =19CB E8C0FF        1998        call disk_select
```

```
4838
4839  =19CE 74F3          19C3          jz ret19a                       ;rz
4840  =                                                                 ;return if bit is set
4841  =19D0 E88AF3        0D5D          call initialize                 ;disk not logged in,
4842  =                                                                 ;set bit and initialize
4843  =19D3 BB0808                      mov bx,offset dlog
4844  =19D6 E831ED        070A          call setcdisk                   ;dlog=set$cdisk(dlog)
4845  =
4846  =                         if BMPM
4847  =19D9 A0C408                      mov al,byte ptr chksiz+1        ;if high order bit of chksum
4848  =19DC D0D0                        rcl al,1                        ;vector is zero, then drive
4849  =19DE 7206          19E6          jb select2                      ;is removeable
4850  =19E0 BB0008                        mov bx,offset rlog
4851  =19E3 E824ED        070A            call setcdisk                 ;rlog=set_cdisk(rlog)
4852  =                         select2:
4853  =
4854  =                         endif
4855  =
4856  =19E6 C3                          ret
4857  =
4858  =                         reselectx:
4859  =                         ;---------
4860  =19E7 32C0                        xor al,al
4861  =19E9 A29E08                      mov high_ext,al
4862  =19EC A29F08                      mov xfcb_read_only,al
4863  =19EF EB27          1A18          jmps reselect1
4864  =
4865  =                         reselect:
4866  =                         ;--------
4867  =19F1 B97F80                      mov cx,807fh                    ;ch = 80h, cl = 7fh
4868  =19F4 BB430B                      mov bx,offset info_fcb+f7
4869  =19F7 8A07                        mov al,[bx]                     ;if fcb(7) & 80h
4870  =19F9 22C5                        and al,ch                       ;then xfcb_read_only = 80h
4871  =19FB A29F08                      mov xfcb_read_only,al           ;else xfcb_read_only = 00h
4872  =19FE 200F                        and [bx],cl                     ;fcb(7) = fcb(7) & 7fh
4873  =1A00 43                          inc bx
4874  =1A01 8A07                        mov al,[bx]
4875  =1A03 22C1                        and al,cl                       ;fcb(8) = fcb(8) & 7fh
4876  =1A05 3A07                        cmp al,[bx]
4877  =1A07 8807                        mov [bx],al                     ;if fcb(8) & 80h
4878  =1A09 B060                        mov al,60h                      ;then high_ext = 60h
4879  =1A0B 7505          1A12          jnz reselect0
4880  =1A0D 8A4704                        mov al,4[bx]                  ;else high_ext = fcb(12) & e0h
4881  =1A10 24E0                          and al,0e0h
4882  =                         reselect0:
4883  =1A12 A29E08                      mov high_ext,al
4884  =1A15 E83FED        0757          call clr_ext
4885  =                         reselect1:      ;check current fcb to see if reselection necessary
4886  =1A18 C6060F08FF                  mov resel,true                  ;mark possible reselect
4887  =1A1D A03C0B                      mov al,info_fcb                 ;drive select code
4888  =1A20 A20B08                      mov fcbdsk,al                   ;save drive
4889  =1A23 241F                        and al,00011111b                ;non zero is auto drive select
4890  =1A25 FEC8                        dec al                          ;drive code normalized to
```

```
4891
4892  =                                                     ;0...30, or 255
4893  =1A27 3CFF                cmp al,0ffh
4894  =1A29 7403          1A2E  jz noselect                 ;auto select function,
4895  =1A2B A27F08                mov seldsk,al             ;save seldsk
4896  =                   noselect:
4897  =1A2E E885FF        1986  call curselect
4898  =                                                     ;set user code
4899  =1A31 A08008                mov al,usrcode            ;0...31
4900  =1A34 A23C0B                mov info_fcb,al
4901  =                   noselect0:
4902  =1A37 E867ED        07A1  CALL TST_LOG_FXS            ;IS FX = 15,17,22,23,30?
4903  =1A3A 7505          1A41  JNZ noselect1
4904  =1A3C C6061008F0            mov rd_dir_flag,0f0h      ; yes - must read directory to
4905  =                   noselect1:
4906  =1A41 E83C00        1A80  call check_media            ;chk media change on current disk
4907  =                         ;jmps check_all_media       ;chk media change on other disks
4908  =
4909  =                   check_all_media:
4910  =                   ;---------------
4911  =                   ;       This routine checks all logged-in drives for
4912  =                   ;       a set DPH media flag and pending buffers. It reads
4913  =                   ;       the directory for these drives to verify the media
4914  =                   ;       has not changed. If the media has changed the drive
4915  =                   ;       gets reset (but not relogged-in).
4916  =
4917  =1A44 32C0                  xor al,al
4918  =1A46 86060E0C              xchg media_flag,al        ;reset SCB media flag
4919  =1A4A 84C0                  test al,al                ;was SCB media flag set
4920  =1A4C 7478          1AC9  jz ret20                    ; no
4921  =1A4E 8B1E0808              mov bx,dlog               ;test logged-in drives only
4922  =1A52 B010                  mov al,16
4923  =                   chk_am1:
4924  =1A54 FEC8                  dec al
4925  =1A56 D1E3731F      1A79  shl bx,1l jnc chk_am5       ;is drive logged-in
4926  =1A5A 5053                  push axl push bx          ; yes
4927  =1A5C E83CFF        199B    call disk_select          ;get DPH
4928  =1A5F 8B1E8408              mov bx,dat_bcba
4929  =1A63 8B1F                  mov bx,[bx]
4930  =                   chk_am2:
4931  =1A65 0BDB740E      1A77    or bx,bxl jz chk_am4      ;end of bcb list?
4932  =1A69 F64704FF                test byte ptr 4[bx],0ffh  ; no - is buffer pending
4933  =1A6D 7505          1A74      jnz chk_am3             ;  yes
4934  =1A6F 8B5F0C                    mov bx,12[bx]         ;get next bcb
4935  =1A72 EBF1          1A65        jmps chk_am2
4936  =                   chk_am3:
4937  =1A74 E80900        1A80      call check_media        ;write pending, chk media change
4938  =                   chk_am4:
4939  =1A77 5B58                  pop bxl pop ax
4940  =                   chk_am5:
4941  =1A79 0AC075D7      1A54  or al,all jnz chk_am1
4942  =1A7D E936FF        1986  jmp curselect
4943  =
```

```
4944
4945  =                       check_media:    ;chk media if DPh media flag set
4946  =                       ;-----------
4947  =1A80 8B1EA808                  mov bx,drvlbla
4948  =1A84 32C0                      xor al,al
4949  =1A86 864701                    xchg 1[bx],al                   ;RESET MEDIA FLAG
4950  =1A89 0AC0                      or al,al                        ;was media flag set
4951  =1A8B 743C        1AC9          JZ RET20                        ;NO
4952  =                                                             ;DISCARD DIRECTORY BUFFERS
4953  =1A8D E8CBED      0858          CALL discard_dir                ;READ DIRECTORY TO TEST FOR MEDIA
4954  =1A90 E8E9EC      077C          CALL SETENDDIR                  ;CHANGE
4955  =1A93 FF368F08                  push dcnt                       ;save dcnt incase media hasnt changed
4956  =                       check_media1:
4957  =1A97 B100                      MOV CL,FALSE
4958  =1A99 E88BF1      0C27          CALL RDIR
4959  =1A9C 32C0                      xor al,al                       ;reset relog flag
4960  =1A9E 86062108                  xchg relog,al
4961  =1AA2 84C0                      test al,al                      ;HAS CHECKSUM ERROR BEEN DETECTED?
4962  =1AA4 741A        1AC0          jz check_media2                 ; no
4963  =1AA6 803E7A0821                  cmp fx_intrn,fxi48            ;is func# = flush buffer
4964  =1AAB 7418        1AC5            jz check_media3               ; yes
4965  =1AAD A01609                      mov al,adrive                 ;is flush to another drive ?
4966  =1AB0 3A067F08                    cmp al,seldsk
4967  =1AB4 750F        1AC5            jnz check_media3              ; yes
4968  =1AB6 8F068F08                    pop dcnt                      ;restore dcnt
4969  =1ABA E81FED      07DC            call drv_relog
4970  =1ABD E9F6EC      07B6            jmp chk_exit_fxs
4971  =                       check_media2:
4972  =1AC0 E8CAEC      078D          CALL COMPCDR                    ;HAS DIR HIGH WATER MARK BEEN REACHED
4973  =1AC3 72D2        1A97          jc check_media1                 ; no
4974  =                       check_media3:
4975  =1AC5 8F068F08                  pop dcnt                        ;restore dcnt media has not changed
4976  =1AC9 C3                ret20:  ret
4977  =
4978  =                       copy_dma_in_8:
4979  =                       ;-------------
4980  =1ACA B108                      mov cl,8
4981  =
4982  =                       copy_dma_in:    ;copy (cl) bytes from user's
4983  =                       ;-----------
4984  =                       ;       entry:  CL = number of bytes to copy
4985  =
4986  =1ACC 8B358508                  mov si,dma_ofst                 ;dma to common_dma
4987  =1AD0 BFC908                    mov di,offset common_dma
4988  =1AD3 1E                        push ds
4989  =1AD4 8E1E8708                  mov ds,dma_seg
4990  =1AD8 32ED                      xor ch,ch
4991  =1ADA F3A4                      rep movs al,al
4992  =1ADC 1F                        pop ds
4993  =1ADD C3                        ret
4994  =
4995  =                       get_dir_mode:
4996  =                       ;------------
```

```
4997
4998  =                         ;       exit:   AL = directory label data byte
4999  =
5000  =1ADE 881EA808                     mov bx,drvlbla                  ;return dir lbl data byte in al
5001  =1AE2 8A07                         mov al,[bx]
5002  =1AE4 C3                           ret
5003  =
5004  =                         get_xfcb:       ;get xfcb (search mode)
5005  =                         ;--------
5006  =                         ;       exit:   Z flag set if no xfcb exists
5007  =                         ;               BX = .xfcb(pwmode)
5008  =                         ;               DX = .xfcb(0)
5009  =                         ;               AL = pwmode + 1
5010  =
5011  =1AE5 BB3C08                       mov bx,offset info_fcb
5012  =1AE8 8A07                         mov al,[bx]
5013  =1AEA 50                           push ax
5014  =1AEB 800F10                       or b[bx],10h                    ;fcb(0) = fcb(0) | 10h
5015  =1AEE E852F5       1043            call search_ext                 ;returns with zero flag set if
5016  =1AF1 58                           pop ax                          ; xfcb not found
5017  =1AF2 A23C08                       mov info_fcb,al                 ;restore fcb(0)
5018  =1AF5 C6060D0800                   mov lret,0                      ;zero lret
5019  =1AFA 7501         1AFD            jnz get_xfcb1
5020  =1AFC C3                             ret                           ;xfcb not found - al = 0, zflag set
5021  =                         get_xfcb1:
5022  =1AFD E825EC       0725            call get_dptra                  ;get directory pointer
5023  =1B00 8BD3                         mov dx,bx                       ;DX = .directory_xfcb(0)
5024  =1B02 83C30C                       add bx,extnum
5025  =1B05 8A07                         mov al,[bx]
5026  =1B07 24E0                         and al,0e0h
5027  =1B09 0C01                         or al,1                         ;al = xfcb(extnum) & 0e0h + 1
5028  =1B0B C3                           ret                             ;xfcb found, zero flag reset
5029  =
5030  =                         init_xfcb:
5031  =                         ;---------
5032  =1B0C E889EC       0798            call setcdr                     ;may have extended the directory
5033  =1B0F B91410                       mov cx,1014h                    ;ch = 16, cl = 20
5034  =                         init_xfcb0:                               ;ch = fcb(0) logical or mask
5035  =                                                                   ;cl = zero count
5036  =1B12 51                           push cx
5037  =1B13 E80FEC       0725            call get_dptra                  ;does not use cx
5038  =1B16 BE3C0B                       mov si,offset info_fcb          ;xfcb(0) = fcb(0) | ch
5039  =1B19 AC                           lodsb
5040  =1B1A 0AC5                         or al,ch
5041  =1B1C 8807                         mov [bx],al
5042  =1B1E 43                           inc bx
5043  =1B1F B10B                         mov cl,11
5044  =1B21 8BD6                         mov dx,si
5045  =1B23 E8BCE9       04E2            call move                       ;xfcb(1..11) = fcb(1..11)
5046  =1B26 8BD6                         mov dx,si                       ;dx = .fcb(12)
5047  =1B28 8BDF                         mov bx,di                       ;bx = .xfcb(12)
5048  =1B2A 59                           pop cx                          ;cl contains # of remaining bytes
5049  =1B2B 2AED                         sub ch,ch                       ;of xfcb to zero
```

```
5050
5051  =1B2D 32C0                    xor al,al                       ;move sets di to addr of next char
5052  =1B2F F3AA                    rep stos al
5053  =1B31 C3                      ret
5054  =
5055  =                         chk_pw_error:
5056  =                         ;------------
5057  =1B32 C606160800          mov byte ptr xdcnt+1,0              ;disable special searches
5058  =1B37 BE3C0B              mov si,offset info_fcb
5059  =1B3A BF640B              mov di,offset save_fcb              ;save info_fcb
5060  =1B3D B91000              mov cx,16
5061  =1B40 F3A4                rep movsb
5062  =1B42 E8E0EB      0725    call get_dptra
5063  =1B45 88F346              mov si,bx! inc si
5064  =1B48 BF3D0B              mov di,offset info_fcb+1            ;info_fcb = dir_xfcb
5065  =1B4B B90B00              mov cx,11
5066  =1B4E F3A4                rep movsb
5067  =1B50 32C0AA              xor al,al! stos al                  ;fcb(extnum) = 0
5068  =1B53 47AA                inc di!stos al                      ;fcb(modnum) = 0
5069  =1B55 AC                  lods al                             ;save password mode
5070  =1B56 A2D008              mov pw_mode,al
5071  =1B59 E8E3F4      103F    call search_name                    ;does fcb(ext=0,mod=0) exist
5072  =1B5C 743F        1B9D    jz chk_pwe2                         ; no
5073  =1B5E E8E200      1C43      call get_dtba8                    ;does sfcb exist
5074  =1B61 0AC07525    1B8A      or al,al! jnz chk_pwe1            ; no
5075  =1B65 BED008                  mov si,offset pw_mode
5076  =1B68 8A2C                    mov ch,[si]                     ;CH = xfcb.pwmode
5077  =1B6A 8A07                    mov al,[bx]                     ;AL = sfcb.pwmode
5078  =1B6C 8804                    mov [si],al                     ;pw_mode = sfcb.pwmode
5079  =1B6E 0AC0742B    1B9D        or al,al! jz chk_pwe2           ;is sfcb password mode nonzero
5080  =1B72 32C524E0                xor al,ch! and al,0e0h          ; yes
5081  =1B76 7412        1B8A        jz chk_pwe1                     ;do password modes match
5082  =1B78 E86AFF      1AE5        call get_xfcb                   ; no - update xfcb
5083  =1B7B 740D        1B8A        jz chk_pwe1                     ;error - no xfcb
5084  =1B7D A0D008                  mov al,pw_mode
5085  =1B80 8807                    mov [bx],al                     ;xfcb.pwmode = sfcb.pwmode
5086  =1B82 E891EB      0716        call nowrite
5087  =1B85 7503        1B8A        jnz chk_pwe1
5088  =1B87 E883F0      0C0D        call wrdir
5089  =                         chk_pwe1:
5090  =1B8A E82700      1BB4        call chk_pwe3
5091  =1B8D A07A08                  mov al,fx_intrn                 ;is func# = open or make?
5092  =1B90 3C027428    1BBF        cmp al,fxi15! je ret20a
5093  =1B94 3C097427    1BBF        cmp al,fxi22! je ret20a
5094  =                         pw_error:                           ; no
5095  =1B98 B407                    mov ah,7                        ;password error
5096  =1B9A E917E9      04B4        jmp set_aret
5097  =                         chk_pwe2:
5098  =1B9D C6060D0800          mov pw_mode,0
5099  =1BA2 E871EB      0716        call nowrite                    ;delete unused xfcb
5100  =1BA5 750D        1BB4        jnz chk_pwe3
5101  =1BA7 E83BFF      1AE5        call get_xfcb                   ;get xfcb
5102  =1BAA 7408        1BB4        jz chk_pwe3                     ; it should have been there
```

```
5103
5104  =18AC 800E3C0B10                or info_fcb,10h
5105  =18B1 E826F6        11DA        call delete11                ;delete xfcb
5106  =                         chk_pwe3:
5107  =18B4 BE6408                    mov si,offset save_fcb
5108  =18B7 BF3C0B                    mov di,offset info_fcb       ;restore info_fcb
5109  =18BA B91000                    mov cx,16
5110  =18BD F3A4                      rep movsb
5111  =18BF C3                  ret20a: ret
5112  =
5113  =                         chk_password:
5114  =                         ;------------
5115  =18C0 E818FF        1ADE        call get_dir_mode            ;al = dir lbl data byte
5116  =18C3 2480                      and al,80h
5117  =18C5 74F8          18BF        jz ret20a                    ;bit 0 on => passwords active
5118  =18C7 E818FF        1AE5        call get_xfcb                ;al = xfcb password mode | 01h
5119  =18CA 74F3          18BF        jz ret20a                    ;zero set if no xfcb found
5120  =
5121  =                         cmp_pw:         ;compare passwords
5122  =                         ;------
5123  =                         ;       entry:  BX = .xfcb(12)
5124  =                         ;       exit:   Z flag set if password is valid
5125  =
5126  =18CC 43                       inc bx
5127  =18CD 8A2F                     mov ch,[bx]                  ;fcb(13) = xfcb checksum
5128  =18CF 0AED                     or ch,ch
5129  =18D1 7514          18E7       jnz cmp_pw2                  ;jump if xfcb(13) <> 0
5130  =18D3 8BF3                       mov si,bx                  ;save bx
5131  =18D5 83C603                     add si,3                   ;because xfcb checksum is zero
5132  =                                                           ;test for null password
5133  =18D8 B109                       mov cl,9                   ;null passwords are all blanks and/or
5134  =                         cmp_pw1:                          ;zeros that sum to zero
5135  =18DA AC                         lodsb                      ;any password matches a null password
5136  =18DB FEC9                       dec cl
5137  =18DD 74E0          18BF         jz ret20a                  ;password is all blanks and/or zeros
5138  =18DF 0AC0                       or al,al
5139  =18E1 74F7          18DA         jz cmp_pw1
5140  =18E3 3C20                       cmp al,20h
5141  =18E5 74F3          18DA         jz cmp_pw1                 ;password contains a value other
5142  =                                                           ;than blank or zero
5143  =                         cmp_pw2:                          ;bx = .xfcb(13)
5144  =18E7 8D770A                    lea si,10[bx]
5145  =18EA 8D5703                    lea dx,3[bx]               ;dx = .xfcb(16)
5146  =18ED BBC908                    mov bx,offset common_dma
5147  =18F0 B108                      mov cl,8
5148  =18F2 FD                        std                        ;lods decrements si
5149  =                         cmp_pw3:
5150  =18F3 AC                        lods al
5151  =18F4 32C5                      xor al,ch                  ;exclusive or password byte with cks
5152  =18F6 3A07                      cmp al,[bx]
5153  =18F8 7507          1C01        jnz cmp_pw4                ;password mismatch
5154  =18FA 43                          inc bx
5155  =18FB FEC9                        dec cl
```

```
5156
5157  =18FD 75F4        18F3          jnz cmp_pw3
5158  =18FF FC                        cld
5159  =1C00 C3                        ret                             ;password matches
5160  =                         cmp_pw4:
5161  =1C01 FC                        cld
5162  =1C02 BB9508                    mov bx,offset df_password       ;compare xfcb password to default pw
5163  =1C05 B108                      mov cl,8                        ;bx = .default pw, dx = .xfcb password
5164  =1C07 E9E1E8      04EB          jmp compare
5165  =
5166  =                         chk_xfcb_password:
5167  =                         ;-----------------
5168  =                         ;       exit:   BX = .xfcb(12)
5169  =
5170  =1C0A E8F0FE      1AFD          call get_xfcb1                  ;access xfcb in directory buffer
5171  =                                                               ;al = (xfcb(ext) & e0h) | 1
5172  =                         chk_xfcb_password1:
5173  =1C0D 53                        push bx                         ;bx = .xfcb(ex)
5174  =1C0E E8BBFF      18CC          call cmp_pw                     ;check password
5175  =1C11 5B                        pop bx
5176  =1C12 C3                        ret                             ;return with bx = .xfcb(ex)
5177  =
5178  =                           set_pw:         ;set password in xfcb
5179  =                           ;------
5180  =                           ;       entry:  BX = .xfcb(12)
5181  =                           ;               SI = .password in common dma area
5182  =
5183  =1C13 B90800                    mov cx,8                        ;ch = 0, cl = 8
5184  =1C16 8D7F08                    lea di,11[bx]                   ;di = .xfcb(23)
5185  =                         set_pw0:
5186  =1C19 2AE4                      sub ah,ah                       ;zero null password flag
5187  =                         set_pw1:
5188  =1C1B AC                        lodsb
5189  =1C1C 8805                      mov [di],al                     ;copy password char to xfcb
5190  =1C1E 0AC0                      or al,al                        ;is password char zero?
5191  =1C20 7406        1C28          jz set_pw2                      ;yes
5192  =1C22 3C20                      cmp al,20h                      ;is password char blank?
5193  =1C24 7402        1C28          je set_pw2                      ;yes
5194  =1C26 FEC4                        inc ah                        ;password is not null
5195  =                         set_pw2:
5196  =1C28 02E8                      add ch,al                       ;add password char to xfcb chksum
5197  =1C2A 4F                        dec di
5198  =1C2B FEC9                      dec cl
5199  =1C2D 75EC        1C1B          jnz set_pw1
5200  =1C2F 0AE5                      or ah,ch                        ;if (ah | ch) = 0 then password is null
5201  =1C31 7502        1C35          jnz set_pw3
5202  =1C33 8827                        mov [bx],ah                   ;xfcb(ex) = 0, zero password mode
5203  =                         set_pw3:
5204  =1C35 47                        inc di
5205  =1C36 B108                      mov cl,8
5206  =                         set_pw4:
5207  =1C38 3029                      xor [di],ch                     ;encrypt password chars in xfcb
5208  =1C3A 47                        inc di
```

```
5209
5210  =1C3B FEC9                    dec cl
5211  =1C3D 75F9          1C38      jnz set_pw4
5212  =1C3F 886F01                  mov 1[bx],ch                ;bx + 1 = .xfcb(13)
5213  =1C42 C3                      ret                         ;ch = xfcb password checksum
5214  =
5215  =                       get_dtba8:      ;get password mode address
5216  =                       ;---------
5217  =1C43 B508                    mov ch,8
5218  =
5219  =                       GET_DTBA:       ;GET STAMP ADDRESS IN SFCB
5220  =                       ;--------
5221  =                       ;       entry:  CH = offset into sfcb
5222  =                       ;       exit:   BX = address of offset into sfcb
5223  =
5224  =1C45 B003                    MOV AL,3                    ;RET WITH AL <> 0 IF STAMP CANNOT
5225  =1C47 8A268F08                mov ah,byte ptr dcnt        ;BE MADE
5226  =1C4B 80E403                  and ah,dskmsk
5227  =1C4E 3AC4                    CMP AL,AH                   ;IS LRET = 3? (SFCB POSITION)
5228  =1C50 741B          1C6D      JZ RET21A                   ;YES
5229  =1C52 881EAA08                mov bx,buffa
5230  =1C56 83C360                  add bx,96
5231  =1C59 8A07                    MOV AL,[BX]
5232  =1C5B 2C21                    SUB AL,21H                  ;DOES DIR FCB(3) BEGIN WITH 21H?
5233  =1C5D 750E          1C6D      JNZ RET21A                  ;NO
5234  =1C5F 8AC4                    MOV AL,ah
5235  =1C61 B10A                    MOV CL,10
5236  =1C63 F6E1                    MUL CL                      ;AL = LRET*10
5237  =1C65 FEC0                    INC AL                      ;     + 1
5238  =1C67 02C5                    ADD AL,CH                   ;     + CH
5239  =1C69 03D8                    ADD BX,AX                   ;RETURN WITH BX = STAMP ADDR
5240  =1C6B 32C0                    xor al,al
5241  =                       RET21A:
5242  =1C6D C3                      RET
5243  =
5244  =                       QDIRFCB1:
5245  =                       ;--------
5246  =                       ;       exit:   Z flag reset if fcb is first dir fcb of file
5247  =                       ;               CL = preserved
5248  =
5249  =1C6E 8B480B                  mov bx,offset info_fcb+extnum   ;IS FCB IN 1ST DIR FCB OF FILE?
5250  =1C71 8A07                    MOV AL,[BX]
5251  =1C73 8A26BC08                mov ah,extmsk
5252  =1C77 F6D4                    not ah
5253  =1C79 22C4                    and al,ah
5254  =1C7B 241F                    and al,01fh
5255  =1C7D 75EE          1C6D      JNZ RET21A                  ;NO
5256  =1C7F F647023F                TEST BYTE PTR 2[BX],3FH     ;IS FCB(MOD) = 0?
5257  =1C83 C3                      RET                         ;Z FLAG RESET IF NOT
5258  =
5259  =                       QSTAMP:
5260  =                       ;------
5261  =                       ;       exit:   Z flag reset if stamp is requested
```

```
5262
5263  =
5264  =1C84 E8E7FF      1C6E          CALL QDIRFCB1                 ;IS FCB IN 1ST DIR FCB OF FILE?
5265  =1C87 75E4        1C6D          JNZ RET21A                    ;NO
5266  =                         QSTAMP1:
5267  =1C89 E852FE      1ADE          call get_dir_mode             ;ARE REQUIRED DIR LBL OPTIONS SET?
5268  =1C8C 22C17403    1C93          and al,cl1 jz $+5             ; NO
5269  =1C90 E983EA      0716            JMP NOWRITE                 ; yes - VERIFY DISK NOT READ/ONLY
5270  =1C93 FEC0                      inc al                        ;return z flag reset
5271  =1C95 C3                        ret
5272  =
5273  =                         stamp1:
5274  =                         ;------
5275  =1C96 B500                      MOV CH,0
5276  =1C98 EB02        1C9C          jmps stamp3
5277  =
5278  =                         stamp2:
5279  =                         ;------
5280  =1C9A B504                      MOV CH,4
5281  =
5282  =                         stamp3:
5283  =                         ;------
5284  =                         ;       entry:  CH = stamp offset
5285  =
5286  =1C9C E8A6FF      1C45          CALL GET_DTBA                 ;GET STAMP ADDRESS
5287  =1C9F 0AC0                      OR AL,AL
5288  =1CA1 75CA        1C6D          JNZ ret21a                    ;NON-ZERO - CAN'T STAMP
5289  =                         STAMP4:
5290  =1CA3 BA7E00                    mov dx,offset tod
5291  =1CA6 B104                      MOV CL,4
5292  =1CA8 5352                      PUSH BX! PUSH DX
5293  =1CAA E83EE8      04EB          CALL COMPARE                  ;COMPARE EXISTING STAMP TO NEW STAMP
5294  =1CAD 5A5B                      POP DX! POP BX
5295  =1CAF 74BC        1C6D          jz ret21a                     ;EQUAL - SKIP CALL TO WRDIR
5296  =1CB1 B104                      MOV CL,4
5297  =1CB3 E82CE8      04E2          call move
5298  =1CB6 E954EF      0C0D          jmp wrdir
5299  =
5300  =                         STAMP5:         ;DIR LBL STAMP ENTRY
5301  =                         ;------
5302  =                         ;       entry:  CX = stamp offset
5303  =
5304  =1CB9 E869EA      0725          CALL GET_DPTRA
5305  =1CBC 03D9                      ADD BX,CX
5306  =1CBE B8B104                    MOV AX,OFFSET FUNCRET
5307  =1CC1 50                        PUSH AX
5308  =1CC2 EBDF        1CA3          jmps STAMP4
5309  =
5310  =                         update_stamp:
5311  =                         ;------------
5312  =1CC4 B120                      mov cl,20h                    ;is update stamp requested
5313  =1CC6 E8C0FF      1C89          call qstamp1                  ; on drive
5314  =1CC9 75A2        1C6D          jnz ret21a                    ; no
```

```
5315
5316  =1CCB F6064A0B40              test info_fcb+modnum,40h        ;has file been written to since
5317  =1CD0 7598          1C6D      jnz ret21a                      ; it was opened
5318  =1CD2 8A264808                mov ah,info_fcb+extnum          ; yes - search for first
5319  =1CD6 A04A08                  mov al,info_fcb+modnum          ;  dir fcb
5320  =1CD9 50                      push ax                         ;save extnum and modnum
5321  =1CDA E85AF3        1037      call search_1_name
5322  =1CDD 7403          1CE2      jz $+5                          ;perform update stamp
5323  =1CDF E8B8FF        1C9A        call stamp2                   ; if 1st dir fcb found
5324  =1CE2 C6060D0800              mov lret,0
5325  =1CE7 58                      pop ax                          ;restore extnum and modnum
5326  =1CE8 8826480B                mov info_fcb+extnum,ah
5327  =1CEC A24A08                  mov info_fcb+modnum,al
5328  =1CEF C3                      ret
5329  =
5330  =
5331  =                       if BMPM
5332  =
5333  =                       pack_sdcnt:     ;packed_dcnt = sdcnt
5334  =                       ;----------
5335  =1CF0 A10909                  mov ax,sdcnt
5336  =1CF3 A3A108                  mov word ptr packed_dcnt,ax
5337  =1CF6 C606A30800              mov byte ptr packed_dcnt+2,0
5338  =1CFB C3                      ret
5339  =
5340  =                       ;
5341  =                       ;       olist element:
5342  =                       ;           +------+------+------+------+------+------+
5343  =                       ;       00h |  LINK      | ATTS |        DCNT        |
5344  =                       ;           +------+------+------+------+------+------+
5345  =                       ;       06h |   PDADDR    |   OPNCNT    |
5346  =                       ;           +------+------+------+------+
5347  =                       ;
5348  =                       ;       link = 0 -> end of list
5349  =                       ;
5350  =                       ;       atts - 8x - open in locked mode
5351  =                       ;              4x - open in unlocked mode
5352  =                       ;              2x - open in read/only mode
5353  =                       ;              1x - deleted item
5354  =                       ;              xN - drive code (0-f)
5355  =                       ;
5356  =                       ;       dcnt = packed sdcnt+sdblk
5357  =                       ;       pdaddr = process descriptor addr
5358  =                       ;       opncnt = # of open calls - # of close calls
5359  =                       ;                 olist item freed by close when opncnt = 0
5360  =                       ;
5361  =                       ;       llist element:
5362  =                       ;           +------+------+------+------+------+------+
5363  =                       ;       00h |    LINK      | DRV  |       RECORD       |
5364  =                       ;           +------+------+------+------+------+------+
5365  =                       ;       06h |      CNT     | .OLISTITEM  |
5366  =                       ;           +------+------+------+------+
5367  =                       ;
```

```
5368
5369  =                         ;       link = 0 -> end of list
5370  =                         ;
5371  =                         ;       drive - 0x - Prevent Other Reads + Writes
5372  =                         ;               8x - Prevent All Writes - allow reads if PAW locked
5373  =                         ;               xN - drive code (0-f)
5374  =                         ;
5375  =                         ;       record = starting record number of locked records
5376  =                         ;       count = # of locked records starting at record
5377  =                         ;       .olist_item = address of file's olist item
5378  =                         ;
5379  =                         search_olist:
5380  =                         ;------------
5381  =1CFC B80509E809   1D0A          mov bx,offset open_root! jmps search_listo
5382  =
5383  =                         search_llist:
5384  =                         ;------------
5385  =1D01 B80709EB04   1D0A          mov bx,offset lock_root! jmps search_listo
5386  =
5387  =                         searchn_list:
5388  =                         ;------------
5389  =1D06 881EF708                   mov bx,cur_pos
5390  =
5391  =                         search_listo:
5392  =                         ;------------
5393  =1D0A 891EF908                   mov prv_pos,bx
5394  =
5395  =                         ;search_olist, search_llist, searchn_list conventions
5396  =                         ;
5397  =                         ;       entry:  CH = 0 -> return next item
5398  =                         ;               CH = 1 -> search for matching drive
5399  =                         ;               CH = 3 -> search for matching dcnt
5400  =                         ;               CH = 5 -> search for matching dcnt + pdaddr
5401  =                         ;
5402  =                         ;       exit:   Z flag set if found and
5403  =                         ;                 prv_pos -> previous list element
5404  =                         ;                 cur_pos -> found list element
5405  =                         ;                 BX = cur_pos
5406  =                         ;               else prv_pos -> list element to insert after
5407  =                         ;
5408  =                         ;       olist and llist are maintained in drive order
5409  =                         ;
5410  =                         srch_list1:
5411  =1D0E 881F                       mov bx,[bx]
5412  =1D10 0BDB7503     1D17          or bx,bx! jnz srch_list2        ;if (end of list)
5413  =1D14 0C01                       or al,1                         ; return( NZ )
5414  =1D16 C3                         ret
5415  =                         srch_list2:
5416  =1D17 0AED7430     1D43          or ch,ch! je srch_list5
5417  =1D1B 8A4F0280E10F               mov cl,2[bx]! and cl,0fh
5418  =1D21 BEA008                     mov si,offset curdsk
5419  =1D24 AC                         lods al
5420  =1D25 3AC17520     1D49          cmp al,cl! jne srch_list4       ;do drive # match?
```

```
5421
5422  =1D29 80FD017410  1D48          cmp ch,1! je srch_list5     ; is search for matching drive only?
5423  =                                                           ;SI = offset packed_dcnt
5424  =1D2E 8AC5                      mov cl,ch
5425  =1D30 51                        push cx
5426  =1D31 32C0                      xor al,al
5427  =1D33 864705                    xchg al,5[bx]               ;save f1' env switch
5428  =                                                           ;set dcnt+2 to zero
5429  =1D36 32ED                      xor ch,ch                   ;do items match?
5430  =1D38 8D7F03                    lea di,3[bx]                ;DI = offset item.dcnt
5431  =1D3b F3A6                      repe cmps al,al
5432  =1D3D 884705                    mov 5[bx],al                ;restore f1' env switch
5433  =1D40 59                        pop cx
5434  =1D41 7408        1D48          jz srch_list5
5435  =                         srch_list3:
5436  =1D43 891EF908                  mov prv_pos,bx              ; no - check next item
5437  =1D47 EBC5        1D0E          jmps srch_list1
5438  =                         srch_list4:
5439  =1D49 73F8        1D43          jae srch_list3              ;no need to continue if curdsk < drv
5440  =                         srch_list5:
5441  =1D4B 891EF708                  mov cur_pos,bx
5442  =1D4F C3                        ret
5443  =
5444  =                         delete_item:
5445  =                         ;-----------
5446  =                         ;       entry:  BX = item to be deleted
5447  =
5448  =1D50 9CFA                      pushf! cli
5449  =1D52 8B37                      mov si,[bx]
5450  =1D54 8B3EF908                  mov di,prv_pos
5451  =1D58 893EF708                  mov cur_pos,di              ;prv_pos.link = delete_item.link
5452  =1D5C 8935                      mov [di],si
5453  =1D5E 8B365200                  mov si,free_root
5454  =1D62 891E5200                  mov free_root,bx            ;delete_item.link = previous free_root
5455  =1D66 8937                      mov [bx],si
5456  =1D68 9D                        popf
5457  =1D69 8A4F0280E10F              mov cl,2[bx]! and cl,0fh    ;get drive of deleted entry
5458  =1D6F BB0509                    mov bx,offset open_root
5459  =                         chk_open1:
5460  =1D72 8B1F                      mov bx,[bx]                 ;do any open files
5461  =1D74 0BDB740A    1D82          or bx,bx! jz chk_open2      ; exist on drive
5462  =1D78 8A4702240F                  mov al,2[bx]! and al,0fh
5463  =1D7D 3AC1                        cmp al,cl
5464  =1D7F 75F1        1D72            jne chk_open1
5465  =1D81 C3                          ret
5466  =                         chk_open2:
5467  =1D82 BB8800                    mov bx,offset open_vector   ; no - zero drive bit
5468  =1D85 EB04        1D8B          jmps remove_drive1          ; in open file vector
5469  =                               ;ret
5470  =
5471  =                         remove_drive:
5472  =                         ;------------
5473  =1D87 8A0EA008                  mov cl,curdsk
```

```
5474
5475  =
5476  =                         remove_drive1:  ;reset bit in vector at dx
5477  =                         ;-------------
5478  =                         ;       entry:  BX = address of vector
5479  =                         ;               CL = bit in vector to reset
5480  =
5481  =1D8B B80100D3E0                  mov ax,1l shl ax,cl
5482  =1D90 F7D02107                    not axl and [bx],ax
5483  =1D94 C3                          ret
5484  =
5485  =                         remove_locks:
5486  =                         ;------------
5487  =                         ;       entry:  BX = offset of olist_item
5488  =                         ;       exit:   BX = preserved
5489  =
5490  =1D95 F64702407423 1DBC          test byte ptr 2[bx],40hl jz ret125
5491  =1D9B FF36F908                    push prv_pos
5492  =1D9F 53                          push bx
5493  =1DA0 B80709                      mov ax,offset lock_root
5494  =1DA3 A3F708                      mov cur_pos,ax
5495  =                         remove_lock1:
5496  =1DA6 b500                        mov ch,0
5497  =1DA8 E858FF        1D06          call searchn_list
5498  =1DAB 750C          1DB9          jnz remove_lock2
5499  =1DAD 5A52                        pop dxl push dx
5500  =1DAF 395708                      cmp 8[bx],dx
5501  =1DB2 75F2          1DA6          jnz remove_lock1
5502  =1DB4 E899FF        1D50          call delete_item
5503  =1DB7 EBED          1DA6          jmps remove_lock1
5504  =                         remove_lock2:
5505  =1DB9 5B8F06F908                  pop bxl pop prv_pos
5506  =                         ret125:
5507  =1DBE C3                          ret
5508  =
5509  =                         compare_pds:
5510  =                         ;-----------
5511  =                         ;       exit:   Z flag set if proccess descriptors match
5512  =
5513  =1DBF A1A408                      mov ax,pdaddr
5514  =1DC2 394706                      cmp 6[bx],ax
5515  =1DC5 C3                          ret
5516  =
5517  =                         tst_olist:
5518  =                         ;---------
5519  =1DC6 C606EF0800                  mov chk_olist_flag,false
5520  =1DCB EB05          1DD2          jmps chk_olist0
5521  =
5522  =                         chk_olist:
5523  =                         ;---------
5524  =1DCD C606EF08FF                  mov chk_olist_flag,true
5525  =                         chk_olist0:
5526  =1DD2 A18F08                      mov ax,dcnt
```

```
5527
5528  =1DD5 A30909                      mov sdcnt,ax
5529  =1DD8 E815FFB503    1CF0          call pack_sdcnt| mov ch,3
5530  =1DDD E81CFF75DC    1CFC          call search_olist| jnz ret125
5531  =
5532  =                           ;     if in compatibility mode f1', r/o mode is the default open mode
5533  =                           ;     otherwise default is locked mode.  branch to openx06 if the open
5534  =                           ;     mode is not the default mode.
5535  =
5536  =1DE2 E8AAE8        068F          call get_cmp_mode
5537  =1DE5 D0C08A4702                  rol al,1| mov al,2[bx]
5538  =1DEA 7304          1DF0          jnb chk_olist01
5539  =1DEC D0C0D0C0                      rol al,1| rol al,1
5540  =                           chk_olist01:
5541  =1DF0 D0C0732F      1E23          rol al,1| jnb chk_olist05
5542  =1DF4 E8C8FF        1DBF            call compare_pds
5543  =1DF7 752A          1E23            jnz chk_olist05
5544  =
5545  =                           ;     verify another process does not have the file open when
5546  =                           ;     the default open mode is r/o as a result of compatibility
5547  =                           ;     attribute f1' = 1.
5548  =
5549  =1DF9 E893E8        068F              call get_cmp_mode
5550  =1DFC D0C0730C      1E0C              rol al,1| jnb chk_olist02
5551  =1E00 B503E801FF    1D06                mov ch,3| call search_nlist
5552  =1E05 741C          1E23                jz chk_olist05
5553  =1E07 B503E8F0FE    1CFC                mov ch,3| call search_olist
5554  =                           chk_olist02:
5555  =1E0C F606EF08FF74 1E33               test chk_olist_flag,true| jz ret11
5556        20            1E33
5557  =
5558  =                           ;     check to see if the lock list item is extended and if the interface
5559  =                           ;     attribute requesting that the lock item continue to be extended is
5560  =                           ;     set. if so, control is returned to the routine calling chk$olist and
5561  =                           ;     the lock item is retained.  otherwise, the lock item is deleted and
5562  =                           ;     pdcnt is incremented unless an extended lock was placed on the file.
5563  =
5564  =1E13 807F09FF                        cmp byte ptr 9[bx],0ffh
5565  =1E17 750D          1E26              jne chk_olist07
5566  =1E19 F606DE0880                      test attributes,080h
5567  =1E1E 7813          1E33              js ret11
5568  =1E20 E92DFF        1D50              jmp delete_item
5569  =
5570  =                           chk_olist05:
5571  =1E23 E92307        2549          jmp openx06
5572  =
5573  =                           chk_olist07:
5574  =
5575  =1E26 E827FF        1D50          call delete_item
5576  =
5577  =                           inc_pdcnt:
5578  =                           ;---------
5579  =1E29 A09D08                      mov al,pdcnt
```

```
5580
5581  =                         chk_olist1:
5582  =1E2C 041074FC      1E2C          add al,16! jz chk_olist1
5583  =1E30 A29D08                      mov pdcnt,al
5584  =1E33 C3                retl1:    ret
5585  =
5586  =                         pr_term:
5587  =                         ;-------
5588  =1E34 B80509                      mov ax,offset open_root
5589  =1E37 A3F708                      mov cur_pos,ax
5590  =                         pr_term1:
5591  =1E3A B500                        mov ch,0
5592  =1E3C E8C7FE        1D06          call search_nlist
5593  =1E3F 7510          1E51          jnz pr_term2
5594  =1E41 A1A408                        mov ax,pdaddr
5595  =1E44 394706                        cmp 6[bx],ax
5596  =1E47 75F1          1E3A            jnz pr_term1
5597  =1E49 E849FF        1D95            call remove_locks
5598  =1E4C E801FF        1D50            call delete_item
5599  =1E4F EBE9          1E3A            jmps pr_term1
5600  =                         pr_term2:
5601  =1E51 E87E0C        2AD2          call flush0
5602  =1E54 C3                retl2:    ret
5603  =
5604  =                         flush_files:
5605  =                         ;-----------
5606  =1E55 B0FF                        mov al,true
5607  =1E57 86060209                    xchg flushed,al
5608  =1E5B 84C075F5      1E54          test al,all jnz retl2
5609  =                         flush_file0:
5610  =1E5F B80509                      mov ax,offset open_root
5611  =1E62 A3F708                      mov cur_pos,ax
5612  =                         flush_file1:
5613  =1E65 1E07                        push ds! pop es
5614  =1E67 B501                        mov ch,1
5615  =1E69 E89AFE75E6    1D06          call search_nlist! jnz retl2
5616  =1E6E E824FF        1D95          call remove_locks
5617  =1E71 E8DCFE        1D50          call delete_item
5618  =1E74 8E062909                    mov es,uda_save
5619  =1E78 26F6061A0001 1E65           test u_pd_cnt,1! jnz flush_file1
5620        75E5          1E65
5621  =1E80 26800E1A0001                or u_pd_cnt,1
5622  =1E86 1E07                        push ds! pop es
5623  =1E88 3B1EA4087507 1E65           cmp bx,pdaddr! jnz flush_file1
5624  =1E8E E898FFEBD2    1E29          call inc_pdcnt! jmps flush_file1
5625  =
5626  =                         free_files:
5627  =                         ;----------
5628  =                         ;       free_mode = 1 - remove curdsk files for process
5629  =                         ;                   0 - remove all files for process
5630  =
5631  =1E93 B80509                      mov ax,offset open_root
5632  =1E96 A3F708                      mov cur_pos,ax
```

```
5633
5634  =                         free_files1:
5635  =1E99 8A2EF608                    mov ch,free_mode
5636  =1E9D E866FE        1D06          call search_nlist
5637  =1EA0 7582          1E54          jnz ret12
5638  =1EA2 A1A408                      mov ax,pdaddr
5639  =1EA5 394706                      cmp 6[bx],ax
5640  =1EA8 75EF          1E99          jnz free_files1
5641  =1EAA 837F03FF7506 1EB6           cmp word ptr 3[bx],0ffffh! jnz free_files2
5642  =1EB0 807F05FF7405 1EBB           cmp byte ptr 5[bx],0ffh! jz free_files3
5643  =                         free_files2:
5644  =1EB6 C606F508FF                  mov incr_pdcnt,true
5645  =                         free_files3:
5646  =1EBB E8D7FE        1D95          call remove_locks
5647  =1EBE E88FFE        1D50          call delete_item
5648  =1EC1 EBD6          1E99          jmps free_files1
5649  =
5650  =                         create_item:
5651  =                         ;-----------
5652  =                         ;       exit:   BX = offset of new item if successful
5653  =                         ;               Z flag set if no free items
5654  =
5655  =1EC3 8B1E5200                    mov bx,free_root
5656  =1EC7 0BDB7489      1E54          or bx,bx! je ret12
5657  =1ECB 891EF708                    mov cur_pos,bx                  ;free_root = free_root.link
5658  =1ECF 8B3789365200                mov si,[bx]! mov free_root,si
5659  =1ED5 8B36F908883C                mov si,prv_pos! mov di,[si]     ;create_item.link = prv_pos.link
5660  =1EDB 893F                        mov [bx],di                     ;prv_pos.link = .create_item
5661  =1EDD 891C                        mov [si],bx
5662  =1EDF C3                          ret
5663  =
5664  =                         create_olist_item:
5665  =                         ;-----------------
5666  =1EE0 8501E817FE    1CFC          mov ch,1! call search_olist
5667  =1EE5 E8D8FF        1EC3          call create_item
5668  =                         create_olist_item1:
5669  =1EE8 A0A008                      mov al,curdsk
5670  =1EEB 0A06DE08                    or al,attributes
5671  =1EEF 884702                      mov 2[bx],al                    ; set drive and attributes
5672  =1EF2 BEA108                      mov si,offset packed_dcnt
5673  =1EF5 8D7F03                      lea di,3[bx]
5674  =1EF8 B90500                      mov cx,5
5675  =1EFB F3A4                        rep movsb                       ;copy dcnt and pd_addr
5676  =1EFD 33C0AB                      xor ax,ax! stosw                ;open_cnt = 0;
5677  =1F00 8A0EA008                    mov cl,curdsk
5678  =1F04 40D3E0                      inc ax! shl ax,cl               ;set curdsk bit in
5679  =1F07 09068800                    or open_vector,ax               ; open file vector
5680  =1F0B C3                          ret
5681  =
5682  =                         create_llist_item:
5683  =                         ;-----------------
5684  =                         ;       exit:   BX = llist item address
5685  =                         ;               DI = offset l_item.rec
```

```
5686
5687  =
5688  =1F0C B501E8F0FD   1D01          mov ch,1! call search_llist
5689  =1F11 E8AFFF       1EC3          call create_item
5690  =1F14 8D7F03                     lea di,3[bx]
5691  =1F17 A1FE08                     mov ax,file_id
5692  =1F1A 894708                     mov 8[bx],ax
5693  =1F1D C3                         ret
5694  =
5695  =                          count_opens:
5696  =                          ;-----------
5697  =1F1E C606FC0800                 mov open_cnt,0
5698  =1F23 B80509                     mov ax,offset open_root
5699  =1F26 A3F708                     mov cur_pos,ax
5700  =                          count_open1:
5701  =1F29 B500E8D8FD   1D06          mov ch,0! call searchn_list
5702  =1F2E 750E         1F3E          jnz count_open2
5703  =1F30 A1A408                       mov ax,pdaddr
5704  =1F33 394706                       cmp 6[bx],ax
5705  =1F36 75F1         1F29            jnz count_open1
5706  =1F38 FE06FC08EBEB 1F29            inc open_cnt! jmps count_open1
5707  =                          count_open2:
5708  =1F3E A0FC08                     mov al,open_cnt
5709  =1F41 2AD58B00                   sub al,open_max
5710  =                          ret13:
5711  =1F45 C3                         ret
5712  =
5713  =                          check_free0:
5714  =                          check_free:     ;check for required # of entries in free list
5715  =                          ;-----------
5716  =                          ;       entry:  DL = required # of free entries
5717  =
5718  =1F46 BB5200                     mov bx,offset free_root
5719  =1F49 32F6                       xor dh,dh                      ;for (free_entries = 0;
5720  =                          check_free1:
5721  =1F4B 8B1F                       mov bx,[bx]
5722  =1F4D 0BDB7407     1F58          or bx,bx! jz check_free2       ; (next_free_item ~= end_of_flist) &&
5723  =1F51 FEC6                         inc dh                       ; (free_entries < required_#);
5724  =1F53 3AF272F4     1F48            cmp dh,dl! jb check_free1    ; free_entries++);
5725  =                          ret14:
5726  =1F57 C3                           ret
5727  =                          check_free2:
5728  =1F58 5B                         pop bx                         ;remove return address
5729  =1F59 B00E                       mov al,14                      ;(end_of_flist) return(error -
5730  =1F5B E950E5       04AE          jmp set_lret                   ; no room in sys lock list);
5731  =
5732  =                          move_arecord:   ;packed_dcnt = arecord
5733  =                          ;-------------
5734  =1F5E BA1709                     mov dx,offset arecord
5735  =1F61 BBA108                     mov bx,offset packed_dcnt
5736  =1F64 B103E979E5   04E2          mov cl,3! jmp move
5737  =
5738  =                          chk_rec:
```

```
5739
5740  =                       ;-------
5741  =                       ;       exit:   Z flag reset if unwritten data
5742  =
5743  =1F69 E8FFE5      056B         call index
5744  =1F6C 740C        1F7A         jz chk_rec1                    ;error - unwritten data
5745  =1F6E A01509                   mov al,vrecord
5746  =1F71 3A051309                 cmp al,rcount
5747  =1F75 7303        1F7A         jnb chk_rec1
5748  =1F77 32C0                       xor al,al
5749  =1F79 C3                         ret
5750  =                       chk_rec1:
5751  =1F7A 0C01                     or al,1
5752  =1F7C C3                       ret
5753  =
5754  =                       check_rec:
5755  =                       ;---------
5756  =1F7D 8926F009                 mov lock_sp,sp
5757  =1F81 C606F208FF               mov lock_shell,true
5758  =1F86 B1FF                     mov cl,true                    ;assume read operation for rseek
5759  =1F88 E871F5      14FC         call rseek
5760  =1F8B C606F20800               mov lock_shell,false
5761  =1F90 E85AE6      05ED         call getfcb
5762  =1F93 F6060D08FF               test lret,0ffh
5763  =1F98 7514        1FAE         jnz ret16
5764  =
5765  =                       ;       verify that the block has not been allocated by another
5766  =                       ;       process before returning error code 1 (reading unwritten data)
5767  =
5768  =1F9A E8CCFF      1F69         call chk_rec
5769  =1F9D 740F        1FAE         jz ret16
5770  =1F9F C606FB08FF               mov dont_close,true
5771  =1FA4 E842F6      15E9         call test_disk_fcb
5772  =1FA7 7406        1FAF         jz lock_err1                   ;error - unwritten data
5773  =1FA9 E8BDFF      1F69           call chk_rec
5774  =1FAC 7501        1FAF           jnz lock_err1
5775  =                       lock_err:
5776  =                       ;--------
5777  =                       ret16:
5778  =1FAE C3                         ret
5779  =                       lock_err1:
5780  =1FAF E9FAE4      04AC         jmp set_lret1
5781  =
5782  =                       cmp_recs:       ;compare l_item.rec(+l_item.cnt) and rand_rec(+mult_cnt)
5783  =                       ;---------
5784  =                       ;       entry:  SI = l_item.rec (1st record #) offset
5785  =                       ;               DI = rand_rec   (2nd record #) offset
5786  =                       ;               CL = 00h - no counts added
5787  =                       ;                  = 01h - add mult_cnt to rand_rec
5788  =                       ;                  = 02h - add l_item.cnt to l_item.rec
5789  =                       ;       exit:   Z,C flags set
5790  =                       ;               SI,DI = preserved
5791  =
```

```
5792
5793  =1FB2 8A1C8B64401              mov bl,[si]| mov ax,1[si]       ;AX,BL = l_item.rec
5794  =1FB7 8A3D8B5501               mov bh,[di]| mov dx,1[di]       ;DX,BH = rand_rec
5795  =1FBC F6C1027406    1FC7       test cl,02h| jz cmp_rec1
5796  =1FC1 025C03                     add bl,3[si]
5797  =1FC4 150000                     adc ax,0                       ;AX,BL = l_item.rec + l_item.cnt
5798  =                          cmp_rec1:
5799  =1FC7 F6C1017407    1FD3       test cl,01h| jz cmp_rec2
5800  =1FCC 023E9408                   add bh,mult_cnt
5801  =1FD0 83D200                     adc dx,0                       ;DX,BH = rand_rec + mult_cnt
5802  =                          cmp_rec2:
5803  =1FD3 3BC27502      1FD9       cmp ax,dx| jne cmp_rec3
5804  =1FD7 3ADF                       cmp bl,bh
5805  =                          cmp_rec3:
5806  =1FD9 C3                         ret
5807  =
5808  =                          srch_llist:     ;search lock list for matching item
5809  =                          ;---------
5810  =                          ;       exit:   AL = 0ffh - end of llist
5811  =                          ;                  = 0feh - incompatable lock type
5812  =                          ;                  = 001h - complete overlap
5813  =                          ;                  = 002h - subset overlap
5814  =                          ;                  = 004h - pre partial overlap
5815  =                          ;                  = 008h - post partial overlap
5816  =                          ;               BX = cur_pos  if AL ~= 0ffh
5817  =                          ;               DI = o_item.pdaddr  if AL ~= 0ffh
5818  =
5819  =1FDA B501E827FD    1D06       mov ch,1| call searchn_list     ;search for matching drive
5820  =1FDF 7403          1FE4       jz srch_ll                      ;while(not_end_of_llist) and
5821  =1FE1 B0FF                       mov al,0ffh
5822  =1FE3 C3                         ret
5823  =                          srch_ll:
5824  =1FE4 883EFE08                   mov di,file_id                  ; if (file_id.dcnt == o_item.dcnt);
5825  =1FE8 B90300                     mov cx,3
5826  =1FEB 8B7708                     mov si,8[bx]                    ;  SI = o_item
5827  =1FEE 03F103F9                   add si,cx| add di,cx
5828  =1FF2 F3A6                       repe cmpsb
5829  =1FF4 75E4          1FDA       jne srch_llist
5830  =
5831  =1FF6 8D7703                     lea si,3[bx]                    ;  SI = l_item.rec
5832  =1FF9 BF5D0B                     mov di,offset info_fcb+ranrec
5833  =
5834  =                                ;check for no overlap
5835  =1FFC B101E8B1FF    1FB2       mov cl,01h| call cmp_recs       ;if (l_item.rec >=
5836  =2001 7307          1FDA       jae srch_llist                  ;    rand_rec + mult_cnt) or
5837  =2003 B102E8AAFF    1FB2       mov cl,02h| call cmp_recs       ;   (l_item.rec + cnt <= rand_rec)
5838  =2008 76D0          1FDA       jbe srch_llist                  ; no_olap - search for next l_item
5839  =
5840  =                                ;check for complete overlap
5841  =200A B100E8A3FF    1FB2       mov cl,00h| call cmp_recs       ;else if (l_item.rec >= rand_rec)
5842  =200F 720F          2020       jnae srch_13
5843  =2011 B103E89CFF    1FB2         mov cl,03h| call cmp_recs     ; if (l_item.rec + cnt <=
5844  =2016 7704          201C         jnbe srch_12a                 ;     rand_rec + mult_cnt)
```

```
5845
5846  =2018 B001                          mov al,01h                  ;  srch_ret = complete_olap;
5847  =201A EB11          2020            jmps srch_14
5848  =                             srch_12a:                         ; else ;(l_item.rec + cnt >
5849  =201C B004                          mov al,04h                  ;       rand_rec + mult_cnt)
5850  =201E EB0D          2020            jmps srch_14                ;  srch_ret = pre_part_olap;
5851  =                             srch_13:
5852  =                                   ;check for subset overlap
5853  =2020 B103                          mov cl,03h                  ;else ;(l_item.rec < rand_rec)
5854  =2022 E88DFF        1FB2            call cmp_recs               ; if (l_item.rec + cnt >
5855  =2025 7604          202B            jna srch_13a                ;     rand_rec + mult_cnt)
5856  =2027 B002                          mov al,02h                  ;  srch_ret = subset_olap;
5857  =2029 EB02          2020            jmps srch_14
5858  =                             srch_13a:                         ; else ;(l_item.rec + cnt <=
5859  =                                                               ;       rand_rec + mult_cnt)
5860  =202B B008                          mov al,08h                  ;  srch_ret = post_part_olap;
5861  =                             srch_14:
5862  =                                   ;check for incompatable lock type
5863  =202D 8B1EF708                      mov bx,cur_pos              ;BX = l_item;
5864  =2031 8B7708                        mov si,8[bx]                ;SI = o_item;
5865  =2034 8B7C06                        mov di,6[si]                ;DI = o_item.pd_addr
5866  =2037 393EA408                      cmp pdaddr,di               ;if (o_item.pd_addr == pd_addr)
5867  =203B 740F          204C            je srch_17                  ; return(srch_ret);
5868  =
5869  =203D F6060E0880                    test attributes,80h
5870  =2042 7406          204A            jz srch_16                  ;if (l_item.attrib ~= paw_lock) or
5871  =2044 F6470280                      test byte ptr 2[bx],80h
5872  =2048 7502          204C            jnz srch_17                 ;   (request_type ~= paw_lock)
5873  =                             srch_16:
5874  =204A B0FE                          mov al,0feh                 ; return(incompatable_lock_type);
5875  =                             srch_17:
5876  =204C C3                            ret
5877  =
5878  =                             test_lock:      ;check llist for records with an incompatable lock type
5879  =                             ;---------
5880  =204D F6069E0880                    test high_ext,80h           ;was file opened in unlocked mode?
5881  =2052 7417          206B            jz ret_17                   ; no
5882  =2054 803E7A0809                    cmp fx_intrn,fxi22
5883  =2059 7303          205E            jnb rand_op
5884  =205B E81509        2973              call func36               ;set random record #
5885  =                             rand_op:
5886  =205E B80709                        mov ax,offset lock_root
5887  =2061 A3F708                        mov cur_pos,ax
5888  =                             test_lock1:
5889  =2064 E873FF        1FDA            call srch_llist             ;while(not_end_of_llist);
5890  =2067 3CFF7501      206C            cmp al,0ffh! jne test_lock2
5891  =                             ret17:
5892  =206B C3                            ret                         ; return(no errors)
5893  =                             test_lock2:
5894  =206C 3CFE                          cmp al,0feh                 ; if (incompatable_lock_type)
5895  =206E 7411          2081            je test_lock3               ;  return(error)
5896  =2070 F60600D9FF                    test rmf,true
5897  =2075 75ED          2064            jnz test_lock1              ; if (write) and
```

```
5898
5899  =2077 8B1EF708                    mov bx,cur_pos
5900  =207B F6470280                    test byte ptr 2[bx],80h       ;    (l_item.attrib == paw_lock)
5901  =207F 74E3          2064          jz test_lock1
5902  =                           test_lock3:
5903  =2081 5B                         pop bx                        ;  remove return address
5904  =2082 B008                       mov al,8                      ;  return(error -
5905  =2084 E927E4        04AE         jmp set_lret                  ;  locked by an other proccess);
5906  =
5907  =                           lock:           ;record lock or unlock
5908  =                           ;----
5909  =2087 B102                       mov cl,2
5910  =2089 E840FA        1ACC         call copy_dma_in
5911  =208C E8F0E4        057F         call get_atts                 ;get & clear atts f5' - f8'
5912  =208F E80CE6        069E         call check_fcb
5913  =2092 F6069E0880                 test high_ext,80h             ;if (open mode ~= unlocked)
5914  =2097 74D2          206B         jz ret17                      ; yes - return ok
5915  =
5916  =2099 8B1EC908                   mov bx,word ptr common_dma
5917  =209D A1A408                     mov ax,pdaddr
5918  =20A0 3947067405    20AA         cmp 6[bx],ax! jz lock01       ;if (olist.pd_addr ~= pdaddr)
5919  =20A5 B00DE904E4    04AE           mov al,13! jmp set_lret     ; yes - invalid file id
5920  =                           lock01:
5921  =20AA 891EFE08                   mov file_id,bx
5922  =
5923  =20AE F606F408FF                 test lock_unlock,true         ;if (~lock_unlock)
5924  =20B3 7508          20BF         jnz lock01a
5925  =20B5 F6060E0880                   test attributes,80h         ; if (attributes && 80h)
5926  =20BA 7403          20BF           jz lock01a
5927  =20BC E9D6FC        1D95             jmp remove_locks          ;  remove all locks
5928  =                           lock01a:
5929  =
5930  =20BF A05D08                     mov al,info_fcb+ranrec        ;if (fcb(ranrec) > 3ffffh)
5931  =20C2 8B1E5E08                   mov bx,word ptr info_fcb+ranrec+1
5932  =20C6 80FF04                     cmp bh,4                      ; return with lret = 6
5933  =20C9 7310          20DB         jae lock01b
5934  =20CB 8A269408                   mov ah,mult_cnt
5935  =20CF FECC                       dec ah
5936  =20D1 02C4                       add al,ah                     ;if fcb(ranrec)+mult_cnt > 3ffffh
5937  =20D3 83D300                     adc bx,0                      ; return with lret = 6
5938  =20D6 80FF04                     cmp bh,4
5939  =20D9 7206          20E1         jb lock01c
5940  =                           lock01b:
5941  =20DB C6060D0806                 mov lret,6                    ;error - random record #
5942  =20E0 C3                         ret                           ; out of range
5943  =                           lock01c:
5944  =                                ;mov olap_type,0              ;olap_type = no_olap;
5945  =
5946  =                                ; check llist for records with an incompatable lock type
5947  =20E1 B80709                     mov ax,offset lock_root
5948  =20E4 A3F708                     mov cur_pos,ax
5949  =                           lock02a:
5950  =20E7 E8F0FE        1FDA         call srch_llist               ;while(not_end_of_llist);
```

```
5951
5952  =20EA 3CFF741C      210A          cmp al,0ffh! je lock04a
5953  =20EE 3CFE750C      20FE            cmp al,0feh! jne lock03a      ; if (incompatable_lock_type)
5954  =20F2 F606F408FF                      test lock_unlock,true       ;  yes - if (lock)
5955  =20F7 74EE          20E7              jz lock02a
5956  =20F9 B008E9B0E3    04AE              mov al,8! jmp set_lret      ;   yes - return(error -
5957  =                                                                 ;   locked by an other proccess);
5958  =                         lock03a:                                ; else
5959  =20FE 393EA408                        cmp pdaddr,di               ; if (o_item.pdaddr == pdaddr)
5960  =2102 75E3          20E7              jne lock02a
5961  =2104 080E1908                        or olap_type,al             ;  olap_type |= llist_ret;
5962  =2108 EBDD          20E7              jmps lock02a
5963  =                         lock04a:
5964  =                                 ;determine number of lock entries required by overlap type
5965  =210A BBDF08                      mov bx,offset required_table
5966  =210D A01908                      mov al,olap_type
5967  =2110 D7                          xlat required_table             ;AL = required_table[olap_type];
5968  =2111 F606F408FF                  test lock_unlock,true
5969  =2116 7506          211E          jnz lock05a                     ;if (unlock) &&
5970  =2118 0AC07402      211E          or al,al! jz lock05a            ;   (required_# ~= 0)
5971  =211C FEC8                          dec al                        ; required_#--;
5972  =                         lock05a:
5973  =                                 ;check if max entries / proccess will be exceeded
5974  =211E 8B16FE08                    mov dx,file_id
5975  =2122 C706F7080709                mov cur_pos,offset lock_root
5976  =2128 32E4                        xor ah,ah                       ;lock_cnt = 0;
5977  =                         count_lock1:
5978  =212A B500                        mov ch,0
5979  =212C 5052                        push ax! push dx
5980  =212E E8D5FB        1D06          call searchn_list               ;get next llist item
5981  =2131 5A58                        pop dx! pop ax
5982  =2133 7509          213E          jnz count_lock2                 ;while(not_end_of_llist)
5983  =2135 395708                        cmp 8[bx],dx
5984  =2138 75F0          212A            jnz count_lock1               ; if (l_item.olist == file_id)
5985  =213A FEC4                          inc ah                        ;  lock_cnt++;
5986  =213C EBEC          212A            jmps count_lock1
5987  =                         count_lock2:
5988  =213E 8AD0                        mov dl,al                       ;DL = required #, for check_free
5989  =2140 02C47206      214A          add al,ah! jb lock02            ;if (lock_max <
5990  =2144 38068A007305  214F          cmp lock_max,al! jae lock05     ;   (lock_cnt + required_#))
5991  =                         lock02:
5992  =214A B00CE95FE3    04AE            mov al,12! jmp set_lret       ; return(error - lock limit exceeded);
5993  =                         lock05:
5994  =                                 ;check for required # of entries in free list
5995  =214F E8F4FD        1F46         call check_free
5996  =
5997  =                                 ;check for record in file if required
5998  =2152 F606F408FF                  test lock_unlock,true
5999  =2157 7423          217C          jz lock4                        ;if (lock) &&
6000  =2159 F606DE0840                  test attributes,40h             ; (lock_type == rec_required_in_file)
6001  =215E 751C          217C          jnz lock4
6002  =2160 8A269408                      mov ah,mult_cnt
6003  =                         lock2:                                  ;for (i = 0; i < mult_cnt; i++);
```

```
6004
6005  =2164 50                          push ax
6006  =2165 E815FE      1F70            call check_rec              ; check_for_record(rand_rec + i);
6007  =2168 58                          pop ax
6008  =2169 F6050D08FF                  test lret,0ffh              ; if (record_not_in_file)
6009  =216E 7403        2173            jz $+5
6010  =2170 E95BE2      03CE              jmp reset_rr              ;  return(error - unwritten data);
6011  =2173 FECC7405    217C            dec ah! jz lock4
6012  =2177 E886E2EBE8  0400              call incr_rr! jmps lock2
6013  =                       lock4:
6014  =217C E84FE2      03CE          call reset_rr
6015  =
6016  =                             ;remove over lapping records from lock list
6017  =217F F60619080F                test olap_type,00Fh         ;if (olap_type ~= no_olap)
6018  =2184 7503        2189          jnz $+5
6019  =2186 E98B00      2214          jmp lock08a
6020  =2189 B80709                      mov ax,offset lock_root
6021  =218C A3F708                      mov cur_pos,ax
6022  =                       lock06a:
6023  =218F E848FE      1FDA            call srch_llist
6024  =2192 3CFF7503    2199            cmp al,0ffh! jne $+5        ; while(not_end_of_llist);
6025  =2196 E97B00      2214            jmp lock08a
6026  =2199 393EA408                    cmp pdaddr,di               ; if (o_item.pdaddr == pdaddr)
6027  =219D 75F0        218F            jne lock06a
6028  =219F A801                       test al,01h                 ;  if (llist_ret && complete_olap)
6029  =21A1 7405        21A8            jz lock07a
6030  =21A3 E8AAFB      1D50              call delete_item          ;   delete(cur_item);
6031  =21A6 EBE7        218F              jmps lock06a
6032  =                       lock07a:                              ;  else
6033  =21A8 A804                        test al,04h                 ;  if (llist_ret && pre_part_olap)
6034  =21AA 7421        21CD            jz lock07b
6035  =21AC 8A6F03                      mov ch,3[bx]                ;   l_item.cnt = (l_item.rec+cnt) -
6036  =21AF 026F06                      add ch,6[bx]                ;                (rand_rec+mult_cnt);
6037  =21B2 8A0E5D08                    mov cl,info_fcb+ranrec      ;   CH = low(l_item.rec + l_item.cnt);
6038  =21B6 A15E08                      mov ax,word ptr info_fcb+ranrec+1
6039  =21B9 020E9408                    add cl,mult_cnt
6040  =21BD 150000                      adc ax,0                    ;   AX,CL = rand_rec + mult_cnt
6041  =21C0 2AE9                        sub ch,cl
6042  =21C2 886F06                      mov 6[bx],ch                ;   store l_item.cnt
6043  =
6044  =21C5 884F03                      mov 3[bx],cl                ;   l_item.rec = rand_rec + mult_cnt;
6045  =21C8 894704                      mov 4[bx],ax                ;   store l_item.rec
6046  =21CB EBC2        218F            jmps lock06a
6047  =                       lock07b:                              ;  else
6048  =21CD A808                        test al,08h                 ;  if (llist_ret && post_part_olap)
6049  =21CF 740C        21DD            jz lock07c
6050  =21D1 8A0E5D08                    mov cl,info_fcb+ranrec      ;   l_item.cnt = rand_rec-l_item.rec;
6051  =21D5 2A4F03                      sub cl,3[bx]                ;   CL = low(rand_rec - l_item.rec);
6052  =21D8 884F06                      mov 6[bx],cl                ;   store l_item.cnt
6053  =21DB EBB2        218F            jmps lock06a
6054  =                       lock07c:                              ;  else ;(llist_ret && subset_olap)
6055  =21DD 53                          push bx                     ;   save current item address
6056  =21DE E82BFD      1F0C            call create_llist_item      ;   create new llist item
```

```
6057
6058  =21E1 5E                          pop si
6059  =21E2 8A4402884702                mov al,2[si]! mov 2[bx],al    ;   set attributes and drive
6060  =21E8 8A0E5D08                    mov cl,info_fcb+ranrec        ;   rec = (rand_rec + mult_cnt)
6061  =21EC A15E08                      mov ax,word ptr info_fcb+ranrec+1
6062  =21EF 020E9408                    add cl,mult_cnt
6063  =21F3 150000                      adc ax,0                      ;   AX,CL = rand_rec + mult_cnt
6064  =21F6 884F03                      mov 3[bx],cl
6065  =21F9 894704                      mov 4[bx],ax                  ;   store new l_item.rec
6066  =
6067  =21FC 8A6C03026C06               mov ch,3[si]! add ch,6[si]    ;   cnt = (l_item + cnt) -
6068  =2202 2AE9                        sub ch,cl                     ;         (rand_rec+mult_cnt)
6069  =2204 886F06                      mov 6[bx],ch                  ;   store new l_item.cnt
6070  =
6071  =2207 8A0E5D08                    mov cl,info_fcb+ranrec        ;   l_item.cnt = rand_rec -
6072  =220B 2A4C03                      sub cl,3[si]                  ;                l_item.rec;
6073  =220E 884C06                      mov 6[si],cl                  ;   store l_item.cnt
6074  =2211 E978FF        218F          jmp lock06a
6075  =                          lock08a:                             ; end of while
6076  =2214 F606F408FF                 test lock_unlock,true         ;if (lock)
6077  =2219 7418          2236         jz lock09a
6078  =221B E8EEFC        1F0C           call create_llist_item       ; create(rand_rec,mult_cnt);
6079  =221E A0A008                      mov al,cur_dsk
6080  =2221 0A05DE08                    or al,attributes
6081  =2225 884702                      mov 2[bx],al                  ; set drive and attributes
6082  =2228 BE5D08                      mov si,offset info_fcb+ranrec
6083  =222B B90300                      mov cx,3
6084  =222E F3A4                        rep movsb                     ; copy record #
6085  =2230 A09408                      mov al,mult_cnt
6086  =2233 884706                      mov 6[bx],al                  ; l_item.cnt = mult_cnt;
6087  =                          lock09a:
6088  =2236 C3                         ret
6089  =
6090  =                          fix_olist_item:
6091  =                          ;--------------
6092  =2237 A11508                     mov ax,xdcnt
6093  =223A 3B060909                   cmp ax,sdcnt             ; is xdcnt < sdcnt?
6094  =223E 7363          22A3         jae ret18                ; no
6095  =                                ;yes - update olist entries
6096  =2240 E8C7ED        100A         call swap
6097  =2243 A11508                     mov ax,xdcnt
6098  =2246 A30909                     mov sdcnt,ax
6099  =2249 B80509                     mov ax,offset open_root
6100  =224C A3F708                     mov cur_pos,ax
6101  =                                ;find file's olist entries & change dcnt values
6102  =                          fix_ol1:
6103  =224F E8B8ED        100A         call swap
6104  =2252 E89BFA        1CF0         call pack_sdcnt
6105  =2255 E8B2ED        100A         call swap
6106  =2258 B503                       mov ch,3
6107  =225A E8A9FA7544    1006         call searchn_list! jnz ret18
6108  =                          ;              update olist item with new dcnt value
6109  =225F 53E88DFA58    1CF0         push bx! call pack_sdcnt! pop bx
```

```
6110
6111  =2264 83C303                      add bx,3
6112  =2267 BAA108                      mov dx,offset packed_dcnt
6113  =226A B103                        mov cl,3
6114  =226C E873E2        04E2          call move
6115  =226F EBDE          224F          jmps fix_oll
6116  =
6117  =                           diskreset:
6118  =                           ;---------
6119  =                           ;       exit:   Z flag set if disk reset is allowed
6120  =
6121  =2271 33C0                        xor ax,ax
6122  =2273 A30408                      mov ntlog,ax
6123  =2276 A20009                      mov set_ro_flag,al
6124  =2279 8B1E8108                    mov bx,info
6125  =                           intrnl_disk_reset:
6126  =227D 833E050900                  cmp open_root,0
6127  =2282 741F          22A3          jz ret18
6128  =2284 A0A00850                    mov al,curdskl push ax
6129  =2288 B500                        mov ch,0
6130  =                           dskrst1:
6131  =228A D1EB7216      22A4          shr bx,1l jb dskrst3
6132  =                           dskrst2:
6133  =228E FEC5                          inc ch
6134  =2290 83FB0075F5    228A           cmp bx,01 jnz dskrst1
6135  =2295 58A2A008                      pop axl mov curdsk,al
6136  =2299 8B1E0408                      mov bx,ntlog
6137  =229D 091E0208                      or tlog,bx
6138  =22A1 FEC0                          inc al
6139  =                           ret18:
6140  =22A3 C3                            ret
6141  =                           dskrst3:
6142  =22A4 5153                        push cxl push bx
6143  =22A6 882EA0088ACD                mov curdsk,chl mov cl,ch
6144  =22AC 8B160208                    mov dx,tlog
6145  =22B0 E86BE452      071E          call test_vectorll push dx
6146  =22B4 8B160008                    mov dx,rlog
6147  =22B8 E853E452      071E          call test_vectorll push dx
6148  =22BC 8B160608                    mov dx,rodsk
6149  =22C0 E858E458      071E          call test_vectorll pop bx
6150  =22C4 0A1E0009                    or bl,set_ro_flag
6151  =22C8 0ADA5A                      or bl,dll pop dx
6152  =22CB 0ADA881E0109                or bl,dll mov check_disk,bl
6153  =22D1 BB0509                      mov bx,offset open_root
6154  =22D4 891EF708                    mov cur_pos,bx
6155  =                           dskrst4:
6156  =22D8 B501                        mov ch,1
6157  =22DA E829FA7523    1D06          call search_nlistl jnz dskrst6
6158  =22DF F6060109FF                  test check_disk,true
6159  =22E4 7414          22FA          jz dskrst5
6160  =22E6 53                          push bx
6161  =22E7 E805FA7405    1DBF          call compare_pdsl jz dskrst45
6162  =22EC 5A                          pop dx
```

```
6163
6164  =22ED 32C0EB11      2302          xor al,al! jmps dskrst6
6165  =                           dskrst45:
6166  =22F1 BB0408                      mov bx,offset ntlog
6167  =22F4 E813E4        070A          call set_cdisk
6168  =22F7 5BEBDE        2208          pop bx! jmps dskrst4
6169  =                           dskrst5:
6170  =22FA BB8108                      mov bx,offset info
6171  =22FD E887FA        1D87          call remove_drive
6172  =2300 0C01                        or al,1
6173  =                           dskrst6:
6174  =2302 5B59                        pop bx! pop cx
6175  =2304 7403          2309          jz $+5
6176  =2306 E985FF        228E            jmp dskrst2
6177  =
6178  =                           ;       error - olist item exists for another process
6179  =                           ;               on a removeable or read/only drive to be reset
6180  =                           ;               or any kind of drive if function is write protect disk
6181  =
6182  =2309 58A2A008                    pop ax! mov curdsk,al
6183  =230D 803E9308FF                  cmp error_mode,0ffh
6184  =2312 7411          2325          je dskrst7
6185  =2314 882E2F09                      mov err_drv,ch
6186  =2318 8BDA8B4706                    mov bx,dx! mov ax,6[bx]
6187  =231D A33009                        mov err_pd_addr,ax
6188  =2320 C6061B08FF                    mov err_type,0ffh
6189  =                           dskrst7:
6190  =2325 58C7060D08FF                pop bx! mov aret,0ffffh
6191        FF
6192  =232C C3                          ret
6193  =                           endif
6194  =
6195  =                           ;********** end bdos file system part 3 **********
6196
```

```
6197
6198  =                         eject 1 include file4.bdo       ; file system part 4
6199  =
6200  =                    ;************ bdos file system part 4 ************
6201  =
6202  =                    ;       individual function handlers
6203  =                    ;-----------------------------------------------------
6204  =
6205  =                    func13:         ;reset disk system
6206  =                    ;======
6207  =                    ;       test all drives, not just logged in drives, because
6208  =                    ;       some drives may be in tlog state.
6209  =
6210  =2320 B8FFFF               mov ax,0ffffh
6211  =2330 A38108               mov info,ax
6212  =
6213  =                    if BMPM
6214  =2333 E83BFF       2271    call diskreset
6215  =2336 7405         2330    jz reset_all
6216  =2338 E84B06       2986      call reset_37
6217  =233B EB0E         2348      jmps func13_cont
6218  =                    reset_all:
6219  =
6220  =                    endif
6221  =                    if BCPM
6222  =                         call reset_37x
6223  =                    endif
6224  =
6225  =233D 33C0                 xor ax,ax
6226  =233F A30608               mov rodsk,ax
6227  =2342 A30808               mov dlog,ax
6228  =
6229  =                    if BMPM
6230  =2345 A30008               mov rlog,ax
6231  =2348 A30208               mov tlog,ax
6232  =                    func13_cont:
6233  =
6234  =                    endif
6235  =
6236  =234B 32C0                 xor al,al
6237  =234D E81A00       236A    call func14a                    ;set default disk to A:
6238  =2350 FEC8                 dec al
6239  =2352 A2A008               mov curdsk,al
6240  =                    if BMPM
6241  =2355 1E8E1E2909           push ds! mov ds,uda_save
6242  =235A C706020080000        mov ds:u_dma_ofst,080h          ;set dma offset to 80h
6243  =2360 1F                   pop ds
6244  =                    endif
6245  =                    if BCPM
6246  =                         mov u_dma_ofst,080h             ;set dma offset to 80h
6247  =                    endif
6248  =2361 E95F07       2AC3    jmp func48                      ;flush buffers
6249  =
```

```
6250
6251  =                         func14:         ;select disk info
6252  =                         ;======
6253  =2364 E848F6        1982          call tmp_select
6254  =2367 A07F08                      mov al,seldsk
6255  =                         func14a:
6256  =
6257  =                         if BMPM
6258  =236A 8B1E6800                    mov bx,rlr
6259  =236E 884712                      mov p_dsk[bx],al
6260  =                         endif
6261  =                         if BCPM
6262  =                                 mov p_dsk,al
6263  =                         endif
6264  =
6265  =2371 C3                          ret
6266  =
6267  =                         func15:         ;open file
6268  =                         ;======
6269  =2372 E855F7        1ACA          call copy_dma_in_8
6270  =2375 E8D9E3        0751          call clrmodnum                  ;clear the module number
6271  =
6272  =                         if BMPM
6273  =2378 C606D90800                  mov make_flag,false
6274  =237D C6060A09FF                  mov byte ptr sdcnt+1,0ffh       ;initialize sdcnt
6275  =2382 BB9B23                      mov bx,offset openfile          ;check_fcb1 returns to open_file
6276  =2385 53                          push bx                         ;if checksum invalid
6277  =2386 C606F308FF                  mov check_fcb_ret,true
6278  =238B E815E3        06A3          call check_fcb1                 ;returns normally if fcb cksum valid
6279  =238E 5B                          pop bx
6280  =238F 803E9E0860                  cmp high_ext,60h                ;is file open in read/only-user 0 mode?
6281  =2394 7505          239B          jne open_file                   ;no
6282  =2396 E8E3E3        077C            call set_end_dir
6283  =2399 EB65          2400            jmps open_user_zero
6284  =                         open_file:
6285  =239B C6059F0800                  mov xfcb_read_only,0            ;clear in case set by calling process
6286  =23A0 C6050A09FF                  mov byte ptr sdcnt+1,0ffh       ;initialize sdcnt
6287  =23A5 E853E3        06FB          call reset_checksum_fcb         ;invalidate fcb chksum
6288  =                         endif
6289  =
6290  =23A8 E880EA        0E2B          call check_wild                 ;check for ?s in fcb
6291  =
6292  =                         if BMPM
6293  =23AB E8D1E1        057F          call get_atts                   ;get & clear atts f5'-f8'
6294  =23AE 24C0                        and al,0c0h                     ;lst 2 specify open mode
6295  =23B0 3CC0                        cmp al,0c0h                     ;if both on default to read/only
6296  =23B2 7502          23B6          jne att_ok
6297  =23B4 2440                          and al,40h                    ;remove unlocked mode bit
6298  =                         att_ok:
6299  =23B6 A29E08                      mov high_ext,al                 ;save open mode
6300  =23B9 8AE0                        mov ah,al                       ;define attributes
6301  =23BB D0EC                        shr ah,1                        ;bit 0 on = locked mode
6302  =23BD 7502          23C1          jnz att_set                     ;bit 1 on = unlocked mode
```

```
6303
6304  =23BF B480                        mov ah,80h                  ;bit 2 on = read/only mode
6305  =                           att_set:
6306  =23C1 88260E08                    mov attributes,ah
6307  =23C5 2480                        and al,80h                  ;is open mode unlocked?
6308  =23C7 7514          23DD          jnz call_open               ;yes
6309  =                           endif
6310  =
6311  =23C9 803E800800                  cmp usrcode,0               ;is user 0
6312  =23CE 740D          23DD          je call_open                ;yes
6313  =23D0 B0FE                          mov al,0feh               ;initialize search switches to
6314  =23D2 A21608                        mov byte ptr xdcnt+1,al   ;flag presence of file under user
6315  =23D5 FEC0                          inc al                    ;zero if file not present under
6316  =23D7 A21208                        mov search_user0,al       ;current user #
6317  =
6318  =                           if BMPH
6319  =23DA A20C09                        mov byte ptr sdcnt0+1,al  ;initialize user 0 sdcnt
6320  =                           endif
6321  =
6322  =                           call_open:
6323  =23DD E89DEE        1270          call open                   ;attempt to open fcb
6324  =23E0 E83100        2414          call openx                  ;returns if unsuccessful
6325  =23E3 32C0                        xor al,al
6326  =23E5 86061208                    xchg search_user0,al        ;search_user0 = false
6327  =23E9 84C0                        test al,al                  ;is search_user0 true?
6328  =23EB 7501          23EE          jnz $+3                     ;yes
6329  =23ED C3                    ret30:  ret                       ;no - open failed
6330  =23EE 803E1608FE                  cmp byte ptr xdcnt+1,0feh   ;does file exist under user 0?
6331  =23F3 74F8          23ED          je ret30                    ;no
6332  =
6333  =                           if BMPH
6334  =23F5 E812EC        100A          call swap                   ;swap sdcnt & sdcnt0
6335  =                           endif
6336  =
6337  =23F8 E888E3        0783          call set_dcnt               ;reset dcnt
6338  =23FB C6069E0860                  mov high_ext,60h            ;set open mode to read/on-user 0
6339  =                           open_user_zero:
6340  =2400 32C0                        xor al,al                   ;fcb(0) = user 0
6341  =2402 A23C08                      mov info_fcb,al
6342  =2405 B10F                        mov cl,namlen               ;attempt to locate fcb under user zero
6343  =2407 E819EC        1023          call searchi
6344  =240A E83EEC        1048          call searchn
6345  =240D E870EE        1280          call open1
6346  =2410 E80100        2414          call openx                  ;openx returns if search unsuccessful
6347  =2413 C3                          ret
6348  =
6349  =                           openx:
6350  =                           ;-----
6351  =2414 E85EE3        0775          call endofdir               ;was open successful
6352  =2417 74D4          23ED          jz ret30                    ;no
6353  =2419 BB5C08                      mov bx,offset info_fcb+nxtrec
6354  =241C 803FFF                      cmp b[bx],0ffh
6355  =241F 7505          2426          jne openxa
```

```
6356
6357  =2421 A0490B                      mov al,info_fcb+chksum
6358  =2424 8807                        mov [bx],al
6359  =                           openxa:
6360  =2426 5B                          pop bx                      ;discard return address
6361  =2427 A09E08                      mov al,high_ext             ;is open mode read/only - user 0?
6362  =242A 3C60                        cmp al,60h
6363  =
6364  =                           if BCPM
6365  =                                 jne openx0                  ;no
6366  =                           endif
6367  =                           if BMPM
6368  =242C 7417          2445          je openx00                  ;yes
6369  =242E 0AC0                        or al,al                    ;is open mode locked?
6370  =2430 7525          2457          jnz openx0                  ;no
6371  =2432 BB3C08                      mov bx,offset info_fcb      ;set BX for rotest
6372  =2435 0A07                        or al,[bx]                  ;is user number zero?
6373  =2437 751E          2457          jnz openx0                  ;no
6374  =2439 E8F4E2        0730          call rotest                 ;is file read/only attribute set?
6375  =243C 7319          2457          jnc openx0                  ;no
6376  =243E 43                          inc bx                      ;is file system attribute set?
6377  =243F 8A07                        mov al,[bx]
6378  =2441 D0D0                        rcl al,1
6379  =2443 7312          2457          jnc openx0                  ;no
6380  =
6381  =                           openx00:
6382  =2445 C606DE0820                  mov attributes,20h          ;open file in read/only-user 0 mode
6383  =                           endif
6384  =
6385  =244A A0460B                      mov al,info_fcb+sysfil      ;if fcb has system attribute set
6386  =244D 2480                        and al,80h
6387  =244F 7506          2457          jnz openx0                  ;system attribute set
6388  =2451 A29E08                        mov high_ext,al           ;open failed - system attribute reset
6389  =2454 E958EC        10AF            jmp lret_eq_ff
6390  =
6391  =                           openx0:
6392  =
6393  =                           if BMPM
6394  =2457 E8A1E2        06FB          call reset_checksum_fcb     ;invalidate checksum
6395  =                           endif
6396  =                           if 3CPM
6397  =                                 call set_lsn
6398  =                           endif
6399  =
6400  =245A E881F6        1ADE          call get_dir_mode           ;al = dir label data byte
6401  =245D A880                        TEST AL,80H                 ;ARE PASSWORDS ACTIVE?
6402  =245F 745C          248D          JZ OPENX1A                  ;NO
6403  =2461 E80AF8        1C6E            CALL QDIRFCB1             ;IS FCB IN 1ST DIR FCB?
6404  =2464 752C          2492            JNZ OPENX0A               ;NO
6405  =2466 E8DAF7        1C43            call get_dtba8
6406  =2469 0AC0                          OR AL,AL                  ;DOES SFCB EXIST?
6407  =246B 7525          2492            JNZ OPENX0A               ;NO
6408  =246D F607C0                          TEST BC[BX],0C0H        ;IS PASSWORD MODE READ OR WRITE?
```

```
6409
6410  =2470 7448        248D            JZ OPENX1A                      ;NO
6411  =2472 E801ED      1176            call xdcnt_eq_dcnt
6412  =2475 E86DF6      1AE5            call get_xfcb                   ;does xfcb exist
6413  =2478 7522        249C            jnz openx0b                     ; yes
6414  =247A E800ED      117D            call restore_dir_fcb
6415  =247D 7433        24BC            jz ret32
6416  =247F E8C3F7      1C45            call get_dtba                   ;does sfcb still exist
6417  =2482 0AC07537    248D            or al,al! jnz openx1a           ; no - (error)
6418  =2486 8807                        mov [bx],al                     ;sfcb.pwmode = 0
6419  =2488 E888E2      0716            call nowrite                    ;update sfcb
6420  =248B 7530        248D            jnz openx1a
6421  =248D E87DE7      0C0D            call wrdir
6422  =2490 EB28        248D            jmps openx1a
6423  =                         OPENX0A:
6424  =2492 E8E1EC      1176          CALL XDCNT_EQ_DCNT                ;SAVE DCNT
6425  =2495 E84DF6      1AE5          call get_xfcb
6426  =2498 24C0                      and al,0c0h                       ;is pw mode read(0) or write(1)?
6427  =249A 741B        24B7          jz openx1                         ;no - don't chk password
6428  =                         openx0b:
6429  =249C E82DF7      1BCC            call cmp_pw                     ;check passwords
6430  =249F 7416        24B7            jz openx1                       ;jump if password checks
6431  =24A1 E88EF6      1832            call chk_pw_error
6432  =24A4 A0DD08                      mov al,pw_mode
6433  =24A7 24C0                        and al,0c0h
6434  =24A9 740C        24B7            jz openx1
6435  =24AB A880                        test al,80h                     ;is pw mode read(1)?
6436  =24AD 7403        24B2            jz $+5                          ; yes
6437  =24AF E9E6F6      1B98              jmp pw_error                  ;password error
6438  =24B2 C6059F0880                  mov xfcb_read_only,80h          ;prohibit any writes to file
6439  =                         openx1:
6440  =24B7 E8C3EC      117D          CALL RESTORE_DIR_FCB
6441  =24BA 7501        24BD          JNZ OPENX1A
6442  =24BC C3                  RET32:    RET                           ; error
6443  =
6444  =                         OPENX1A:
6445  =
6446  =                         if BMPM
6447  =24BD E830F8      1CF0        call pack_sdcnt
6448  =
6449  =                             ;       set open mode to r/o if in default mode and
6450  =                             ;       if compatibility attribute f1' is set.
6451  =
6452  =24C0 F606DE08A0                  test attributes,0a0h
6453  =24C5 7411        24D8            jz openx101
6454  =24C7 E8C5E1      068F              call get_cmp_mode
6455  =24CA D0C0730A    24D8              rol al,1! jnb openx101
6456  =24CE C606190880                    mov olap_type,80h             ;set f1' locked overlap switch
6457  =24D3 C606DE0820                    mov attributes,20h
6458  =                         openx101:
6459  =24D8 B503                        mov ch,3
6460  =24DA E81FF8741B  1CFC            call search_olist! jz openx04   ;is this file currently open?
6461  =                         openx01:
```

```
6462
6463  =24DF 833E520000                  cmp free_root,0
6464  =24E4 7505          24EB          jne openx03                 ;no - is olist full?
6465  =                           openx02:
6466  =24E6 B408                        mov ah,11
6467  =24E8 E9C9DF        04B4          jmp set_aret                ; yes - error
6468  =                           openx03:
6469  =24EB E830FA        1F1E          call count_opens            ;has process exceeded
6470  =24EE 7205          24F5          jb openx035                 ; open file maximum?
6471  =                           openx034:
6472  =24F0 B40A                        mov ah,10
6473  =24F2 E98FDF        04B4          jmp set_aret                ; yes - error
6474  =                           openx035:
6475  =24F5 E8E8F9        1EE0          call create_olist_item      ;create new olist element
6476  =24F8 EB5B          2555          jmps openx08
6477  =                           openx04:
6478  =24FA 837F08FF                    cmp word ptr 8[bx],0ffffh   ;if file has extended lock,
6479  =24FE 750A          250A          jne openx041                ; allow change in open mode
6480  =2500 E8BCF8        1D8F            call compare_pds
6481  =2503 7544          2549            jne openx06
6482  =2505 E8E0F9        1EE8            call create_olist_item1
6483  =2508 EB4B          2555            jmps openx08
6484  =                           openx041:
6485  =250A F6059E0880                  test attributes,80h         ;is open mode locked?
6486  =250F 7415          2526          jz openx042                 ;no
6487  =2511 F6470580                    test byte ptr 5[bx],80h     ;does ll.item indicate fl' env?
6488  =2515 740F          2526          jz openx042                 ;no
6489  =2517 C6069F0880                    mov xfcb_read_only,80h    ;treat as write password error
6490  =251C C6069E0820                    mov attributes,20h        ;change open mode to read/only
6491  =2521 C606190880                    mov olap_type,80h         ;set locked overlap switch
6492  =                           openx042:
6493  =2526 A09E080A4702                mov al,attributes! or al,2[bx]
6494  =252C 3A47027518    2549          cmp al,2[bx]! jnz openx06   ;do file attributes match?
6495  =2531 24807519      254E            and al,80h! jnz openx07   ; yes - is open mode locked?
6496  =2535 8B1EF908                      mov bx,prv_pos            ; no
6497  =2539 891EF708                      mov cur_pos,bx            ;has this file been
6498  =253D B505                          mov ch,5                  ; opened by this process?
6499  =253F E8C4F7759B    1D06            call search_nlist! jnz openx01
6500  =                           openx05:
6501  =2544 FF4708                        inc word ptr 8[bx]        ; yes - increment open file count
6502  =2547 EB0C          2555            jmps openx08
6503  =                           openx06:
6504  =                           ;       error - file opened by another process in
6505  =                           ;               incompatible mode
6506  =
6507  =2549 B405                       mov ah,5
6508  =254B E966DF        04B4          jmp set_aret
6509  =
6510  =                           openx07:        ;does this olist item belong to this process?
6511  =254E E86EF8        1DBF          call compare_pds
6512  =2551 75F6          2549          jnz openx06             ; no - error
6513  =2553 EBEF          2544          jmps openx05            ; yes
6514  =                           openx08:        ;open successful was file opened in unlocked mode?
```

```
6515
6516  =2555 F606DE0840          test attributes,40h
6517  =255A 7408        2567    jz openx09                ; no
6518  =255C A1F708                mov ax,cur_pos          ;return .olist_item in ranrec field of fcb
6519  =255F A35D0B                mov word ptr info_fcb+ranrec,ax
6520  =2562 C6060C0823            mov parlg,35            ;copy 35 bytes back to user fcb
6521  =                     openx09:
6522  =2567 F606DE0820          test attributes,20h       ;is open mode read/only?
6523  =256C 741B        2589    jz openx09b               ;no
6524  =256E F605190880            test olap_type,80h      ;fl' locked overlap case?
6525  =2573 7414        2589      jz openx09b             ;no
6526  =2575 B80509                  mov ax,offset open_root     ;set fl' env bit in all
6527  =2578 A3F708                  mov curpos,ax               ;lock items for file
6528  =                     openx09a:
6529  =257B B503                    mov ch,3
6530  =257D E886F77507  1D06        call search_nlist1 jnz openx09b
6531  =2582 804F0580                  or byte ptr 5[bx],80h     ;set fl' env bit
6532  =2586 E9F2FF      2579          jmp openx09a
6533  =                     openx09b:
6534  =2589 C6061108FF          mov comp_fcb_cks,true
6535  =258E F6060D08FF          test make_flag,true
6536  =2593 7546        25DB    jnz ret34
6537  =                     endif
6538  =                     if BDOS
6539  =                         mov comp_fcb_cks,true
6540  =                     endif
6541  =
6542  =2595 B140                MOV CL,40H                  ;is access stamping requested
6543  =                     openx2:                         ; on drive
6544  =2597 E8EAF6      1C84   CALL QSTAMP
6545  =259A 753F        25DB   JNZ RET34                   ; no - stamping requested
6546  =259C E9F7F6      1C96   JMP STAMP1
6547  =
6548  =                     func16:         ;close file
6549  =                     ;======
6550  =
6551  =                     if BNPM
6552  =259F E8D3DF      057F    call get_atts               ;al = interface attributes
6553  =25A2 BBC025              mov bx,offset close00       ;check_fcb1 goes to close00 if
6554  =25A5 53                  push bx                     ;a chksum error is detected
6555  =25A6 C606F308FF          mov check_fcb_ret,true
6556  =25AB E8F5E0      06A3    call check_fcb1
6557  =25AE 5B                  pop bx                      ;chksum ok
6558  =                     close000:
6559  =25AF C6060A09FF          mov byte ptr sdcnt+1,0ffh   ;initializes sdcnt
6560  =25B4 A04A0B              mov al,info_fcb+modnum      ;perform partial close if
6561  =25B7 2480                and al,80h                  ;fcb(modnum) & 80h = 80h
6562  =25B9 7511        25CC    jnz close01
6563  =25BB E8E0ED      139E      call close                ;perform full close
6564  =25BE EB14        25D4      jmps close02
6565  =                     close00:
6566  =
6567  =                     ;       if comp att f3' = 0 then report close checksum error
```

```
6568
6569  =                       ;       otherwise allow close to proceed by jumping to close000
6570  =
6571  =25C0 E8CCE0    068F             call get_cmp_mode
6572  =25C3 2420                       and al,20h
6573  =25C5 75E8      25AF             jnz close000
6574  =
6575  =25C7 B406                       mov ah,6                        ;checksum error
6576  =25C9 E9E8DE    04B4             jmp set_aret
6577  =                       close01:
6578  =25CC C606FB08FF                 mov dont_close,true             ;perform close but don't update
6579  =25D1 E8DEED    13B2             call close1                     ;directory fcb
6580  =                       close02:
6581  =
6582  =                       endif
6583  =                       if BCPM
6584  =                                call set_lsn
6585  =                                call chek_fcb                   ;check fcb chksum
6586  =                                mov ah,6
6587  =                                jz $+5
6588  =                                  jmp set_aret                  ;password error
6589  =                                call close                      ;close file
6590  =                       endif
6591  =
6592  =25D4 803E0D08FF                 cmp lret,0ffh
6593  =25D9 7501      25DC             jne $+3
6594  =25DB C3                ret34:     ret                          ;error on close
6595  =25DC E85705    2B36             call flush                      ;flush blocking/deblocking buffers
6596  =
6597  =                       if BNPM
6598  =25DF F606DE0880                 test attributes,80h             ;is this a partial close
6599  =25E4 75F5      25DB             jnz ret34                       ;yes - return
6600  =25E6 E807F7    1CF0             call pack_sdcnt
6601  =                                ;find olist item for this process and file
6602  =25E9 B505                       mov ch,5
6603  =25EB E80EF77542    1CFC         call search_olist! jnz close03
6604  =                                  ;decrement open count
6605  =
6606  =                       ;        default to partial close if comp att f2' = 1
6607  =                       ;        if permanent close, don't delete lock list item
6608  =                       ;        if interface attribute f6' = 1
6609  =
6610  =25F0 E89CE0    068F             call get_cmp_mode
6611  =25F3 2440753B  2632             and al,40h! jnz close03
6612  =25F7 FF4F08                     dec word ptr 8[bx]
6613  =25FA 807F09FF                   cmp byte ptr 9[bx],0ffh
6614  =25FE 7532      2632             jnz close03
6615  =2600 C64708FF                     mov byte ptr 8[bx],0ffh
6616  =2604 53E8F3E05B  06FB             push bx! call reset_checksum_fcb! pop bx
6617  =2609 891EFE08                     mov file_id,bx
6618  =260D A09E08                       mov al,high_ext               ;is file open in locked mode?
6619  =2610 0AC0750D  2621               or al,al! jnz close025        ;no
6620  =2614 F6470580                       test byte ptr 5[bx],80h     ;is file in F1' compatibility env?
```

```
6621
6622  =2618 7507          2621              jnz close025                ;yes
6623  =261A A0DE08                            mov al,attributes         ;did process request extended lock?
6624  =261D 24407511      2632                and al,40h| jnz close03 ;yes
6625  =                         close025:
6626  =2621 E82CF7        1D50              call delete_item            ;permanently close file
6627  =2624 F6069E0880                      test high_ext,80h
6628  =2629 7407          2632              jz close03                  ;if unlocked file,
6629  =262B 8B1EFE08                          mov bx,file_id            ; remove file's lock|o| entries
6630  =262F E863F7        1D95              call remove_locks
6631  =                         close03:
6632  =
6633  =                         endif
6634  =
6635  =2632 C3                        RET
6636  =
6637  =                         func17:         ;search for first occurrence of a file
6638  =                         ;======
6639  =
6640  =                         if BMPM
6641  =2633 A12B09                    mov ax,parametersegment         ;save user segment & fcb offset
6642  =2636 A38D08                    mov searchabase,ax
6643  =2639 A18108                    mov ax,info
6644  =263C A38B08                    mov searchaofst,ax
6645  =                         endif
6646  =
6647  =263F 32C0                      xor al,al
6648  =                         csearch:
6649  =2641 9C                        pushf                           ;z flag = 1 for search, 0 for searchn
6650  =2642 803E3C0B3F                cmp info_fcb,'?'                ;does fcb(0) = '?'
6651  =2647 750A          2653        jne csearch1                    ;no
6652  =2649 E86AF3        19B6          call curselect                ;current selected disk is searched
6653  =264C E8E8F3        1A37          call noselect0
6654  =264F 32C9                        xor cl,cl                     ;return first or next fcb
6655  =2651 EB13          2666          jmps csearch3                 ;all fcbs match
6656  =                         csearch1:
6657  =2653 BB480B                    mov bx,offset info_fcb+extnum   ;does fcb(ext) = '?'
6658  =2656 803F3F                    cmp b[bx],'?'
6659  =2659 7406          2661        je csearch2                     ;yes
6660  =265B E8F9E0        0757        call clr_ext                    ;clear high 3 bits of fcb(ext)
6661  =265E E8F0E0        0751        call clrmodnum                  ;fcb(modnum) = 0
6662  =                         csearch2:
6663  =2661 E883F3        19E7        call reselectx                  ;fcb(0) may specify drive
6664  =2664 8A0F                      mov cl,namlen
6665  =                         csearch3:
6666  =2666 9D9C                      popf| pushf
6667  =2668 741A          2684        jz csearch4
6668  =266A A18F088B08                  mov ax,dcnt| mov bx,ax
6669  =266F 2403                        and al,dskmsk
6670  =2671 3C03740F      2684          cmp al,3| je csearch4
6671  =2675 8BC3                          mov ax,bx
6672  =2677 50                            push ax
6673  =2678 24FC                          and al,0fch
```

```
6674
6675  =267A A38F08                          mov dcnt,ax
6676  =267D E877E5        08F7              call rddir
6677  =2680 8F058F08                        pop dcnt
6678  =                             csearch4:
6679  =2684 9D                              popf
6680  =2685 7505              268C        jnz csearch5
6681  =2687 E8BBE9              1045          call search
6682  =268A EB0A                2696          jmps dir_to_user
6683  =                             csearch5:
6684  =268C 8A0E8A08                        mov cl,searchl
6685  =2690 E890E9              1023        call searchi
6686  =2693 E885E9              1048        call searchn
6687  =                                     ;jmps dir_to_user
6688  =
6689  =                             dir_to_user:
6690  =                             ;-----------
6691  =                             ;       copy the directory entry to the user buffer
6692  =                             ;       after call to search or searchn by user code
6693  =
6694  =2696 803E0D08FF                      cmp lret,0ffh
6695  =269B 7416                26B8        je ret35
6696  =269D A18F08                          mov ax,dcnt
6697  =26A0 2403                            and al,dskmsk
6698  =26A2 A20D08                          mov lret,al
6699  =
6700  =26A5 8B16AA08                        mov dx,buffa                    ;source is directory buffer
6701  =26A9 8B1E8508                        mov bx,dma_ofst                 ;destination is user dma addr.
6702  =26AD B180                            mov cl,recsiz                   ;copy entire record
6703  =26AF 06                              push es                         ;move to user segment
6704  =26B0 8E068708                        mov es,dma_seg
6705  =26B4 E82BDE              04E2        call move
6706  =26B7 07                              pop es
6707  =26B8 C3                    ret35:    ret
6708  =
6709  =
6710  =                             func18:         ;search for next occurrence of a file name
6711  =                             ;======
6712  =
6713  =                             if BMPM
6714  =
6715  =26B9 A18D08                          mov ax,searchabase              ;2.1 fix - search next without
6716  =26BC 0B068B08                        or ax,searchaofst                       ; search first crashes the system
6717  =26C0 74F6                26B8        jz ret35
6718  =
6719  =26C2 A18D08                          mov ax,searchabase              ;set search parameters
6720  =26C5 A32B09                          mov parametersegment,ax
6721  =26C8 A18B08                          mov ax,searchaofst
6722  =26CB A38108                          mov info,ax
6723  =26CE E813DD              03E4        call parsave33
6724  =26D1 C7058308C08                     mov searcha,offset info_fcb
6725  =                             endif
6726  =
```

```
6727
6728  =26D7 0C01                        or al,1
6729  =26D9 E965FF        2641          jmp csearch
6730  =                                 ;ret
6731  =
6732  =                         func19:         ;delete a file
6733  =                         ;======
6734  =26DC E8E8F3        1ACA          call copy_dma_in_8
6735  =26DF E9A6EA        1188          jmp delet
6736  =                                 ;ret
6737  =
6738  =                         func20:         ;read a file
6739  =                         ;======
6740  =
6741  =                         if BMPM                           ;if comp att f4' = 1 then
6742  =26E2 E8B2DF        0697          call cond_check_fcb       ; don't call check_fcb
6743  =                         endif
6744  =                         if BCPM
6745  =                                 call check_fcb
6746  =                         endif
6747  =
6748  =26E5 E93AF0        1722          jmp diskread
6749  =                                 ;ret
6750  =
6751  =                         func21:         ;write a file
6752  =                         ;======
6753  =
6754  =                         if BMPM                           ;if comp att f4' = 1 then
6755  =26E8 E8ACDF        0697          call cond_check_fcb       ; don't call check_fcb
6756  =                         endif
6757  =                         if BCPM
6758  =                                 call check_fcb
6759  =                         endif
6760  =
6761  =26EB E98CF0        177A          jmp diskwrite
6762  =                                 ;ret
6763  =
6764  =                         func22:         ;make a file
6765  =                         ;======
6766  =26EE B109                        mov cl,9
6767  =26F0 E8D9F3        1ACC          call copy_dma_in
6768  =26F3 E889DE        057F          call get_atts             ;get and clear interface atts
6769  =26F6 E85EE0        0757          call clr_ext              ;clear high 3 bits of fcb(ext)
6770  =26F9 E855E0        0751          call clrmodnum
6771  =26FC E8E8F2        19E7          call reselectx
6772  =
6773  =                         if BMPM
6774  =26FF E8F9DF        06FB          call reset_checksum_fcb
6775  =                         endif
6776  =
6777  =2702 E826E7        0E28          call check_wild           ;verify no wild chars exist in fcb
6778  =2705 C7061508FFFF                mov xdcnt,0ffffh          ;init xdcnt for open,make
6779  =
```

```
6780
6781  =                       if BMPM
6782  =2708 C6060A09FF                mov byte ptr sdcnt+1,0ffh
6783  =                       endif
6784  =
6785  =2710 E85AEB    127D           call open                       ;attempt to open fcb
6786  =
6787  =                       if BMPM
6788  =2713 E8E5DF    06FB           call reset_checksum_fcb         ;invalidate fcb chksum in case file
6789  =                       endif                                  ;already exists
6790  =
6791  =2716 E85CE0    0775           call endofdir                   ;does file exist?
6792  =2719 740A      2725           jz make0a
6793  =271B E87CDE    059A             call get_dir_ext
6794  =271E 3A077203  2725             cmp al,[bx+1] jc make0a
6795  =2722 E98D0D    0482              jmp file_exists              ;yes - error
6796  =                       make0a:
6797  =2725 9C                        pushf
6798  =
6799  =                       if BMPM
6800  =2726 A0DE08                    mov al,attributes
6801  =2729 2480                      and al,80h
6802  =272B D0E87502  2731            shr al,1 jnz makex00
6803  =272F B080                        mov al,80h
6804  =                       makex00:
6805  =
6806  =                       ;       set the open mode to r/o if in default mode and
6807  =                       ;       attribute f1' is set.
6808  =
6809  =2731 8AE8                      mov ch,al
6810  =2733 D0C07309  2740            rol al,1 jnc makex001
6811  =2737 E855DF    068F              call get_cmp_mode
6812  =273A D0C07302  2740              rol al,1 jnc makex001
6813  =273E B520                          mov ch,20h
6814  =                       makex001:
6815  =2740 882ED908                  mov make_flag,ch
6816  =
6817  =2744 803E0A09FF                cmp byte ptr sdcnt+1,0ffh
6818  =2749 740A      2755            jz makex01
6819  =274B E8A2F5    1CF0              call pack_sdcnt
6820  =274E B503                        mov ch,3
6821  =2750 E8A9F5740A  1CFC            call search_olist! jz make_x02
6822  =                       makex01:
6823  =2755 833E520000                cmp free_root,0
6824  =275A 751A      2776            jne makex03
6825  =275C E987FD    24E6              jmp openx02
6826  =
6827  =                       makex02:
6828  =275F A0D908                    mov al,makeflag
6829  =2762 224702                    and al,2[bx]
6830  =2765 7503      276A            jnz $+5
6831  =                       makex025:
6832  =2767 E9DFFD    2549              jmp openx06
```

```
6833
6834  =276A E852F67407   1DBF          call compare_pas! jz makex03
6835  =276F F6050D90880                 test make_flag,80h
6836  =2774 75F1         2767           jnz makex025
6837  =                             make_x03:
6838  =
6839  =                             endif
6840  =
6841  =2776 E8F5F4       1C6E          call qdirfcb1                   ;is fcb 1st fcb for file?
6842  =2779 7420         2795          jz makex04                      ; yes
6843  =277B E860F3       1ADE          call get_dir_mode               ;does dir lbl require passwords?
6844  =277E 2480                       and al,80h
6845  =2780 7419         279B          jz makex04                      ; no
6846  =2782 E860F3       1AE5          call get_xfcb                   ;does xfcb exist with mode 1 or 2
6847  =2785 24C0                       and al,0c0h                     ;password?
6848  =2787 7412         279B          jz makex04                      ; no
6849  =2789 E881F4       1C0D          call chk_xfcb_password1         ;check password
6850  =278C 740D         279B          jz makex04
6851  =278E E8A1F3       1B32          call chk_pw_error               ;verify password error
6852  =2791 F606DD08C0                 test pw_mode,0c0h
6853  =2796 7403         279B          jz makex04
6854  =2798 E9FDF3       1B98            jmp pw_error
6855  =                             makex04:
6856  =279B 9D                         popf
6857  =279C 7203         27A1          jc $+5
6858  =279E E83DEC       13DE            call make                     ;make fcb
6859  =
6860  =                             if BMPM
6861  =27A1 E857DF       06FB          call reset_checksum_fcb         ;invalidate fcb chksum in case
6862  =                             endif                             ;make fails
6863  =
6864  =27A4 E8CEDF       0775          call endofdir                   ;did make fail?
6865  =27A7 7501         27AA          jnz $+3                         ;no
6866  =27A9 C3                     ret36:  ret                         ;yes - no room in directory
6867  =
6868  =                             if BCPM
6869  =                                 call set_lsn
6870  =                             endif
6871  =
6872  =27AA E831F3       1ADE          call get_dir_mode               ;get dir lbl data byte
6873  =27AD A880                       TEST AL,80H                     ;ARE PASSWORDS ACTIVATED?
6874  =27AF 745E         280F          JZ MAKE3A                       ;NO
6875  =27B1 F6050E0840                 TEST ATTRIBUTES,40H             ;HAS USER SUPPLIED A PASSWORD?
6876  =27B6 7457         280F          JZ MAKE3A                       ;NO
6877  =27B8 E8B3F4       1C6E          CALL QDIRFCB1                   ;IS FCB IN 1ST DIR FCB?
6878  =27BB 7552         280F          JNZ MAKE3A                      ;NO
6879  =27BD E8B6E9       1176            CALL XDCNT_EQ_DCNT
6880  =27C0 E822F3       1AE5            CALL GET_XFCB                 ;DOES XFCB EXIST FOR FILE?
6881  =27C3 7513         27D8            JNZ MAKE00                    ;YES
6882  =27C5 C6051308FF                     mov make_xfcb,true          ;attempt to make xfcb
6883  =27CA E811EC       13DE              call make
6884  =27CD 7509         27D8              jnz make00                  ;make succeeded
6885  =27CF E86DE8       103F              call search_name            ;delete the fcb that was created
```

```
6886
6887  =27D2 E800EA      11D5            CALL DELETE10               ;above
6888  =27D5 E9D7E8      10AF            jmp lret_eq_ff              ;return no room in dir error
6889  =                           make00:
6890  =27D8 E831F3      1B0C          call init_xfcb                ;initialize xfcb
6891  =27DB BEC908                    mov si,offset common_dma      ;pick up password mode from 9th
6892  =27DE 83C608                    add si,8                      ;byte of dma
6893  =27E1 AC                        lods al
6894  =27E2 24E0                      and al,0e0h                   ;has password mode been specified?
6895  =27E4 7502        27E8          jnz make2                     ;yes
6896  =27E6 B080                      mov al,80h                    ;default to read mode
6897  =                           make2:
6898  =27E8 A2DD08                    MOV pw_mode,AL                ;SAVE PASSWORD MODE
6899  =27EB 50                        push ax
6900  =27EC E80EF3      1AFD          call get_xfcb1                ;bx = .dirbuff xfcb(ext)
6901  =27EF 58                        pop ax
6902  =27F0 8807                      mov [bx],al                   ;xfcb(ext) = password mode
6903  =27F2 BEC908                    mov si,offset common_dma
6904  =27F5 E818F4      1C13          call set_pw                   ;xfcb(s1) = password sum
6905  =27F8 E8CE04      2CC9          CALL SDL3                     ;FIX HASH AND UPDATE DIRECTORY
6906  =27FB E87FE9      1170          CALL RESTORE_DIR_FCB          ;RETURN TO FCB THAT WAS MADE ABOVE
6907  =27FE 74A9        27A9          JZ RET36                      ;THIS ERROR SHOULD NOT OCCUR
6908  =2800 E840F4      1C43          call get_dtba8                ;GET SFCB ADDR
6909  =2803 0AC0                      OR AL,AL
6910  =2805 7508        280F          JNZ MAKE3A                    ;SFCB DOESNT EXIST
6911  =2807 A0DD08                      MOV AL,pw_mode
6912  =280A 8807                      MOV [BX],AL                   ;STORE PASSWORD MODE IN SFCB
6913  =280C E8FEE3      0C0D            CALL wrdir                  ;UPDATE SFCB
6914  =                           MAKE3A:
6915  =280F B150                  MOV CL,050h
6916  =2811 E883FD      2597        call openx2                     ;place create or access stamp in SFCB
6917  =2814 B120                    mov cl,20h                      ;is update stamp requested
6918  =2816 E863F4      1C84          call qstamp                   ; on drive
6919  =2819 7508        2823          jnz make3b                    ; no
6920  =281B E87CF4      1C9A            call stamp2                 ;make create or access stamp
6921  =281E 800E4A0B40                  or info_fcb+modnum,40h      ;set file write flag
6922  =                           make3b:
6923  =
6924  =                           if BMPM
6925  =2823 A0D908                    mov al,make_flag              ;set attributes to open attributes
6926  =2826 A2DE08                    mov attributes,al
6927  =2829 2440                      and al,40h
6928  =282B D0E0                      shl al,1
6929  =282D A29E08                    mov high_ext,al               ;high_ext = shl(attributes,1)
6930  =
6931  =2830 803E0A09FF                cmp byte ptr sdcnt+1,0ffh
6932  =2835 740C        2843          je makexx02
6933  =2837 A11508                      mov ax,xdcnt
6934  =283A A30909                      mov sdcnt,ax
6935  =283D E8B0F4      1CF0            call pack_sdcnt
6936  =2840 E9A8FC      24EB            jmp openx03
6937  =                           makexx02:
6938  =2843 E8F1F9      2237          call fix_olist_item
```

```
6939
6940  =2846 E974FC      2480          jmp openx1a
6941  =                         endif
6942  =
6943  =                         if BDPM
6944  =                               ret
6945  =                         endif
6946  =
6947  =                         func23:         ;rename a file
6948  =                         ;======
6949  =2849 E87EF2      1ACA          call copy_dma_in_8
6950  =                               ;jmp rename
6951  =
6952  =                         rename:
6953  =                         ;------
6954  =                         ;       rename the file described by the first half of
6955  =                         ;       the currently addressed file control block. the
6956  =                         ;       new name is contained in the last half of the
6957  =                         ;       currently addressed file conrol block. the file
6958  =                         ;       name and type are changed, but the reel number
6959  =                         ;       is ignored.  the user number is identical.
6960  =
6961  =                         if BMPM
6962  =284C E8300D      057F          call getatts
6963  =                         endif
6964  =
6965  =284F E8D9E5      0E2B          call check_wild                 ;check for ?s in 1st filename
6966  =2852 E863F3      18C0          call chk_password               ;check password
6967  =2855 7403        285A          jz $+5
6968  =2857 E8D8F2      1B32            call chk_pw_error
6969  =285A E8F0E8      114D          call init_xfcb_search
6970  =
6971  =285D E814EA      1274          call copy_user_no               ;fcb(16) = fcb(0)
6972  =2860 891E8308                  mov searcha,bx
6973  =2864 E8C7E5      0L2E          call check_wild0                ;check 2nd filename for ?s
6974  =
6975  =2867 B10C                     mov cl,extnum                   ;check to see if new filename
6976  =2869 E8BEE7      102A          call searchi1                   ;already exist on drive
6977  =286C E8D9E7      104d          call search1
6978  =286F 7403        2874          jz $+5
6979  =2871 E93EDC      0482            jmp file_exists               ;error - file exists
6980  =2874 E8E1E8      1158          call does_xfcb_exist            ;delete xfcb if present
6981  =2877 7403        287C          jz $+5
6982  =2879 E85EE9      11DA            call deletell
6983  =287C E8F5E9      1274          call copy_user_no               ;fcb(16) = fcb(0) again
6984  =287F E8CBE8      1140          call init_xfcb_search
6985  =                                                               ;search up to the extent field
6986  =2882 E8BEE7      1043          call search_ext
6987  =2885 7501        2888          jnz $+3
6988  =2887 C3                        ret
6989  =2888 E8ADDE      0738          call ckrodir                    ;check for read/only file
6990  =
6991  =                         if BMPM
```

```
6992
6993  =288B E83FF5       1DCD          call chk_olist
6994  =                        endif
6995  =
6996  =                        renam0:
6997  =288E B110                       mov cl,dskmap                ;not end of directory,
6998  =2890 B20C                       mov dl,extnum                ;rename next element
6999  =2892 E8C0E9       1255          call copy_dir
7000  =2895 E838E7       0FD3          CALL FIX_HASH
7001  =
7002  =                                                             ;element renamed, move to next
7003  =2898 E8B0E7       1048          call searchn
7004  =289B 75F1         288E          jnz renam0
7005  =289D E888E8       1158          call does_xfcb_exist         ;rename xfcb if present
7006  =28A0 7503         28A5          jnz $+5
7007  =28A2 E96EE5       0E13            jmp cpydirloc
7008  =28A5 E8CCE9       1274          call copy_user_no            ;xfcb(16) = xfcb(0)
7009  =28A8 EBE4         288E          jmps renam0
7010  =
7011  =                        func27:         ;return the allocation vector address
7012  =                        ;======
7013  =28AA E809F1       1986          call curselect
7014  =28AD 8C1E2D09                   mov returnseg,ds
7015  =28B1 8B1EB008                   mov bx,alloca
7016  =28B5 EB70         2927          jmps sthlret
7017  =                                ;ret
7018  =
7019  =                        func28:         ;write protect current disk
7020  =                        ;======
7021  =
7022  =                        if BMPM
7023  =28B7 C606000901                 mov set_ro_flag,1
7024  =28BC 8A0E7F08                   mov cl,seldsk
7025  =28C0 BB0100                     mov bx,1
7026  =28C3 D3E3                       shl bx,cl                    ;form drive vector from seldsk
7027  =28C5 E885F9       227D          call intrnl_disk_reset       ;verify no other process has files open
7028  =                        endif
7029  =
7030  =28C8 BB0608                     mov bx,offset rodsk          ;set selected disk to read only
7031  =28CB 8A0E7F08                   mov cl,seldsk
7032  =28CF E83CDE       070E          call set_cdisk1              ;sets bit to 1
7033  =                                                             ;high water mark in directory
7034  =28D2 8B163F08                   mov dx,dirmax                ;goes to max
7035  =28D6 42                         inc dx
7036  =28D7 8B1EA608                   mov bx,cdrmaxa               ;bx = .cdrmax
7037  =28DB 8917                       mov [bx],dx                  ;cdrmax = dirmax
7038  =28DD C3                         ret
7039  =
7040  =                        func30:         ;set file indicators
7041  =                        ;======
7042  =28DE E8E9F1       1ACA          call copy_dma_in_8
7043  =28E1 E847E5       0E2B          call check_wild
7044  =28E4 E800F1       19E7          call reselectx
```

```
7045
7046  =28E7 E8D6F2        1DC0          call chk_password
7047  =28EA 7403          28EF          jz $+5
7048  =28EC E843F2        1B32            call chk_pw_error
7049  =
7050  =                                 ;       set file indicators for current fcb
7051  =
7052  =28EF E88DDC        057F          call get_atts
7053  =28F2 E84EE7        1043          call search_ext                 ;through file type
7054  =28F5 7434          2928          jz ret37
7055  =
7056  =                                 if BMPM
7057  =28F7 E8D3F4        1DCD          call chk_olist
7058  =                                 endif
7059  =
7060  =                                 indic0:
7061  =28FA B100                        mov cl,0                        ;not end of directory,
7062  =28FC B20C                        mov dl,extnum                   ;continue to change
7063  =28FE E847E9        1248          call copy_dir2                  ;copy name
7064  =2901 E8DED8        04E2          call move
7065  =2904 F6065DE0840                 TEST ATTRIBUTES,40H             ;IS ATT F6' SET?
7066  =2909 7406          2911          JZ INDIC1                       ;NO
7067  =290B A05C0B                        mov al,info_fcb+nxtrec        ;DIR FCB(S1) = FCB(CR)
7068  =290E A2490B                        mov info_fcb+chksum,al
7069  =                                 INDIC1:
7070  =2911 E8F9E2        0C0D          call wrdir
7071  =2914 E834E7        1048          call searchn
7072  =2917 75E1          28FA          jnz indic0
7073  =2919 E9F7E4        0E13          jmp cpydirloc                   ;lret=dirloc
7074  =                                 ;ret
7075  =
7076  =                                 func31:         ;return address of disk parameter block
7077  =                                 ;======
7078  =291C E897F0        1986          call curselect
7079  =291F 8C1E2D09                    mov returnseg,ds
7080  =2923 8B1EAC08                    mov bx,dpbaddr
7081  =                                 sthlret:
7082  =2927 891E0D08                    mov aret,bx
7083  =292B C3                          ret37:  ret
7084  =
7085  =                                 func33:         ;random disk read operation
7086  =                                 ;======
7087  =
7088  =                                 if BMPM                         ;if comp att f4' = 1 then
7089  =292C E858DD        0697          call cond_check_fcb             ; don't call check_fcb
7090  =                                 endif
7091  =                                 if BCPM
7092  =                                 call check_fcb
7093  =                                 endif
7094  =
7095  =292F B1FF                        mov cl,true                     ;marked as read operation
7096  =2931 E8C8EB        14FC          call rseek
7097  =2934 75F5          2928          jnz ret37
```

```
7098
7099  =2936 E9E9ED       1722          jmp diskread                    ;if seek successful
7100  =
7101  =                        func34:          ;random disk write operation
7102  =                        ;======
7103  =
7104  =                        if BMPM                                  ;if comp att f4' = 1 then
7105  =2939 E858DD       0697          call cond_check_fcb             ; don't call check_fcb
7106  =                        endif
7107  =                        if BCPM
7108  =                              call check_fcb
7109  =                        endif
7110  =
7111  =293C B100                    mov cl,false                     ;marked as write operation
7112  =293E E888E9       14FC        call rseek
7113  =2941 75E8         2928        jnz ret37
7114  =2943 E934EE       177A        jmp diskwrite                   ;if seek successful
7115  =
7116  =                        func35:          ;return file size (0-65536)
7117  =                        ;======
7118  =                        ;       compute logical file size for current fcb
7119  =
7120  =2946 BB5D08                  mov bx,offset info_fcb+ranrec    ;zero receiving ranrec field
7121  =2949 33C0                    xor ax,ax
7122  =294B 8907                    mov [bx],ax
7123  =294D 884702                  mov 2[bx],al
7124  =2950 E8F0E6       1043        call search_ext
7125  =2953 741D         2972        jz setsize
7126  =                        getsize:
7127  =2955 E8C0DD       0725          call get_dptra                 ;current fcb addressed by dptr
7128  =2958 BA0F00                     mov dx,reccnt                  ;ready for compute size
7129  =295B E8C6EF       1924          call compute_rr
7130  =                                                               ;al=0000 0?mm,
7131  =                                                               ;cx = mmmm eeee errr rrrr
7132  =                                                               ;compare with memory, larger?
7133  =295E E8E7EF       1948          call compare_rr
7134  =2961 7205         2968          jb getnextsize                 ;for another try
7135  =                                                               ;fcb is less or equal,
7136  =                                                               ;fill from directory
7137  =2963 884702                       mov 2[bx],al
7138  =2966 890F                         mov [bx],cx
7139  =                        getnextsize:
7140  =2968 E8E0E6       1048          call searchn
7141  =2968 C606D00800               mov lret,0
7142  =2970 75E3         2955          jnz getsize
7143  =                        setsize:
7144  =2972 C3                      ret
7145  =
7146  =                        func36:          ;set random record
7147  =                        ;======
7148  =                        ;       set random record from the current fcb
7149  =
7150  =2973 BB3C08                  mov bx,offset info_fcb
```

```
7151
7152  =2976 8A2000                     mov dx,nxtrec                ;ready params for computesize
7153  =2979 E8A8EF        1924         call compute_rr              ;dx=.info_fcb, al=cy,
7154  =                                                             ;cx=mmmm aeee errr rrrr
7155  =297C 894F21                     mov ranrec[bx],cx            ;store in ranrec field
7156  =297F 884723                     mov ranrec+2[bx],al
7157  =2982 C3                         ret
7158  =
7159  =                          func37:          ;reset drive
7160  =                          ;======
7161  =
7162  =                          if BMPM
7163  =2983 E8E3F8        2271         call diskreset
7164  =                          reset_37:
7165  =
7166  =                          endif
7167  =
7168  =2986 A18108                     mov ax,info
7169  =                          RESET_37X:
7170  =2989 F7D0                       not ax
7171  =298B 50                         push ax
7172  =298C BB0808                     mov bx,offset dlog
7173  =298F 2307                       and ax,[bx]
7174  =2991 8907                       mov [bx],ax
7175  =2993 58                         pop ax
7176  =2994 50                         push ax
7177  =2995 BB0608                     mov bx,offset rodsk
7178  =2998 2307                       and ax,[bx]
7179  =299A 8907                       mov [bx],ax
7180  =299C 58                         pop ax
7181  =299D BB0008                     mov bx,offset rlog
7182  =29A0 2307                       and ax,[bx]
7183  =29A2 8907                       mov [bx],ax
7184  =29A4 8A0EA008                   mov cl,curdsk                ;if curdsk reset, set to 0ffh
7185  =29A8 8B160808                   mov dx,dlog
7186  =29AC E86FDD        071E         call test_vector1
7187  =29AF 7505          29B6         jnz ret38
7188  =29B1 C606A008FF                 mov curdsk,0ffh
7189  =29B6 C3                   ret38: ret
7190  =
7191  =                          if BMPM
7192  =
7193  =                          func38:          ;access drive
7194  =                          ;======
7195  =29B7 C706A108FFFF                mov word ptr packed_dcnt,0ffffh
7196  =29BD C606A30800                  mov byte ptr packed_dcnt+2,0
7197  =29C2 32C0                       xor al,al
7198  =                          acc_drv0:
7199  =29C4 D1E21400                   shl dx,1| adc al,0
7200  =29C8 0BD275F8      29C4         or dx,dx| jnz acc_drv0
7201  =29CC 0AC074E6      29B6         or al,al| jz ret38           ;al = # of open items to create
7202  =29D0 8AD0FEC8                   mov dl,al| dec al
7203  =29D4 50                         push ax
```

```
7204
7205  =29D5 E80400        29DC          call acc_drv02
7206  =29D8 58                          pop ax
7207  =29D9 E90AFB        24E6          jmp openx02
7208  =
7209  =                           acc_drv02:
7210  =                           ;---------
7211  =29DC E867F55B      1F46          call check_free01 pop bx        ;discard return address
7212  =29E0 E838F55B      1F1E          call count_opens1 pop bx        ;restore # of open items requested
7213  =29E4 021EFC087229  2A13          add bl,open_cnt1 jb acc_drv3
7214  =29EA 2A1E8B007323  2A13          sub bl,openmax1 jae acc_drv3
7215  =29F0 8B1E8108                      mov bx,info
7216  =29F4 A0A00850                      mov al,curdsk1 push ax
7217  =29F8 B010                          mov al,16
7218  =                           acc_drv1:
7219  =29FA FEC8                          dec al
7220  =29FC D1E3730A      2A0A            shl bx,11 jnc acc_drv15
7221  =2A00 5053                          push ax1 push bx
7222  =2A02 A2A008                        mov curdsk,al
7223  =2A05 E8D8F4        1EE0            call create_olist_item
7224  =2A08 5B58                          pop bx1 pop ax
7225  =                           acc_drv15:
7226  =2A0A 0AC075EC      29FA            or al,all jnz acc_drv1
7227  =2A0E 58A2A008                      pop ax1 mov curdsk,al
7228  =2A12 C3                            ret
7229  =                           acc_drv3:
7230  =2A13 E9DAFA        24F0          jmp openx034
7231  =
7232  =                           func59:         ;free drive
7233  =                           ;======
7234  =2A16 F7060509FFFF                  test open_root,0ffffh
7235  =2A1C 7445          2A63          JZ FREE_DRV35
7236  =2A1E C606F50800                    mov incr_pdcnt,false
7237  =2A23 8B1E8108                      mov bx,info
7238  =2A27 83FBFF750A    2A36            cmp bx,0ffffh1 jnz free_drv1
7239  =2A2C C606F60800                      mov free_mode,0
7240  =2A31 E85FF4        1E93              call free_files
7241  =2A34 EB23          2A59              jmps free_drv3
7242  =                           free_drv1:
7243  =2A36 C606F60801                    mov free_mode,1
7244  =2A3B A0A00850                      mov al,curdsk1 push ax
7245  =2A3F B010                          mov al,16
7246  =                           free_drv2:
7247  =2A41 FEC8                          dec al
7248  =2A43 D1E3730A      2A51            shl bx,11 jnc free_drv25
7249  =2A47 5053                          push ax1 push bx
7250  =2A49 A2A008                        mov curdsk,al
7251  =2A4C E844F4        1E93            call free_files
7252  =2A4F 5B58                          pop bx1 pop ax
7253  =                           free_drv25:
7254  =2A51 0AC075EC      2A41            or al,all jnz free_drv2
7255  =2A55 58A2A008                      pop ax1 mov curdsk,al
7256  =                           free_drv3:
```

```
7257
7258  =2A59 803EF508FF                  cmp incr_pdcnt,true
7259  =2A5E 7503          2A63          JNZ FREE_DRV35
7260  =2A60 E8C6F3        1E29            CALL DEC_PDCNT
7261  =                         FREE_DRV35:
7262  =2A63 EB5E          2AC3          jmps FUNC48
7263  =                         endif
7264  =                         if BCPM
7265  =
7266  =                         func38  equ     funcret         ;access drive
7267  =                         ;=====
7268  =                         func39  equ     funcret         ;free drive
7269  =                         ;=====
7270  =
7271  =                         endif
7272  =
7273  =                         func40:         ;write random with zero fill
7274  =                         ;=======
7275  =                         ;       random disk write with zero fill of unallocated block
7276  =
7277  =                         if BMPM                                 ;if comp att f4' = 1 then
7278  =2A65 E82FDC        0697          call cond_check_fcb             ; don't call check_fcb
7279  =                         endif
7280  =                         if BCPM
7281  =                               call check_fcb
7282  =                         endif
7283  =
7284  =2A68 B100                      mov cl,false
7285  =2A6A E88FEA        14FC        call rseek
7286  =2A6D 7401          2A70        jz func40a
7287  =2A6F C3                          ret
7288  =                         func40a:
7289  =2A70 E907ED        177A        jmp diskwrite
7290  =                               ;ret
7291  =
7292  =                         if BMPM
7293  =
7294  =                         func42:         ;lock record
7295  =                         ;=======
7296  =2A73 C606F408FF                mov lock_unlock,true
7297  =2A78 E90CF6        2087        jmp lock
7298  =
7299  =                         func43:         ;unlock record
7300  =                         ;=======
7301  =2A7B C606F40800                mov lock_unlock,false
7302  =2A80 E904F6        2087        jmp lock
7303  =
7304  =                         endif
7305  =                         if BCPM
7306  =
7307  =                         func42  equ     funcret         ;lock record
7308  =                         ;=====
7309  =                         func43  equ     funcret         ;unlock record
```

```
7310
7311  =                         ;=====
7312  =
7313  =                         endif
7314  =
7315  =                         func46:         ;get disk free space
7316  =                         ;======
7317  =2A83 E82CEF      1982    call tmp_select                 ;perform temporary select of
7318  =                                                         ;specified drive
7319  =
7320  =2A86 8B368D08            mov si,alloca                   ;si = allocation vector adar
7321  =2A8A E84BE2      0CD8    call get_nalbs                  ;bx = # of allocation vector bytes
7322  =2A8D 33C9                xor cx,cx                       ;cx = true bit counter
7323  =
7324  =                         ;count # of true bits in allocation vector
7325  =
7326  =2A8F AC          gsp1:   lods al                         ;increments si
7327  =2A90 0AC0        gsp2:   or al,al
7328  =2A92 7407        2A9B    jz gsp4
7329  =2A94 D0E8        gsp3:     shr al,1
7330  =2A96 73FC        2A94      jnc gsp3
7331  =2A98 41                    inc cx
7332  =2A99 EBF5        2A90      jmps gsp2
7333  =2A9B 4B          gsp4:   dec bx
7334  =2A9C 75F1        2A8F    jnz gsp1
7335  =
7336  =                         ;       bx = 0 when allocation vector scanned
7337  =                         ;       compute maxall + 1 - bc
7338  =
7339  =2A9E 8B1E8D08            mov bx,maxall
7340  =2AA2 43                  inc bx
7341  =2AA3 2BD9                sub bx,cx                       ;bx = # of available blocks on drive
7342  =2AA5 8A0E8A08            mov cl,blkshf                   ;convert # of blocks to # of records
7343  =2AA9 32ED                xor ch,ch
7344  =2AAB 8AC7                mov al,bh
7345  =2AAD 32E4                xor ah,ah
7346  =2AAF D3E3                shl bx,cl
7347  =2AB1 D3E0                shl ax,cl                       ;ah,bh,bl = # of available records
7348  =2AB3 893E8508            mov di,dma_ofst                 ;store # of records in 1st 3 bytes
7349  =2AB7 1E                  push ds                         ;of user's dma address
7350  =2AB8 8E1E8708            mov ds,dma_seg
7351  =2ABC 8919                mov [di],bx
7352  =2ABE 884502              mov 2[di],ah
7353  =2AC1 1F                  pop ds
7354  =2AC2 C3                  ret
7355  =
7356  =                         func48:         ;flush buffers
7357  =                         ;======
7358  =2AC3 803E7A081A          CMP fx_intrn,fx139              ;DONT MAKE BIOS FLUSH CALL
7359  =2AC8 7408        2AD2    JE flush0                       ;IF func# <> free drive
7360  =2ACA B00C                  mov al,io_flush               ; flush additional
7361  =2ACC E85AD9      0429      CALL xiosif                   ; blocking/deblocking buffers
7362  =2ACF E824DD      07F6      CALL DIOCOMP0
```

```
7363
7364  =                         flush0:
7365  =2AD2 8B1E0808                    MOV BX,DLOG                 ;DO I = 15 TO 0 BY -1
7366  =2AD6 B210                        MOV dl,16
7367  =                         FLUSH1:
7368  =2AD8 FECA                        DEC dl
7369  =2ADA D1E3                        shl bx,1                    ;IS DRIVE I LOGGED IN?
7370  =2ADC 7353          2B31          jnc FLUSH6                  ; no - try next drive
7371  =2ADE 5352                        PUSH BX! push dx            ; yes
7372  =2AE0 E8CFEE        1982          call tmp_select             ;select drive in CL
7373  =2AE3 803E7A0824                  cmp fx_intrn,fx198          ;does func# = reset allocation
7374  =2AE8 7513          2AFD          jne flush1b                 ; no
7375  =
7376  =                         if BMPM
7377  =2AEA B501                        mov ch,1
7378  =2AEC E80DF2        1CFC          call search_olist           ;any open files on drive
7379  =2AEF 7405          2AF6          jz flush1a                  ; yes - dont copy ALV
7380  =                         endif
7381  =
7382  =2AF1 E8EEE1        0CE2            call copy_alv             ;Z reset - copy 2nd ALV to 1st
7383  =2AF4 E839          2B2F            jmps flush4
7384  =
7385  =                         if BMPM
7386  =
7387  =                         flush1a:
7388  =2AF6 C60600D08FF                 mov lret,0ffh
7389  =2AFB EB32          2B2F          jmps flush4
7390  =                         endif
7391  =
7392  =                         flush1b:
7393  =2AFD 803E7A081A                  CMP fx_intrn,fx139          ;DONT FLUSH IF func# = free drive
7394  =2B02 741D          2B21          JE FLUSH3                   ;JUST DISCARD BCB'S
7395  =2B04 803E7A082D                    CMP fx_intrn,fx1143       ;IS func# = TERMINATE PROCESS?
7396  =2B09 740C          2B17            JE FLUSH2                 ; YES
7397  =2B0B E83000        2B3E              CALL FLUSHX             ;FLUSH DISK I
7398  =2B0E 803E1009FF                      CMP LINFO,0FFH          ;WAS FLUSH CALLED WITH DL = 0FFH?
7399  =2B13 751A          2B2F              JNE FLUSH4              ;NO
7400  =2B15 EB11          2B28              JMPS FLUSH35            ;YES - DISCARD PROCESS'S DATA BUFFERS
7401  =                         flush2:
7402  =
7403  =                         if BMPM
7404  =2B17 B501                        mov ch,1
7405  =2B19 E8E0F1        1CFC          call search_olist           ;any open files on drive
7406  =2B1C 7403          2B21          jz flush3                   ; yes - dont copy ALV
7407  =                         endif
7408  =2B1E E8C1E1        0CE2            call copy_alv             ;Z reset - copy 2nd ALV to 1st
7409  =                         FLUSH3:
7410  =2B21 8B1EB208                    MOV BX,DIR_BCBA             ;DEACTIVATE BCB'S IF func# = free drv
7411  =2B25 E854DD        087C          CALL DEACTIVATE             ;OR TERMINATE PROCESS
7412  =                         FLUSH35:
7413  =2B28 8B1EB408                    MOV BX,DAT_BCBA
7414  =2B2C E843DD        087C          CALL DEACTIVATE             ;DEACTIVATE DISCARDS IF func# = 48,39
7415  =                         FLUSH4:
```

```
7416
7417  =2B2F 5A5B                        pop dx! POP BX
7418  =                           flush6:
7419  =2B31 09DB                        OR BX,BX                    ;ARE THERE ANY ADDITIONAL LOGGED-IN
7420  =2B33 75A3          2AD8          JNZ FLUSH1                  ;DRIVES? - YES
7421  =2B35 C3                          RET
7422  =
7423  =                           FLUSH:
7424  =                           ;-----
7425  =2B36 B00C                        mov al,io_flush             ;flush additional
7426  =2B38 E8EED8        0429          CALL xiosif                 ; blocking/deblocking buffers
7427  =2B3B E868DC        07F6          CALL DIOCOMPO
7428  =                           FLUSHX:
7429  =2B3E F605C808FF                  TEST PHYMSK,0FFH            ;IS PHYSICAL RECORD > 128?
7430  =2B43 7501          2B46          JNZ $+3
7431  =2B45 C3                    ret48:  RET                       ;NO - RETURN
7432  =2B46 B404                        MOV AH,4
7433  =2B48 E92EDF        0A79          JMP DEBLOCK_DTA             ;FLUSH DATA BUFFERS
7434  =
7435  =                           if BMPM
7436  =
7437  =  0481                    func49  equ     funcret         ;not implemented
7438  =                           ;=====
7439  =  0481                    func50  equ     funcret         ;direct bios call
7440  =                           ;=====
7441  =
7442  =                           endif
7443  =
7444  =  2AD2                    func98  equ     flush0          ;reset allocation vector
7445  =                           ;=====
7446  =
7447  =                           save_first_dcnt:
7448  =                           ;---------------
7449  =2B4B 803E1608FD                  cmp byte ptr xdcnt+1,0fdh
7450  =2B50 75F3          2B45          jne ret48
7451  =2B52 E9C7E4        101C          jmp save_dcnt_pos
7452  =
7453  =                           func99:         ;truncate file
7454  =                           ;======
7455  =2B55 E872EF        1ACA          call copy_dma_in_8
7456  =2B58 E88CEE        19E7          call reselectx
7457  =2B5B E8CDE2        0E2B          call check_wild
7458  =2B5E E85FF0        1BC0          call chk_password
7459  =2B61 7403          2B66          jz $+5
7460  =2B63 E8CCEF        1B32            call chk_pw_error
7461  =2B66 E816DA        057F          call get_atts
7462  =2B69 B1FF                        mov cl,true
7463  =2B6B E88EE9        14FC          call rseek
7464  =2B6E 7403          2B73          jz $+5
7465  =                           lret_jmp:
7466  =2B70 E93CE5        10AF            jmp lret_eq_ff
7467  =2B73 E897E0        0C0D          call wrdir                  ;needed to check for write
7468  =2B76 E8ACDB        0725          call get_dptra
```

```
7469
7470  =2B79 BA0F00                      mov dx,reccnt
7471  =2B7C E8A5ED        1924          call compute_rr             ;CX,AL = fcb size
7472  =2B7F E8C5ED        1943          call compare_rr             ;is dir fcb size <= rrnrec
7473  =2B82 76EC          2B70          jbe lret_jmp                ; yes
7474  =2B84 E88CE4        1043          call search_ext             ;compute dir fcb size
7475  =2B87 74E7          2B70          jz lret_jmp
7476  =2B89 E8ACDB        0738          call ckrodir                ;may be r/o file
7477  =                           if BMPM
7478  =2B8C E83EF2        1DCD          call chk_olist
7479  =                           endif
7480  =
7481  =2B8F C6061608FD                  mov byte ptr xdcnt+1,0fdh
7482  =                           trunc1:
7483  =2B94 E88EDB        0725          call get_dptra              ;compare module and extent
7484  =2B97 83C30C                      add bx,extnum
7485  =2B9A BE480B                      mov si,offset info_fcb+extnum
7486  =2B9D 8A4402                      mov al,2[si]                ;AL = info_fcb(mod)
7487  =2BA0 243F                        and al,3fh
7488  =2BA2 384702                      cmp 2[bx],al                ;is dirfcb(mod) ~= infofcb(mod)?
7489  =2BA5 750D          2BB4          jne trunc12
7490  =2BA7 8A07                          mov al,[bx]               ;AL = dirfcb(ext)
7491  =2BA9 8A0C                          mov cl,[si]               ;CL = infofcb(ext)
7492  =2BAB E82EDA        05DC            call compext
7493  =2BAE 7404          2BB4            jz trunc12                ;dirfcb(ext) = infofcb(ext)
7494  =2BB0 8A07                            mov al,[bx]
7495  =2BB2 3AC1                            cmp al,cl
7496  =                           trunc12:
7497  =2BB4 7305          2BBB          jnb trunc13                 ;is dirfcb < infofcb?
7498  =2BB6 E892FF        2B4B            call save_first_dcnt
7499  =2BB9 EB15          2BD0            jmps trunc2
7500  =                           trunc13:
7501  =2BBB 9C                          pushf                       ; no
7502  =2BBC B100                        mov cl,0                    ;remove dirfcb blocks
7503  =2BBE E881E1        0D42          call scndmab                ; from allocation vector
7504  =2BC1 9D                          popf
7505  =2BC2 7424          2BE8          jz trunc3                   ;is dirfcb = fcb?
7506  =2BC4 E85EDB        0725          call get_dptra              ; no
7507  =2BC7 C607E5                      mov b[bx],empty             ;delete dirfcb
7508  =2BCA E806E4        0FD3          call fix_hash
7509  =                           trunc15:
7510  =2BCD E83DE0        0C0D          call wrdir
7511  =                           trunc2:
7512  =2BD0 E878E4        1048          call searchn
7513  =2BD3 75BF          2B94          jnz trunc1
7514  =2BD5 E8ECF0        1CC4          call update_stamp
7515  =                           if BMPM
7516  =2BD8 B505                        mov ch,5
7517  =2BDA E81FF1        1CFC          call search_olist
7518  =2BDD 7506          2BE5          jnz trunc25
7519  =2BDF A11508                        mov ax,xdcnt
7520  =2BE2 894703                        mov 3[bx],ax              ;reset dcnt in olist item
7521  =                           trunc25:
```

```
7522
7523  =
7524  =                         endif
7525  =2BE5 E92BE2        0E13          jmp cpydirloc                   ;return - end of truncate
7526  =
7527  =                         trunc3:
7528  =2BE8 E860FF        2b4b          call save_first_dcnt
7529  =2BEB E8FFD9        05E0          call getfcb
7530  =2BEE E84DD9        053E          call dmposition                 ;set up to clear dskmap
7531  =2BF1 FEC0                        inc al
7532  =2BF3 F6061209FF                  test single,0ffh
7533  =2BF8 7502          2BFC          jnz zero_dm1
7534  =2BFA D0E0                          shl al,1
7535  =                         zero_dm1:
7536  =2BFC BF4C0B                      mov di,offset info_fcb+dskmap
7537  =2BFF 32E4                        xor ah,ah
7538  =2C01 03F8                        add di,ax
7539  =2C03 B110                        mov cl,16
7540  =2C05 2AC8                        sub cl,al
7541  =2C07 8AEC                        mov ch,ah                       ;CX = count
7542  =2C09 8AC4                        mov al,ah
7543  =2C0B F3AA                        rep stosb                       ;zero dskmap positions
7544  =2C0D E88AD9        059A          call get_dir_ext
7545  =2C10 3A07                        cmp al,[bx]                     ;infofcb(ext) = dirfcb(ext) after
7546  =2C12 8807                        mov [bx],al                     ; blocks removed
7547  =2C14 9C                          pushf
7548  =2C15 A05C0B                      mov al,info_fcb+nxtrec
7549  =2C18 FEC0                        inc al                          ;fcb(rc) = fcb(cr) + 1
7550  =2C1A BE4B0B                      mov si,offset info_fcb+reccnt
7551  =2C1D 8804                        mov [si],al
7552  =2C1F 9D                          popf
7553  =2C20 7403          2C25          jz $+5                          ;if dirfcb(ext) < infofcb(ext)
7554  =2C22 E89AE6        12BF            call set_rc3                  ; rc = 0 or 128
7555  =2C25 F6061109FF                  test dminx,0ffh
7556  =2C2A 7503          2C2F          jnz $+5                         ;if no blocks remain in fcb
7557  =2C2C E890E6        12BF            call set_rc3                  ; rc = 0
7558  =2C2F E8F3DA        0725          call get_dptra
7559  =2C32 83C30B                      add bx,archiv
7560  =2C35 80277F                      and b[bx],7fh
7561  =2C38 A0480B                      mov al,info_fcb+extnum          ;dirfcb(ext) = infofcb(ext)
7562  =2C3B 884701                      mov 1[bx],al
7563  =2C3E 83C304                      add bx,4
7564  =2C41 BE4B0B                      mov si,offset info_fcb+reccnt
7565  =2C44 8BFB                        mov di,bx
7566  =2C46 B91100                      mov cx,17                       ;dirfcb(rc) and dskmap =
7567  =2C49 F3A4                        rep movsb                       ; infofcb(rc) and dskmap
7568  =2C4B B101                        mov cl,1                        ;restore non-erased block
7569  =2C4D E8F2E0        0D42          call scndmab                    ; indexes in allocation vector
7570  =2C50 E974FF        2BCD          jmp trunc15
7571  =
7572  =                         func100:        ;set directory label
7573  =                         ;=======
7574  =                         ;       dx -> .fcb
```

```
7575
7576  =                       ;       drive location
7577  =                       ;       name & type field's user's discretion
7578  =                       ;       extent field definition
7579  =                       ;       bit 7 (80h): enable passwords on drive
7580  =                       ;       bit 6 (40h): enable file access stamping
7581  =                       ;       bit 5 (20h): enable file update stamping
7582  =                       ;       BIT 4 (10H): ENABLE FILE CREATE STAMPING
7583  =                       ;       bit 0 (01h): assign password to dir lbl
7584  =
7585  =2C53 B110                      mov cl,16
7586  =2C55 E874EE      1ACC          call copy_dma_in
7587  =2C58 E88CED      19E7          call reselectx
7588  =2C5B C6053C0B21                MOV info_fcb,21H                ;SEARCH FOR SFCB'S IN DIRECTORY
7589  =2C60 B101                      MOV CL,1                        ;SFCB(0) = 21
7590  =2C62 E8E0E3      1045          CALL SEARCH
7591  =2C65 7508        2C72          JNZ SDL0                        ;SFCB FOUND
7592  =2C67 BB4809                      mov bx,offset info_fcb+extnum ;NO SFCB'S - DID USER REQUEST
7593  =2C6A F60770                      TEST BEBX],070H               ;DATE & TIME STAMPS?
7594  =2C6D 7403        2C72            JZ SDL0                       ;NO
7595  =2C6F E93DE4      10AF              JMP LRET_EQ_FF              ;YES - ERROR
7596  =                       SDL0:
7597  =2C72 C6063C0B20                mov info_fcb,20h                ;does dir lbl exist on drive?
7598  =2C77 B101                      mov cl,1
7599  =2C79 C7061508FFFF              mov xdcnt,0ffffh                ;initialized for make
7600  =2C7F E8C3E3      1045          call search
7601  =2C82 7517        2C98          jnz sdl1                        ;yes
7602  =2C84 C6061308FF                  mov make_xfcb,0ffh          ;no - make one
7603  =2C89 E852E7      13DE            call make
7604  =2C8C 7501        2C8F            jnz $+3
7605  =2C8E C3                            ret                       ;no directory space
7606  =2C8F E87AEE      1B0C            call init_xfcb
7607  =2C92 B91800                      MOV CX,24
7608  =2C95 E821F0      1CB9            CALL STAMP5                 ;set LABEL create date and time stamp
7609  =2C98 E8FBEF      1C96            call stamp1                 ;SET SFCB CREATE STAMP
7610  =                       sdl1:
7611  =2C9B B91C00                    MOV CX,28
7612  =2C9E E818F0      1CB9          CALL STAMP5                     ;set LABEL update date and time stamp
7613  =2CA1 E8F6EF      1C9A          call stamp2                     ;SET SFCB UPDATE STAMP
7614  =2CA4 E863EF      1C0A          call chk_xfcb_password          ;verify password
7615  =2CA7 7403        2CAC          jz $+5                          ;new dir lbl falls through
7616  =2CA9 E9ECEE      1B98            jmp pw_error
7617  =2CAC 33C9                      xor cx,cx
7618  =2CAE E861EE      1B12          call init_xfcb0                 ;update dir lbl name
7619  =2CB1 8BF2                      mov si,dx                       ;copy user's dir lbl data byte
7620  =2CB3 AC                        lods al                         ;into dir lbl (ex byte)
7621  =2CB4 0C01                      or al,1                         ;set dir lbl exists flag
7622  =2CB6 8807                      mov [bx],al
7623  =2CB8 8B3EA808                  mov di,drvlbla                  ;update drives dir lbl data byte
7624  =2CBC AA                        stos al
7625  =                       sdl2:
7626  =2CBD 4E                        dec si                          ;test for assignment of new
7627  =2CBE AC                        lods al                         ;password to dir lbl or xfcb
```

```
7628
7629  =2CBF 2401                        and al,1
7630  =2CC1 7406          2CC9          jz sd13
7631  =2CC3 BED108                        mov si,offset common_dma+8    ;assign new password
7632  =2CC6 E84AEF        1C13            call set_pw                   ;set password checksum byte
7633  =                   sd13:
7634  =2CC9 E807E3        0FD3          CALL FIX_HASH                   ;FIX DIRECTORY HASH ENTRY
7635  =2CCC E93EDF        0C0D          jmp wrdir                       ;write dir lbl or xfcb to directory
7636  =
7637  =                         func101:        ;return directory label data
7638  =                         ;=======
7639  =2CCF E8E0EC        1982          call tmp_select
7640  =2CD2 E809EE        1ADE          call get_dir_mode
7641  =2CD5 E9D6D7        04AE          jmp set_lret
7642  =
7643  =                         func102:        ;read file xfcb
7644  =                         ;=======
7645  =2CD8 E80CED        19E7          call reselectx
7646  =2CDB E84DE1        0E2B          call check_wild
7647  =2CDE E856E3        1037          CALL search_1_name              ;SEARCH FOR FILE'S 1ST FCB
7648  =2CE1 744E          2D31          JZ ret39                        ;SEARCH UNSUCCESSFUL
7649  =2CE3 B500                        MOV CH,0
7650  =2CE5 E85DEF        1C45          CALL GET_DTBA                   ;GET SFCB ADDRESS
7651  =2CE8 0AC0                        OR AL,AL
7652  =2CEA 7513          2CFF          JNZ RXFCB2                      ;NO SFCB
7653  =2CEC 53                            PUSH BX
7654  =2CED BF4C0B                        mov di,offset info_fcb+dskmap
7655  =2CF0 B90800                        MOV CX,8                      ;ZERO OUT PASSWORD FIELD IN
7656  =2CF3 F3AA                          REP STOSB                     ;USER'S FCB
7657  =2CF5 5E                            pop si
7658  =2CF6 B104                          MOV CL,4
7659  =2CF8 F3A5                          rep movsw                     ;FCB(STAMPS) = SFCB(STAMPS)
7660  =2CFA AC                            LODS AL
7661  =2CFB A24808                        MOV info_fcb+extnum,AL        ;FCB(EXT) = SFCB(PASSWORD MODE)
7662  =2CFE C3                            RET
7663  =                   RXFCB2:
7664  =2CFF E8E3ED        1AE5          call get_xfcb                   ;does xfcb exist for file?
7665  =2D02 B0FF                        mov al,0ffh
7666  =2D04 7503          2D09          jnz $+5
7667  =2D06 E9A5D7        04AE            jmp set_lret                  ;no
7668  =2D09 BB3C0B                      mov bx,offset info_fcb          ;yes - copy xfcb to user's fcb
7669  =2D0C B120                        mov cl,nxtrec
7670  =2D0E E9D1D7        04E2          jmp move
7671  =
7672  =                         func103:        ;write or update file xfcb
7673  =                         ;=======
7674  =2D11 B110                        mov cl,16
7675  =2D13 E8B6ED        1ACC          call copy_dma_in
7676  =2D16 E8CEEC        19E7          call reselectx
7677  =2D19 E8C2ED        1ADE          call get_dir_mode               ;ARE PASSWORDS ACTIVATED IN DRIVE?
7678  =2D1C 2480                        AND AL,80H
7679  =2D1E 7503          2D23          jnz $+5
7680  =2D20 E98CE3        10AF            jmp lret_eq_ff                ;no
```

```
7681
7682  =2D23 E8D5E1        0E2B          call check_wild
7683  =2D26 C7061508FFFF                mov xdcnt,0ffffh              ;initialized for make
7684  =2D2C E814E3        1043          call search_ext               ;does file exist?
7685  =2D2F 7501          2D32          jnz $+3
7686  =2D31 C3                    ret39:  ret                         ;no
7687  =
7688  =                           if BNPM
7689  =2D32 E891F0        1DC6          call tst_olist
7690  =                           endif
7691  =
7692  =2D35 E84DED        1AE5          call get_xfcb                 ;does xfcb exist for file?
7693  =2D38 7503          2D45          jnz wxfcb1                    ;yes
7694  =2D3A A21308                        mov make_xfcb,al            ;make xfcb
7695  =2D3D E89EE6        13DE            call make                   ;does directory space exist for make?
7696  =2D40 74EF          2D31            jz ret39                    ;no
7697  =2D42 E8C7ED        1B0C            call init_xfcb              ;initialize xfcb
7698  =                           wxfcb1:
7699  =2D45 E8C2EE        1C0A          call chk_xfcb_password        ;check password
7700  =2D48 7403          2D4D          jz $+5
7701  =2D4A E94BEE        1B98            jmp pw_error
7702  =2D4D BE480B                      mov si,offset info_fcb+extnum
7703  =2D50 F607FF                      test b[bx],0ffh               ;is current password mode non-zero
7704  =2D53 750B          2D60          jnz wxfcb2                    ;yes
7705  =2D55 AC                          lods al                       ;is user specifying a new password?
7706  =2D56 4E                          dec si
7707  =2D57 2401                        and al,1
7708  =2D59 7505          2D60          jnz wxfcb2                    ;yes
7709  =2D5B E86BFF        2CC9            CALL SDL3
7710  =2D5E EB0C          2D6C            jmps WXFCB4
7711  =                           wxfcb2:
7712  =2D60 AC                    lods al                             ;assign new password mode to xfcb
7713  =2D61 24E0                  and al,0e0h
7714  =2D63 7502          2D67    jnz wxfcb3
7715  =2D65 B080                    mov al,80h
7716  =                           wxfcb3:
7717  =2D67 8807                  mov [bx],al
7718  =2D69 E851FF        2CBD    CALL SDL2                           ;check for new password
7719  =                           WXFCB4:
7720  =2D6C E88EED        1AFD    call get_xfcb1
7721  =2D6F 24E0                  and al,0e0h
7722  =2D71 A2DD08                mov pw_mode,al                      ;UPDATE SFCB WITH NEW PASSWORD MODE
7723  =2D74 50                    push ax
7724  =2D75 E88FE2        1037    CALL search_1_name
7725  =2D78 58                    pop ax
7726  =2D79 A2480B                mov info_fcb+extnum,al              ;set password mode back in user xfcb
7727  =2D7C 7483          2D31    JZ ret39
7728  =2D7E E8C2EE        1C43    CALL GET_DTBA8
7729  =2D81 0AC0                  OR AL,AL
7730  =2D83 75AC          2D31    JNZ ret39
7731  =2D85 A0DD08                MOV AL,pw_mode
7732  =2D88 8807                  MOV [BX],AL
7733  =2D8A E9B00E        0C0D    JMP wrdir
```

```
1734
1735  =
1736  =                         func106:        ;set default password
1737  =                         ;=======
1738  =2D8D B108                        mov cl,8
1739  =2D8F E85BD6        03EA          call parsave
1740  =2D92 C6060C0800                  mov parlg,0
1741  =2D97 BE3C08                      mov si,offset info_fcb
1742  =2D9A 8B0E                        mov bx,si
1743  =2D9C BF9C08                      mov di,offset df_password+7
1744  =2D9F B90800                      mov cx,8
1745  =2DA2 E974EE        1C19          jmp set_pw0
1746  =
1747  =
1748  =                         ;********** end bdos file system part 4 **********
1749
```

```
7750
7751  =                         eject 1 include file5.bdo       ; file system part 5
7752  =
7753  =                       ;************ bdos file system part 5 ************
7754  =
7755  =                       ;       BDOS functions which do not require
7756  =                       ;        ownership of the MXdisk queue
7757  =
7758  =                       func24:         ;return the login vector
7759  =                       ;======
7760  =2DA5 8B1E0808                  mov bx,dlog
7761  =2DA9 C3                        ret
7762  =
7763  =                       func25:         ;return selected disk number
7764  =                       ;======
7765  =
7766  =                       if BCPM
7767  =                               mov bl,p_dsk
7768  =                               ret
7769  =                       endif
7770  =                       if BMPM
7771  =2DAA 8B366800                  mov si,rlr
7772  =2DAE 8A5C12                    mov bl,p_dsk[si]
7773  =2DB1 C3                        ret
7774  =                       endif
7775  =
7776  =                       func26:         ;set the subsequent dma address to info
7777  =                       ;======
7778  =
7779  =2DB2 2689160200                mov u_dma_ofst,dx
7780  =2DB7 C3                        ret
7781  =
7782  =                       func29:         ;return r/o bit vector
7783  =                       ;======
7784  =2DB8 8B1E0608                  mov bx,rodsk
7785  =2DBC C3                        ret
7786  =
7787  =                       func32:         ;set user code
7788  =                       ;======
7789  =
7790  =                       if BCPM
7791  =                               mov al,dl
7792  =                               cmp al,0ffh
7793  =                               jnz setusrcode
7794  =                                 mov bl,p_user                 ;interrogate user code instead
7795  =                                 ret
7796  =                       setusrcode:
7797  =                               and al,0fh
7798  =                               mov p_user,al
7799  =                               ret                           ;jmp goback
7800  =                       endif
7801  =                       if BMPM
7802  =2DBD 8B366800                  mov si,rlr
```

```
7803
7804  =2DC1 80FAFF7504    2DCA          cmp dl,0ffh ! jne setusrcode
7805  =2DC6 8A5C13                        mov bl,p_user[si]
7806  =2DC9 C3                            ret
7807  =                           setusrcode:
7808  =2DCA 80E20F                        and dl,0fh
7809  =2DCD 885413                        mov p_user[si],dl
7810  =2DD0 C3                            ret
7811  =                           endif
7812  =
7813  =                           func44:         ;set multi-sector count
7814  =                           ;======
7815  =2DD1 33DB                          xor bx,bx
7816  =2DD3 0AD2                          or dl,dl
7817  =2DD5 740B          2DE2            jz return_not_ok
7818  =2DD7 80FA81                        cmp dl,129
7819  =2DDA 7306          2DE2            jnb return_not_ok
7820  =2DDC 2688161100                      mov u_mult_cnt,dl
7821  =2DE1 C3                            ret
7822  =                           return_not_ok:
7823  =2DE2 4B                            dec bx                          ;return BX = 0ffffh
7824  =2DE3 C3                            ret
7825  =
7826  =                           func45:         ;set bdos error mode
7827  =                           ;======
7828  =2DE4 2688161000                    mov u_error_mode,dl
7829  =2DE9 C3                            ret
7830  =
7831  =                           func51:         ; set dma base
7832  =                           ;======
7833  =2DEA 2689160400                    mov u_dma_seg,dx
7834  =2DEF C3                            ret
7835  =
7836  =                           func52:         ; get dma
7837  =                           ;======
7838  =2DF0 268B1E0400                    mov bx,u_dma_seg
7839  =2DF5 26891E3000                    mov u_retseg,bx
7840  =2DFA 268B1E0200                    mov bx,u_dma_ofst
7841  =2DFF C3                            ret
7842  =
7843  =                           func104:        ;set current date and time
7844  =                           ;=======
7845  =2E00 8BF2                          mov si,dx
7846  =2E02 BF7E00                        mov di,offset tod
7847  =2E05 B90200                        mov cx,2
7848  =2E08 9CFA                          pushfl cli
7849  =2E0A 061E                          push esl push ds                ;save DS and ES
7850  =2E0C 268E1E2E00                    mov ds,u_wrkseg
7851  =2E11 07                            pop es
7852  =2E12 F3A5                          rep movsw
7853  =2E14 06                            push es
7854  =2E15 1F07                          pop dsl pop es                  ;restore DS and ES
7855  =2E17 C606820000                    mov byte ptr tod+4,0
```

```
7856
7857  =2E1C 9D                          popf
7858  =2E1D C3                          ret
7859  =
7860  =                         func105:        ;get current date and time
7861  =                         ;=======
7862  =2E1E 88FA                        mov di,dx
7863  =2E20 BE7E00                      mov si,offset tod
7864  =2E23 B90200                      mov cx,2
7865  =2E26 06                          push es
7866  =2E27 268E062E00                  mov es,u_wrkseg
7867  =2E2C 9CFA                        pushfl cli
7868  =2E2E F3A5                        rep movsw
7869  =2E30 8A1E8200                    mov bl,tod+4
7870  =2E34 9D                          popf
7871  =2E35 07                          pop es
7872  =2E36 C3                          ret
7873  =
7874  =                         ;********** end.bdos file system part 5 **********
7875
```

```
7876
7877  =                       eject ! include patch.cod
7878  =
7879  =
7880  =                     ;*******************************************************
7881  =                     ;*
7882  =                     ;*      PATCH AREA  --  128 bytes long
7883  =                     ;*
7884  =                     ;*******************************************************
7885  =
7886  =                     patch:
7887  =2E37 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;00-0f
7888  =2E3D 909090909090            nop ! nop ! nop ! nop ! nop ! nop
7889  =2E43 90909090                nop ! nop ! nop ! nop
7890  =
7891  =2E47 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;10-1f
7892  =2E4D 909090909090            nop ! nop ! nop ! nop ! nop ! nop
7893  =2E53 90909090                nop ! nop ! nop ! nop
7894  =
7895  =2E57 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;20-2f
7896  =2E5D 909090909090            nop ! nop ! nop ! nop ! nop ! nop
7897  =2E63 90909090                nop ! nop ! nop ! nop
7898  =
7899  =2E67 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;30-3f
7900  =2E6D 909090909090            nop ! nop ! nop ! nop ! nop ! nop
7901  =2E73 90909090                nop ! nop ! nop ! nop
7902  =
7903  =2E77 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;40-4f
7904  =2E7D 909090909090            nop ! nop ! nop ! nop ! nop ! nop
7905  =2E83 90909090                nop ! nop ! nop ! nop
7906  =
7907  =2E87 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;50-5f
7908  =2E8D 909090909090            nop ! nop ! nop ! nop ! nop ! nop
7909  =2E93 90909090                nop ! nop ! nop ! nop
7910  =
7911  =2E97 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;60-6f
7912  =2E9D 909090909090            nop ! nop ! nop ! nop ! nop ! nop
7913  =2EA3 90909090                nop ! nop ! nop ! nop
7914  =
7915  =2EA7 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;70-7f
7916  =2EAD 909090909090            nop ! nop ! nop ! nop ! nop ! nop
7917  =2EB3 90909090                nop ! nop ! nop ! nop
7918
```

```
7919
7920  =                         eject ! include uda.fmt         ; User Data area
7921  =
7922  =
7923  =                       ;*********************************************************
7924  =                       ;*
7925  =                       ;*      User Data Area - The User Data Area is an
7926  =                       ;*      extension of the process descriptor but it
7927  =                       ;*      travels with the user.  It contains info
7928  =                       ;*      that is needed only while in context.
7929  =                       ;*
7930  =                       ;*      While in the operating system, The Extra
7931  =                       ;*      Segment register points to the beginning
7932  =                       ;*      of the User Data Area.
7933  =                       ;*
7934  =                       ;*********************************************************
7935  =
7936  =                               eseg
7937  =                               org     0
7938  =
7939  =0000                                   RW      1
7940  =                       ;u_dparam       rw      1       ; arg to dispatch
7941  =
7942  =                       ;       this area overlays part of BDOS
7943  =
7944  =0002                  u_dma_ofst       rw      1       ; BDOS dma offset
7945  =
7946  =0004                  u_dma_seg        rw      1       ; BDOS dma segment
7947  =
7948  =0006                  u_func           rb      1       ; actual function number
7949  =0007                                   RB      1
7950  =                      ;u_searchl       rb      1       ; BDOS search length
7951  =0008                                   RW      1
7952  =                      ;u_searcha       rw      1       ; BDOS search FCB offset
7953  =000A                                   RW      1
7954  =                      ;u_searchabase   rw      1       ; BDOS search user's segment
7955  =000C                                   RW      1
7956  =                      ;u_dcnt          rw      1       ; BDOS directory count
7957  =000E                                   RW      1
7958  =                      ;u_dblk          rw      1       ; BDOS directory block #
7959  =0010                  u_error_mode     rb      1       ; BDOS error mode
7960  =0011                  u_mult_cnt       rb      1       ; BDOS multi-sector count
7961  =0012                                   RB      8
7962  =                      ;u_df_password   rb      8       ; BDOS default password
7963  =
7964  =001A                  u_pd_cnt         rb      1       ; BDOS process count
7965  =  0019                uda_ovl_len      equ     (offset $)-(offset u_dma_ofst)
7966  =                      ;       end of overlay area
7967  =
7968  =
7969  =001B                                   RB      1
7970  =                      ;u_in_int        rb      1
7971  =001C                                   RW      1
```

```
7972
7973  =                 ;u_sp           rw      1       ; save register area
7974  =001E                             RW      1
7975  =                 ;u_ss           rw      1
7976  =0020                             RW      1
7977  =                 ;u_ax           rw      1
7978  =0022                             RW      1
7979  =                 ;u_bx           rw      1
7980  =0024                             RW      1
7981  =                 ;u_cx           rw      1
7982  =0026                             RW      1
7983  =                 ;u_dx           rw      1
7984  =0028                             RW      1
7985  =                 ;u_di           rw      1
7986  =002A                             RW      1
7987  =                 ;u_si           rw      1
7988  =002C                             RW      1
7989  =                 ;u_bp           rw      1
7990  =
7991  =002E            u_wrkseg        rw      1       ; curr seg addr of buf
7992  =0030            u_retseg        rw      1       ; usr ES return
7993  =
7994  =0032                             RW      1
7995  =                 ;u_ds_sav       rw      1       ;\
7996  =0034                             RW      1
7997  =                 ;u_stack_sp     rw      1       ; usr stack segment
7998  =0036                             RW      1
7999  =                 ;u_stack_ss     rw      1       ; usr stack pointer
8000  =0038                             RW      10
8001  =                 ;u_ivectors     rw      10      ; save int 0-4
8002  =004C                             RW      1
8003  =                 ;u_es_sav       rw      1       ; > Used during interrupts
8004  =004E                             RW      1
8005  =                 ;u_flag_sav     rw      1       ;/
8006  =0050                             RW      1
8007  =                 ;u_initcs       rw      1
8008  =0052                             RW      1
8009  =                 ;u_initds       rw      1
8010  =0054                             RW      1
8011  =                 ;u_inites       rw      1
8012  =0056                             RW      1
8013  =                 ;u_initss       rw      1
8014  =0058                             RW      1
8015  =                 ;u_mpm_ip       rw      1       ; MPM vec save
8016  =005A                             RW      1
8017  =                 ;u_mpm_cs       rw      1
8018  =005C                             RW      1
8019  =                 ;u_debug_ip     rw      1       ; RTS,Debug Vector Save
8020  =005E                             RW      1
8021  =                 ;u_debug_cs     rw      1
8022  =0060                             RB      1
8023  =                 ;u_insys        rb      1       ; # times through user_entry
8024  =0061                             RB      1
```

```
8025
8026  =
8027  =                       ;u_stat_sav     rb      1
8028  =0062
8029  =0064                   u_conccb        rw      1
8030  =                                       RW      1
8031                          ;u_lstccb       rw      1
```

```
8032
8033  =                         eject ! include sysdat.bdo
8034  =
8035  =                       ;*****************************************************
8036  =                       ;*
8037  =                       ;*      bdos data area
8038  =                       ;*
8039  =                       ;*****************************************************
8040  =
8041  =                         CSEG
8042  =                         org     000Ch
8043  =000C 0600                dw      6       ;development version #
8044  =
8045  =                       ;               SYTEM DATA EQUATES
8046  =
8047  =                                       DSEG
8048  =
8049  =                                       ORG     0
8050  =0000                   SUPMOD          RW      4
8051  =
8052  =                                       ORG     28H
8053  =0028                   XIOSMOD         RW      4
8054  =
8055  =                                       ORG     52H
8056  =0052                   FREE_ROOT       RW      1
8057  =
8058  =                                       ORG     68H
8059  =0068                   RLR             RW      1
8060  =
8061  =                                       ORG     7EH
8062  =007E                   TOD             RB      0
8063  =
8064  =                                       org     88h
8065  =0088                   open_vector     rw      1
8066  =
8067  =                                       ORG     8AH
8068  =008A                   LOCK_MAX        RB      1
8069  =008B                   OPEN_MAX        RB      1
8070  =
8071  =                                       org     92h
8072  =0092                   err_intercept   rw      2
8073  =
8074  =                                       org     0c0eh
8075  =0C0E                   media_flag      rb      1
8076  =
8077
```

```
8078
8079  =                         eject 1 include data.bdo
8080  =
8081  =                 ;*******************************************************
8082  =                 ;*
8083  =                 ;*      bdos data area
8084  =                 ;*
8085  =                 ;*******************************************************
8086  =
8087  =  FFFF           CCPMOFF equ     true
8088  =
8089  =                 if BCPM
8090  =
8091  =                 ;
8092  =                 ;       8086 variables that must reside in code segment
8093  =                 ;
8094  =                 cseg    $
8095  =                 ;
8096  =                 axsave          dw      0       ; register saves
8097  =                 ss_save         dw      0
8098  =                 sp_save         dw      0
8099  =                 stack_begin     dw      endstack
8100  =                 ;
8101  =                 ;       variables in data segment:
8102  =                 ;
8103  =                 dseg    cpmsegment
8104  =                         org     bdosoffset+bdoscodesize
8105  =
8106  =                 header  rs      128
8107  =                         rs      72
8108  =
8109  =                 pag0    dw      0               ;address of user's page zero
8110  =                 ip0     db      0               ;initial page value for ip register
8111  =                 ;
8112  =                 ;     memory control block
8113  =                 ;
8114  =                 umembase        dw      0       ;user'sbase for memory request
8115  =                 umemlg  dw      0               ;length of memory req
8116  =                 contf   db      0               ;flag indicates added memory is avail
8117  =                 ;
8118  =                 ;
8119  =                 hold_info       dw      0       ;save info
8120  =                 hold_spsave     dw      0       ;save user sp during program load
8121  =                 hold_sssave     dw      0       ;save user ss during program load
8122  =
8123  =                 mod8080         db      0
8124  =                 ;
8125  =                 ;       byte i/o variables:
8126  =                 ;
8127  =                 compcol db      0               ;true if computing column position
8128  =                 strtcol db      0               ;starting column position after read
8129  =                 column  db      0               ;column position
8130  =                 listcp  db      0               ;listing toggle
```

```
8131
8132  =                       kbchar  db      0               ;initial key char = 00
8133  =
8134  =                       endif
8135  =
8136  =                               dseg
8137  =                       if BMPM
8138  =                               org     0c00h   ;org for MP/M system
8139  =                       endif
8140  =
8141  =                       if BMPM and CCPMOFF
8142  =                               org     0800h   ;org for CCP/M system
8143  =                       endif
8144  =
8145  =                       if BMPM
8146  =0800 0000              rlog            dw      0       ;removeable logged-in disks
8147  =0802 0000              tlog            dw      0       ;removeable disk test login vector
8148  =0804 0000              ntlog           dw      0       ;new tlog vector
8149  =
8150  =                       endif
8151  =
8152  =0806 0000              rodsk           dw      0       ;read only disk vector
8153  =0808 0000              dlog            dw      0       ;logged-in disks
8154  =
8155  =080A 00                open_fnd        db      0               ;open file found in srch_olist
8156  =
8157  =                       ;the following variables are set to zero upon entry to file system
8158  =
8159  =080B 00                fcbdsk          db      0               ;disk named in fcb
8160  =080C 00                parlg           db      0               ;length of parameter block copied
8161  =080D 0000              aret            dw      0               ;adr value to return
8162  =  080D                 iret            equ     byte ptr aret   ;low(aret)
8163  =080F 00                resel           db      0               ;reselection flag
8164  =0810 00                rd_dir_flag     db      0               ;must read/locate directory record
8165  =0811 00                comp_fcb_cks    db      0               ;compute fcb checksum flag
8166  =0812 00                search_user0    db      0               ;search user 0 for file (open)
8167  =0813 00                make_xfcb       db      0               ;make & search xfcb flag
8168  =0814 00                find_xfcb       db      0               ;search find xfcb flag
8169  =0815 0000              xdcnt           dw      0               ;empty directory dcnt
8170  =0817 00                DIR_CNT         DB      0               ;DIRECT I/O COUNT
8171  =0818 00                MULT_NUM        DB      0               ;MULTI-SECTOR COUNT used in bios
8172  =0819 00                olap_type       db      0               ;record lock overlap type
8173  =081A 00                ff_flag         db      0               ;0ffh xios error return flag
8174  =081B 00                err_type        db      0               ;type of error to print if not zero
8175  =081C                                   rb      4               ;reserved bytes
8176  =  0015                 zerolength      equ     (offset $)-(offset fcbdsk)
8177  =0820 01                mult_sec        db      1               ;multi sector count passed to xios
8178  =0821 00                RELOG           DB      0               ;AUTOMATIC RELOG SWITCH
8179  =
8180  =0822 00                BLK_OFF         DB      0               ;RECORD OFFSET WITHIN BLOCK
8181  =0823 0000              blk_num         dw      0               ;block number
8182  =                       ;
8183  =0825 2708              ua_lroot        dw      offset ua0      ;unallocated block list root
```

```
8184
8185  =                       ;
8186  =0827 20080000         ua0     dw      offset ua1,0
8187  =0826 FF00                     db      0ffh,0
8188  =082D 33080000         ua1     dw      offset ua2,0
8189  =0831 FF00                     db      0ffh,0
8190  =0833 39080000         ua2     dw      offset ua3,0
8191  =0837 FF00                     db      0ffh,0
8192  =0839 3F080000         ua3     dw      offset ua4,0
8193  =083D FF00                     db      0ffh,0
8194  =083F 45080000         ua4     dw      offset ua5,0
8195  =0843 FF00                     db      0ffh,0
8196  =0845 00000000         ua5     dw      0,0
8197  =0849 FF00                     db      0ffh,0
8198  =
8199  =084B                  HASH            RB      4               ;HASH CODE WORK AREA
8200  =084F 00               HASHL           DB      0               ;HASH SEARCH LENGTH
8201  =
8202  =0850 0B               LOG_FXS         db      11              ;number of log_fxs entrys
8203  =                                      ;       fxi15,fxi17,fxi19,fxi22,fxi23
8204  =0851 020406090A                       db      02h  ,04h  ,06h  ,09h  ,0ah
8205  =                                      ;       fxi30,fxi35,fxi99,fxi100,fxi102,fxi103
8206  =0856 111625262829                     db      11h  ,16h  ,25h  ,26h  ,28h  ,29h
8207  =085C 00000000                         db      0,0,0,0         ;reserved
8208  =0860 07               rw_fxs          db      7               ;number of rw_fxs entrys
8209  =                                      ;       fxi20,fxi21,fxi33,fxi34,fxi40,fxi42,fxi43
8210  =0861 070914151B1C                     db      07h  ,08h  ,14h  ,15h  ,1bh  ,1ch  ,1dh
8211        1D
8212  =0868 0000                             db      0,0             ;reserved
8213  =086A 02               sc_fxs          db      2               ;number of sc_fxs entrys
8214  =                                      ;       fxi16,fxi18
8215  =086B 0305                             db      03h  ,05h
8216  =086D 0000                             db      0,0             ;reserved
8217  =
8218  =086F 00               DEBLOCK_FX      DB      0               ;DEBLOCK FUNCTION #
8219  =0870 00               PHY_OFF         DB      0               ;RECORD OFFSET WITHIN PHYSICAL RECORD
8220  =0871 0000             CUR_BCBA        DW      0               ;CURRENT BCB OFFSET
8221  =0873 0000             ROOT_BCBA       DW      0               ;ROOT BCB OFFSET
8222  =0875 0000             EMPTY_BCBA      DW      0               ;EMPTY BCB OFFSET
8223  =
8224  =                       if BNPM
8225  =
8226  =0877 0000             P_LAST_BCBA     DW      0               ;PROCESS'S LAST BCB OFFSET
8227  =0879 00               P_BCB_CNT       DB      0               ;PROCESS BCB COUNT IN BCB LIST
8228  =
8229  =                       endif
8230  =
8231  =087A 00               fx_intrn        db      0               ;internal BDOS function number
8232  =
8233  =087B 0000             TRACK           DW      0       ;BCB RECORD'S TRACK
8234  =087D 0000             SECTOR          DW      0       ;BCB RECORD'S SECTOR
8235  =
8236  =                       ;       seldsk,usrcode are initialized as a pair
```

```
8237
8238  =087F 00              seldsk          db      0       ;selected disk num
8239  =0880 00              usrcode         db      0       ;curr user num
8240  =
8241  =0881 0000            info            dw      0       ;info adr
8242  =0883 0000            searcha         dw      0       ;search adr
8243  =
8244  =                     ;the following variable order is critical
8245  =
8246  =                     ;variables copied from uda for mp/m                     x
8247  =
8248  =                     ;variables included in fcb checksum for mp/m and cp/m       x
8249  =
8250  =                     ;variables used to access system lock list for mp/m           x
8251  =
8252  =0885 0000            dma_ofst        dw      0       ;dma offset             1
8253  =0887 0000            dma_seg         dw      0       ;dma segment            2
8254  =0889 00              func            db      0       ;bdos function #        3
8255  =088A 00              searchl         db      0       ;search len             4
8256  =
8257  =                     if BMPM
8258  =
8259  =088B 0000            searchaofst     dw      0       ;search adr ofst        5
8260  =088D 0000            searchabase     dw      0       ;search adr base        6
8261  =
8262  =                     endif
8263  =
8264  =088F 0000            dcnt            dw      0       ;directory counter      7
8265  =0891 0000            dblk            dw      0       ;directory block        8 ?? - not used - ??
8266  =0893 00              error_mode      db      0       ;bdos error mode        9
8267  =0894 00              mult_cnt        db      0       ;bdos multi-sector cnt 10
8268  =0895                 df_password     rb      8       ;process default pw    11
8269  =
8270  =                     if BMPM
8271  =
8272  =089D 00              pd_cnt          db      0       ;bdos process cnt      12 1
8273  =
8274  =                     endif
8275  =
8276  =089E 00              high_ext        db      0       ;fcb high extent bits     2
8277  =089F 00              xfcb_read_only  db      0       ;xfcb read only flag      3
8278  =08A0 FF              curdsk          db      0ffh    ;current disk             4 1
8279  =
8280  =                     if BMPM
8281  =
8282  =08A1 00              packed_dcnt     db      0       ;packed dblk+dcnt            2
8283  =08A2 00                              db      0
8284  =08A3 00                              db      0
8285  =08A4 0000            pdaddr          dw      0       ;process descriptor addr     3
8286  =
8287  =                     endif
8288  =
8289  =                             org ((offset $) + 1) and 0fffeh
```

```
8290
8291  =
8292  =                         ;       curtrka - alloca are set upon disk select
8293  =                         ;       (data must be adjacent)
8294  =
8295  =08A6 0000               curdrmaxa       dw      0       ;ptr to cur dir max val
8296  =08A8 0000               drvlbla         dw      0       ;drive label data byte addr
8297  =08AA 0000               buffa           dw      0       ;ptr to dir dma addr
8298  =08AC 0000               dpbaddr         dw      0       ;curr disk param block addr
8299  =08AE 0000               checka          dw      0       ;curr checksum vector addr
8300  =08B0 0000               alloca          dw      0       ;curr alloc vector addr
8301  =08B2 0000               DIR_BCBA        DW      0       ;DIRECTORY BUFFER CONTROL BLOCK ADDR
8302  =08B4 0000               DAT_BCBA        dw      0       ;DATA BUFFER CONTROL BLOCK ADDR
8303  =08B6 0000               HASH_SEG        dw      0       ;HASH TABLE SEGMENT
8304  =  000C                  addlist         equ     12      ;"$-buffa" = addr list size
8305  =
8306  =                         ;       sectpt - offset obtained from disk parm block at dpbaddr
8307  =                         ;       (data must be adjacent)
8308  =
8309  =08B8 0000               sectpt          dw      0       ;sectors per track
8310  =08BA 00                 blkshf          db      0       ;block shift factor
8311  =08BB 00                 blkmsk          db      0       ;block mask
8312  =08BC 00                 extmsk          db      0       ;extent mask
8313  =08BD 0000               maxall          dw      0       ;max alloc num
8314  =08BF 0000               dirmax          dw      0       ;max dir num
8315  =08C1 0000               dirblk          dw      0       ;reserved alloc bits for dir
8316  =08C3 0000               chksiz          dw      0       ;size of checksum vector
8317  =08C5 0000               offsetv         dw      0       ;offset tracks at beginning
8318  =08C7 00                 PHYSHF          DB      0       ;PHYSICAL RECORD SHIFT FACTOR
8319  =08C8 00                 PHYMSK          DB      0       ;PHYSICAL RECORD MASK
8320  =08C9                    endlist         rs      0       ;end of list
8321  =  0011                  dpblist         equ     (offset endlist)-(offset sectpt)
8322  =                                                       ;size
8323  =
8324  =                         ;       local variables
8325  =
8326  =08C9                    common_dma      rb      16      ;copy of user's dma lst 16 bytes
8327  =08D9 00                 make_flag       db      0       ;make function flag
8328  =08DA 00                 actual_rc       db      0       ;directory ext record count
8329  =08DB 00                 save_xfcb       db      0       ;search xfcb save flag
8330  =08DC 00                 save_mod        db      0       ;open_reel module save field
8331  =08DD 00                 pw_mode         db      0       ;password mode
8332  =08DE 00                 attributes      db      0       ;fcb interface attributes hold byte
8333  =
8334  =                         if BNPM
8335  =
8336  =                         ;       number of lock list items required for lock operation
8337  =08DF 010002020101       required_table  db      1,0,2,2,1,1,2,2,1,1,2,2,1,1,2,2
8338        020201010202
8339        01010202
8340  =
8341  =08EF 00                 chk_olist_flag  db      0       ;check | test olist flag
8342  =08F0 0000               lock_sp         dw      0       ;lock stack ptr
```

```
8343
8344  =08F2 00              lock_shell      db      0       ;lock shell flag
8345  =08F3 00              check_fcb_ret   db      0       ;check_fcb return switch
8346  =08F4 00              lock_unlock     db      0       ;lock | unlock function flag
8347  =08F5 00              incr_pdcnt      db      0       ;increment process_cnt flag ??
8348  =08F6 00              free_mode       db      0       ;free lock list entries flag ??
8349  =                                                     ;1=free entries for curdsk
8350  =                                                     ;0=free all entries
8351  =08F7 0000            cur_pos         dw      0       ;current position in lock list
8352  =08F9 0000            prv_pos         dw      0       ;previous position in lock list
8353  =
8354  =08FB 00              dont_close      db      0       ;inhibit actual close flag
8355  =08FC 00              open_cnt        db      0       ;process open file count
8356  =08FD 00              lock_cnt        db      0       ;process locked record count
8357  =08FE 0000            file_id         dw      0       ;address of file' lock list entry
8358  =0900 00              set_ro_flag     db      0       ;set drive r/o flag
8359  =0901 00              check_disk      db      0       ;disk reset open file check flag
8360  =0902 00              flushed         db      0       ;lock list open file flush flag
8361  =
8362  =                     ;free_root, lock_max, open_max initialized by sysgen
8363  =
8364  =0903 5200                            dw      offset free_root
8365  =0905 0000            open_root       dw      0       ;lock list open file list root
8366  =0907 0000            lock_root       dw      0       ;lock list locked record list root
8367  =
8368  =                     endif
8369  =
8370  =0909 0000            sdcnt           dw      0       ;saved dcnt of file's 1st fcb
8371  =090B 0000            sdcnt0          dw      0       ;saved dcnt (user 0 pass)
8372  =
8373  =                     if BCPM
8374  =
8375  =                     chain_flag      db      0       ;chain flag ??
8376  =                     tod             db      0ffh,0ffh,0ffh,0ffh ??
8377  =
8378  =                     endif
8379  =
8380  =
8381  =090D 00              rmf             db      0       ;read mode flag for open$reel
8382  =090E 00              wflag           db      0       ;xios/bios write flag
8383  =090F 00              dirloc          db      0       ;directory flag in rename, etc.
8384  =0910 00              linfo           db      0       ;low(info)
8385  =0911 00              dminx           db      0       ;local for diskwrite
8386  =0912 00              single          db      0       ;set true if single byte
8387  =                                                     ;alloc map
8388  =0913 00              rcount          db      0       ;record count in curr fcb
8389  =0914 00              extval          db      0       ;extent num and extmsk
8390  =0915 00              vrecord         db      0       ;curr virtual record
8391  =0916 00              ADRIVE          DB      0       ;CURRENT DISK - must precede arecord
8392  =0917 0000            arecord         dw      0       ;curr actual record
8393  =0919 00                              db      0       ;curr actual record high byte
8394  =091A 0000            ARECORD1        dw      0       ;curr actual block# * blkmsk
8395  =091C 0000            drec            dw      0       ;curr actual directory record
```

```
8396
8397  =091E 0000            CUR_dma_seg     DW      0
8398  =0920 0000            CUR_DMA         DW      0
8399  =
8400  =                     ;       local variables for directory access
8401  =
8402  =0922 00              dptr            db      0       ;directory pointer 0,1,2,3
8403  =  088F               ldcnt           equ     byte ptr dcnt   ;low(dcnt)
8404  =0923 00              user_zero_pass  db      0       ;search user zero flag
8405  =
8406  =                     ;       shell variables
8407  =
8408  =0924 0000            shell_si        dw      0       ;bdos command offset
8409  =0926 000000          shell_rr        db      0,0,0   ;r0,r1,r2 save area
8410  =
8411  =                     ;       special 8086 variables:
8412  =
8413  =0929 0000            uda_save        dw      0       ;user data area saved value
8414  =092B 0000            parametersegment dw     0       ;user parameter segment
8415  =092D 0000            returnseg       dw      0       ;user return segment
8416  =
8417  =                     ;       error messages
8418  =
8419  =092F 00              err_drv         db      0
8420  =0930 0000            err_pd_addr     dw      0
8421  =
8422  =0932 0D0A43502F4D    dskmsg          db      13,10,'CP/M Error On '
8423        204572726F72
8424        204F6E20
8425  =0942 203A2000        dskerr          db      ' : ',0
8426  =
8427  =0946 000060096909    xerr_list       dw      0,permsg,rodmsg,rofmsg,selmsg
8428        78098709
8429  =0950 B309C709DC09                    dw      xe5,xe6,xe7,xe8,xe9,xe10,xe11,xe3
8430        EB09FF09100A
8431        290A9509
8432  =
8433  =0960 446973632049    permsg          db      'Disk I/O',0
8434        2F4F00
8435  =0969 526561642F4F    rodmsg          db      'Read/Only Disk',0
8436        6F6C79204469
8437        736B00
8438  =0978 526561642F4F    rofmsg          db      'Read/Only File',0
8439        6E6C79204669
8440        6C6500
8441  =0987 496E76616C69    selmsg          db      'Invalid Drive',0
8442        642044726976
8443        6500
8444  =
8445  =0995 46696C65204F    xe3             db      'File Opened in Read/Only Mode',0
8446        70656E656420
8447        696E20526561
8448        642F4F6E6C79
```

```
8449
8450            20436F646500
8451    =
8452    =09B3 46696C652043      xe5             db      'File Currently Open',0
8453          757272656E74
8454          6C79204F7065
8455          6E00
8456    =09C7 436C6F736520      xe6             db      'Close Checksum Error',0
8457          436865636B73
8458          756D20457272
8459          6F7200
8460    =09DC 50617373776F      xe7             db      'Password Error',0
8461          726420457272
8462          6F7200
8463    =09EB 46696C652041      xe8             db      'File Already Exists',0
8464          6C7265616479
8465          20457869737473
8466          7300
8467    =09FF 496C6C656761      xe9             db      'Illegal ? in FCB',0
8468          6C203F20696E
8469          2046434200
8470    =0A10 4F70656E2046      xe10            db      'Open File Limit Exceeded',0
8471          696C65204C69
8472          6D6974204578
8473          636565646564
8474          00
8475    =0A29 4E6F20526F6F      xe11            db      'No Room in System Lock List',0
8476          6D20696E2053
8477          797374656D20
8478          4C6F636B204C
8479          69737400
8480    =
8481    =0A45 0D0A00            crlf_str        db      13,10,0
8482    =0A48 0D0A42646F73      pr_fx           db      13,10,'Bdos Function = '
8483          2046756E6374
8484          696F6E203D20
8485    =0A5A 202020            pr_fx1          db      '   '
8486    =0A5D 2046696C6520      pr_fcb          db      ' File = '
8487          3D20
8488    =0A65                   pr_fcb1         rs      12
8489    =0A71 00                                db      0
8490    =
8491    =0A72 0D0A44697361      deniedmsg       db      13,10,'Disk reset denied, Drive '
8492          207265736574
8493          2064656E6965
8494          642C20447269
8495          766520
8496    =0A8D 003A              denieddrv       db      0,':'
8497    =0A8F 20436F6E736F                      db      ' Console '
8498          6C6520
8499    =0A98 00                deniedcns       db      0
8500    =0A99 2050726F6772                      db      ' Program '
8501          616D20
```

```
8502
8503  =0AA2 313233343536       deniedprc       db      '12345678',0
8504        373800
8505  =
8506  =                        if BMPM
8507  =
8508  =                        ;       bdos stack switch variables and stack
8509  =                        ;       used for all bdos disk functions
8510  =
8511  =                                org     ((offset $) + 1) and 0fffeh
8512  =
8513  =                                ; 69 word bdos stack
8514  =
8515  =0AAC CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8516  =0AB2 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8517  =0AB8 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8518  =0ABE CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8519  =0AC4 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8520  =0ACA CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8521  =0AD0 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8522  =0AD6 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8523  =0ADC CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8524  =0AE2 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8525  =0AE8 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8526  =0AEE CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8527  =0AF4 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8528  =0AFA CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8529  =0B00 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8530  =0B06 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8531  =0B0C CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8532  =0B12 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8533  =0B18 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8534  =0B1E CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8535  =0B24 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8536  =0B2A CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8537  =0B30 CCCCCCCCCCCC                       dw      0cccch,0cccch,0cccch
8538  =0B36                    bdosstack       rw      0
8539  =
8540  =0B36                    save_sp         rw      1
8541  =0B38                    ss_save         rw      1
8542  =0B3A                    sp_save         rw      1
8543  =
8544  =                        ;       local buffer area:
8545  =
8546  =0B3C                    info_fcb        rb      40      ;local user FCB
8547  =0B64                    save_fcb        rb      16      ;fcb save area for xfcb search
8548  =
8549  =0B74 0000              mxdiskqd        dw      0       ;link
8550  =0B76 0000                               db      0,0     ;net,org
8551  =0B78 0300                               dw      qf_keep+qf_mx   ;flags (MX queue)
8552  =0B7A 4D5864697368               db      'MXdisk  '
8553        2020
8554  =0B82 00000100                           dw      0,1     ;msglen,nmsgs
```

```
8555
8556  =0886 00000000                    dw      0,0       ;nq,dq
8557  =088A 01000000                    dw      1,0       ;msgcnt,out
8558  =088E 0000                        dw      0         ;buffer ptr
8559  =
8560  =0890 00                  mxdiskqpb db      0         ;flgs
8561  =0891 00                          db      0         ;net
8562  =0892 7408                        dw      mxdiskqd          ;qaddr
8563  =0894 0100                        dw      1         ;nmsgs
8564  =0896 0000                        dw      0         ;buffer
8565  =0898 4D5864697368                db      'MXdisk  '
8566        2020
8567  =
8568  =                         ;       Error Message sync param block for mutual exclusion
8569  =
8570  =08A0 0000                msg_spb         dw      0         ;link   error Message sync parameter block
8571  =08A2 0000                                dw      0         ;owner
8572  =08A4 0000                                dw      0         ;wait
8573  =08A6 0000                                dw      0         ;next
8574  =                         endif
8575  =
8576  =                         if BCPM
8577  =
8578  =                         ;
8579  =                         ;       special 8086 variables:
8580  =                         ;
8581  =                         ioloc           db      0         ;iobyte
8582  =                         user_parm_seg   dw      0         ;holds user parameter seg during load
8583  =                         nallocmem       db      0         ;no. of allocated memory segments
8584  =                         ncrmem          db      0         ;no. of available memory segments
8585  =                         crmem           dw      0,0       ;memory table (16 elements)
8586  =                                         dw      0,0,0,0,0,0
8587  =                                         dw      0,0,0,0,0,0,0,0
8588  =                                         dw      0,0,0,0,0,0,0,0
8589  =                                         dw      0,0,0,0,0,0,0,0
8590  =                         ;
8591  =                         mem_stack_length        equ     40
8592  =                         memstack        rs      mem_stack_length
8593  =                                         ;8 possible allocations
8594  =                         stbase  equ     word ptr 0
8595  =                         stlen   equ     word ptr 2
8596  =                         ccpflag equ     byte ptr 4
8597  =                         nccpalloc       db      0         ;number of current ccp allocations
8598  =                         mem_stk_ptr     dw      0         ;current memory stack location
8599  =
8600  =                         stackarea       rw      ssize     ;stack size
8601  =                         endstack        rb      0         ;top of stack
8602  =                         ;
8603  =                         endif
8604  =                                         org     0fffh
8605  =                         if CCPMOFF
8606  =                                         org     0bfffh
8607  =                         endif
```

```
8608
8609  =0BFF 00                              db      0
8610
8611                                  end
8612
8613
8614  END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR: 78%
```

```
CHKAM3              1A74 L  4933  4936#
CHKAM4              1A77 L  4931  4938#
CHKAM5              1A79 L  4925  4940#
CHKEXITFXS          0786 L  1834#  1909  2477  4970
CHKEXT              10DE L  3318  3329#
CHKINVFCB           0765 L  1582  1760#  1767  3845  4087
CHKMEDIA2           06E7 L  1610#  1841
CHKOLIST            1DC0 L  3491  5522#  6993  7057  7478
CHKOLIST0           1DD2 L  5520  5525#
CHKOLIST01          1DF0 L  5538  5540#
CHKOLIST02          1E0C L  5550  5554#
CHKOLIST05          1E23 L  5541  5543  5552  5570#
CHKOLIST07          1E26 L  5565  5573#
CHKOLIST1           1E2C L  5581#  5582
CHKOLISTFLAG        08EF V  5519  5524  5555  8341#
CHKOPEN1            1D72 L  5459#  5464
CHKOPEN2            1D82 L  5461  5466#
CHKPASSWORD         18C0 L  5113#  6966  7046  7458
CHKPWE1             18BA L  5074  5081  5083  5087  5089#
CHKPWE2             18B9D L  5072  5079  5097#
CHKPWE3             18B4 L  5090  5100  5102  5106#
CHKPWERROR          1832 L  3471  5055#  6431  6851  6968  7048  7460
CHKREC              1F69 L  5738#  5768  5773
CHKREC1             1F7A L  5744  5747  5750#
CHKSIZ              08C3 V  1824  1892  2558  2801  3706  4847  8316#
CHKSUM              000D N    92#  1534  1620  1622  1630  1641  3313  3891  6357  7068
CHKSUMFCB           0666 L  1503  1508#  1618  1628
CHKWILD             0E19 L  2884#  2905  2947  3461
CHKWILD1            0E1F L  2892#  2897
CHKXFCBPASSWORD     1C0A L  3469  5166#  7614  7699
CHKXFCBPASSWORD1    1C0D L  5172#  6849
CIO                 0004 N   368#   499
CKRODIR             0738 L  1704#  3456  6989  7476
CKROFILE            073B L  1708#  4472
CLOSE               139E L  3827#  3945  4126  4697  6563  6589
CLOSE00             25C0 L  6553  6565#
CLOSE000            25AF L  6558#  6573
CLOSE01             25CC L  6562  6577#
CLOSE02             25D4 L  6564  6580#
CLOSE025            2621 L  6619  6622  6625#
CLOSE03             2632 L  6603  6611  6614  6624  6628  6631#
CLOSE1              13B2 L  1589  3844#  4207  6579
CLOSEFCB            12E5 L  3704#  3861
CLOSEFCB1           12F1 L  3707  3709#
CLREXT              0757 L  1739#  4884  6660  6769
CLRMODNUM           0751 L  1734#  6270  6661  6770
CMPEC0              0639 L  1466#  1491  1517  1521  1526
CMPEC1              064A L  1488#  1494
CMPEC2              063B L  1474#  1478
CMPECS              0641 L  1481#  2565
CMPPW               18CC L  5121#  5174  6429
CMPPW1              18DA L  5134#  5139  5141
CMPPW2              18E7 L  5129  5143#
```

```
CURSELECT       1986 L  1857  2412  4823# 4997  4942  6652  7013  7078
DAIBCBA         08B4 V  1927  2301  2346  2583  2807  4928  7413  8302#
DBLK            0891 V  8265#
DCNT            088F V  1777  1786  1794  1804  1859  2423  2516  2513  3037  3072
                        3113  3115  3156  3186  3334  3422  4955  4968  4975  5225
                        5526  6668  6675  6677  6696  8264# 8403
DEACT1          0882 L  1968# 1988
DEACT11         08A3 L  1981  1983#
DEACT2          08A6 L  1971  1976  1982  1985#
DEACTIVATE      087C L  1961# 7411  7414
DEBLKIO1        0A4B L  2270  2274#
DEBLKIO2        0A4E L  2269  2276#
DEBLOCK         0ADC L  2295  2297  2304  2344  2349#
DEBLOCK0        030C L  2372  2375#
DEBLOCK01       0B21 L  2379  2384#
DEBLOCK1        0B32 L  2388  2391#
DEBLOCK15       0B3C L  2393  2396#
DEBLOCK2        0B6E L  2386  2395  2413#
DEBLOCK25       0B8A L  2383  2419  2422  2426#
DEBLOCK4        0B96 L  2425  2432#
DEBLOCK45       0B9F L  2390  2436#
DEBLOCK6        0BB9 L  2444  2447#
DEBLOCK7        0BD4 L  2454  2458#
DEBLOCK9        0BDD L  2057  2385  2433  2465#
DEBLOCKDIR      0A66 L  2290# 2500
DEBLOCKDTA      0A79 L  2287  2299# 4634  7433
DEBLOCKFLUSH    0A88 L  2306# 2347
DEBLOCKFLUSH0   0A90 L  2310# 2335
DEBLOCKFLUSH1   0AB6 L  2313  2316  2321  2327  2332#
DEBLOCKFX       096F V  2358  2377  2414  2442  8218#
DEBLOCKID       0A3B L  2259# 2409  2424  2430
DELET           1188 L  3433# 6735
DELETE00        1195 L  3444# 3475
DELETE01        119C L  3448  3464#
DELETE02        11CD L  3459  3462  3468  3470  3473#
DELETE10        11D5 L  3480# 3514  6887
DELETE11        11DA L  3463  3481  3483# 5105  6982
DELETE12        11EF L  3487  3498#
DELETE13        11F2 L  3496  3500#
DELETE14        11FC L  3503  3505#
DELETE15        1207 L  3508  3511#
DELETEITEM      1D50 L  5444# 5502  5568  5575  5598  5617  5647  6030  6626
DELETEX         118B L  3436# 3472
DENIEDCNS       0A98 V   886  8499#
DENIEDDRV       0A8D V   892  8496#
DENIEDERR       02E5 L   828   881#
DENIEDMSG       0A72 V   893  8491#
DENIEDPRC       0AA2 V   888  8503#
DENIEDTYP       0236 L   794   804#
DEVVER          000C V   559#
DFPASSWORD      0895 V  5162  7743  8268#
DIOC            0802 L  1886  1899#
DIOC0           0830 L  1899  1905#
```

```
DIOC1             0832 L  1903  1908#
DIOC15            0842 L  1911  1914#
DIOC2             0848 L  1891  1893  1896  1918#
DIOC3             0850 L  1916  1920  1922#
DIOCOMP           07F8 L  1872  1891#
DIOCOMPO          07F6 L  1878#  7362  7427
DIRBCBA           0882 V  1933  2293  4593  7410  8301#
DIRBLK            08C1 V  2720  2814  8315#
DIRCNT            0817 V  4238  4242  4363  8170#
DIRLOC            090F V  2879  3196  3374  8383#
DIRMAX            08BF V  2515  3035  7034  8314#
DIRREC            0004 N    66#   68
DIRTOUSER         2696 L  6682  6689#
DISCARD           085F L  1935#  2584  2808
DISCARD0          0861 L  1929  1939#
DISCARD1          0867 L  1947#  1958
DISCARD2          0874 L  1952  1955#
DISCARDDATA       0853 L  1925#  4614  4639
DISCARDDIR        085B L  1931#  2809  4953
DISKEOF           1777 L  4411  4416  4429  4446#
DISKREAD          1722 L  4376#  6748  7099
DISKREAD0         1730 L  4390#  4402  4426
DISKRESET         2271 L  6117#  6214  7163
DISKSELECT        1998 L  4798#  4837  4927
DISKSELECT1       199E L  2406  4804#
DISKWR10          1825 L  4571#
DISKWRITE         177A L  4450#  6761  7114  7289
DISKWRITE1        17C4 L  4494#  4510
DISKWRITE2        17CD L  4498  4502#
DLOG              0808 V  1894  4806  4843  4921  6227  7172  7185  7365  7760  8153#
DMAOFST           0885 V   728   733   735   776   979  2443  2507  4936  6701  7348
                         8252#
DMASEG            0887 V   735  2451  2505  4989  6704  7350  8253#
DMINX             0911 V  1333  1384  3672  4299  4521  4553  7555  8385#
DMPOSITION        053E L  1282#  1332  4297  4519  7530
DMSET             133B L  3745  3763#
DOESXFCBEXIST     1158 L  3406#  6980  7005
DONTCLOSE         08FB V  1588  3806  3833  4205  4220  4386  4490  5770  6578  8354#
DONTSAVE          10ED L  3333  3336#
DONTWRITE         18CA L  4631  4635  4651#
DPBADDR           08AC V  4775  4778  7080  8298#
DPBLIST           0011 N  4780  8321#
DPTR              0922 V  1687  1860  2535  8402#
DREC              091C V  2496  2557  2566  8395#
DRV               0000 N    77#   949
DRVLBL            0E08 L  2656  2867#
DRVLBLA           08A8 V  1527  1533  1639  2803  2821  2869  4772  4947  5000  7623
                         8296#
DRVRELOG          07DC L  1856#  4969
DS                SREG V   725   726   731   867   869   876   889  1021  1022  1024
                         1103  2292  2453  2459  2462  3052  3074  3163  4603  4988
                         4989  4992  5613  5622  6241  6241  6242  6243  7014  7079
                         7349  7350  7353  7849  7850  7854
```

```
DSKERR          0942 V    830  3425#
DSKMAP          0010 N     95# 1314  1762  2711  2713  3603  3714  3716  3717  4551
                         6997  7536  7654
DSKMSG          0932 V    831  8422#
DSKMSK          0003 N     68# 2528  5226  6669  6697
DSKRECL         0080 N    115#
DSKRST1         228A L   6130# 6134
DSKRST2         228E L   6132# 6176
DSKRST3         22A4 L   6131  6141#
DSKRST4         22D8 L   6155# 6168
DSKRST45        22F1 L   6161  6165#
DSKRST5         22FA L   6159  6169#
DSKRST6         2302 L   6157  6164  6173#
DSKRST7         2325 L   6184  6189#
DSKSHF          0002 N     67# 2495
DSKWI1          189C L   4574  4579  4629#
DSKWR0          178C L   4482  4485  4487#
DSKWR1          181E L   4501  4567#
DSKWR2          18F4 L   4655  4687#
DSKWR2A         1912 L   4694  4699#
DSKWR3          1921 L   4706  4710#
DSKWRU          181C L   4559  4565#
EMPTY           00E5 N     63# 2851  3262  3276  3877  7507
EMPTYBCBA       0875 V   2015  2030  2042  2126  8222#
ENDDIR          FFFF N     58# 1786
ENDLIST         08C9 V   8320# 8321
ENDOFDIR        0775 L   1773# 2794  2838  3254  6351  6791  6864
ENDSEARCH       1117 L   3304  3364#
ENDSEARCH1      112B L   3366  3372#
ENTRY           0004 L    554   656#
ERRDRV          092F V    807  1204  6185  8419#
ERRINTERCEPT    0092 V    820   822  8072#
ERRORMODE       0893 V   1147  1195  1205  6183  8266#
ERRPDADDR       0930 V    805  6187  8420#
ERRTYPE         081B V    789  1202  6188  8174#
ES              SREG V    722   723   725   726   772   775   867   868   872   876
                          889   889   892  1056  1056  1058  1076  1076  1078  1100
                         1103  2453  2460  2462  3040  3053  3060  3074  3084  3096
                         3110  3159  3164  5613  5618  5622  6703  6704  6706  7849
                         7851  7853  7854  7865  7866  7871
EXIT            00F3 L    670   674#
EXTMSK          08BC V   1399  1420  1446  3006  5251  8312#
EXTNUM          000C N     91#  944  1405  1447  1741  2868  3205  3215  3317  3416
                         3773  3784  3785  3862  3936  3958  4011  4099  4119  4139
                         4187  4728  5024  5249  5318  5326  6657  6975  6998  7062
                         7484  7485  7561  7592  7661  7702  7726
EXTVAL          0914 V   1292  1448  8389#
F1              0001 N     78#
F5              0005 N     82#
F7              0007 N     84#  938  4868
F8              0008 N     85#  941  1347
FALSE           0000 N    113#  548  1563  1879  2784  2792  3245  3251  3279  3357
                         4452  4957  5519  5760  6273  7111  7236  7284  7301
```

```
FCBDSK          0808 V   739   948  4888  8159# 8176
FCBLEN          0020 N   624   66    97  2713  3717  3773  3899  3914
FCBNZERO        1315 L  3732  3735#
FCBSHF          0005 N    69# 2530
FCCONATTCH      00A2 N   472# 1043
FCCONPRINT      040E N   499# 1047
FCONSTAT        000B N   398#
FFFLAG          081A V  1887  1910  8173#
FILEEXISTS      0482 L  1182# 6795  6979
FILEID          08FE V  5691  5824  5921  5974  6617  6629  8357#
FILL00          1851 L  4595# 4598
FILL2           1878 L  4611# 4625
FINDXFCB        0814 V  3092  3287  3401  8168#
FIXHASH         0FD3 L  2845  3148# 3512  3922  7000  7508  7634
FIXOL1          224F L  6102# 6115
FIXOLISTITEM    2237 L  3984  4167  6090# 6938
FLUSH           2836 L  6595  7423#
FLUSH0          2AD2 L  5601  7359  7364# 7444
FLUSH1          2AD8 L  7367# 7420
FLUSH1A         2AF6 L  7379  7387#
FLUSH1B         2AFD L  7374  7392#
FLUSH2          2817 L  7396  7401#
FLUSH3          2821 L  7394  7406  7409#
FLUSH35         2828 L  7400  7412#
FLUSH4          282F L  7383  7389  7399  7415#
FLUSH6          2831 L  7370  7418#
FLUSHED         0902 V  2784  2792  5607  8360#
FLUSHFILE0      1E5F L  2587  5609#
FLUSHFILE1      1E65 L  5612# 5619  5623  5624
FLUSHFILES      1E55 L  2606  5604#
FLUSHX          283E L  4367  7397  7428#
FQREAD          0089 N   447#  690
FQWRITE         008B N   449#  810
FREEDRV1        2A36 L  7238  7242#
FREEDRV2        2A41 L  7246# 7254
FREEDRV25       2A51 L  7248  7253#
FREEDRV3        2A59 L  7241  7256#
FREEDRV35       2A63 L  7235  7259  7261#
FREEFILES       1E93 L  5626# 7240  7251
FREEFILES1      1E99 L  5634# 5640  5648
FREEFILES2      1EB6 L  5641  5643#
FREEFILES3      1EBB L  5642  5645#
FREEMODE        08F6 V  5635  7239  7243  8348#
FREEROOT        0052 V  5453  5454  5655  5658  5718  6463  6823  8056# 8364
FREESPB         0307 L   873   880   895#
FSYNC           0215 N   481#  817
FTERMINATE      008F N   453#  911
FUNC            0889 V  8254#
FUNC100         2C53 L   646  7572#
FUNC101         2CCF L   647  7637#
FUNC102         2CD8 L   648  7643#
FUNC103         2D11 L   649  7672#
FUNC104         2E00 L   650  7843#
```

```
FUNC105          2C1E L    651  7860#
FUNC106          2D8D L    652  7736#
FUNC13           232D L    607  6205#
FUNC13CONT       234B L   6217  6232#
FUNC14           2364 L    608  6251#
FUNC14A          236A L   6237  6255#
FUNC15           2372 L    609  6267#
FUNC16           259F L    610  6548#
FUNC17           2633 L    611  6637#
FUNC18           26B9 L    612  6710#
FUNC19           26DC L    613  6732#
FUNC20           26E2 L    614  6738#
FUNC21           26E8 L    615  6751#
FUNC22           26EE L    616  6764#
FUNC23           2849 L    617  6947#
FUNC24           2DA5 L    618  7758#
FUNC25           2DAA L    619  7763#
FUNC26           2DB2 L    620  7776#
FUNC27           28AA L    621  7011#
FUNC28           28B7 L    622  7019#
FUNC29           2DB8 L    623  7782#
FUNC30           28DE L    624  7040#
FUNC31           291C L    625  7076#
FUNC32           2DBD L    626  7787#
FUNC33           292C L    627  7085#
FUNC34           2939 L    628  7101#
FUNC35           2946 L    629  7116#
FUNC36           2973 L    630  5884  7146#
FUNC37           2983 L    631  7159#
FUNC38           29B7 L    632  7193# 7266#
FUNC39           2A16 L    633  7232# 7268#
FUNC40           2A65 L    634  7273#
FUNC40A          2A70 L   7286  7288#
FUNC42           2A73 L    635  7294# 7307#
FUNC43           2A7B L    636  7299# 7309#
FUNC44           2DD1 L    637  7813#
FUNC45           2DE4 L    638  7826#
FUNC46           2A83 L    639  7315#
FUNC48           2AC3 L    640  6248  7262  7356#
FUNC49           04B1 L   7437#
FUNC50           04B1 L   7439#
FUNC51           2DEA L    641  7831#
FUNC52           2DF0 L    642  7836#
FUNC98           2AD2 L    644  7444#
FUNC99           2B55 L    645  7453#
FUNCRET          04B1 L   1178# 5306  7266  7268  7307  7309  7437  7439
FUNCTAB          004A V    607#  666
FUNSYNC          0216 N    482#  899
FWFMSK           0080 N     72# 1754  3842  4690  4708
FX1100           0026 N    646
FX1102           0028 N    648
FX1103           0029 N    649
FX1143           002D N    653  7395
```

```
INIT                0049 L   553   5747#
INIT1               0DA2 L  2802  2806#
INIT2               0DD9 L  2835# 2849  2852  2866
INIT23              0DE3 L  2805  2840#
INITCS              0CAF L  2570  2609#
INITIAL3            0E03 L  2858  2864# 2871
INITIALIZE          0D5D L  2771# 4841
INITIALIZE1         0D8D L  2781  2787  2793  2797#
INITXFCB            1B0C L  5030# 6890  7605  7697
INITXFCB0           1B12 L  5034# 7618
INITXFCBSEARCH      114D L  3393# 6969  6984
INITXFCBSEARCH1     114F L  3397# 3413  3438
INTRNLDISKRESET     227D L  6125# 7027
IOFLUSH             000C N   525# 7360  7425
IOREAD              000A N   521# 1876
IOSELDSK            0009 N   520# 4764
IOWRITE             000B N   522# 1669
JMPSETARET          1791 L  4464# 4469
LDCNT               088F V  2527  8403#
LDIABSIZ            00AA N   128#
LEFTIST             1225 L  3539# 3562
LINFO               0910 V   747  7398  8384#
LOCK                2087 L  5907# 7297  7302
LOCK01              20AA L  5918  5920#
LOCK01A             20BF L  5924  5926  5928#
LOCK01B             20D8 L  5933  5940#
LOCK01C             20E1 L  5939  5943#
LOCK02              214A L  5989  5991#
LOCK02A             20E7 L  5949# 5955  5960  5962
LOCK03A             20FE L  5953  5958#
LOCK04A             210A L  5952  5963#
LOCK05              214F L  5990  5993#
LOCK05A             211E L  5969  5970  5972#
LOCK06A             218F L  6022# 6027  6031  6046  6053  6074
LOCK07A             21A8 L  6029  6032#
LOCK07B             21CD L  6034  6047#
LOCK07C             21DD L  6049  6054#
LOCK08A             2214 L  6019  6025  6075#
LOCK09A             2236 L  6077  6087#
LOCK2               2164 L  6003# 6012
LOCK4               217C L  5999  6001  6011  6013#
LOCKCNT             08FD V  8356#
LOCKERR             1FAE L  5775#
LOCKERR1            1FAF L  5772  5774  5779#
LOCKMAX             008A V  5990  8068#
LOCKROOT            0907 V  5385  5493  5886  5947  5975  6020  8366#
LOCKSHELL           08F2 V  1154  5757  5760  8344#
LOCKSP              08F0 V  1157  5756  8342#
LOCKUNLOCK          08F4 V  5923  5954  5968  5998  6076  7296  7301  8346#
LOGFXS              0850 V  1826  8202#
LRET                080D V  1177  1591  3375  3830  3948  4129  4147  4161  4191  4209
                            4415  4484  5018  5324  5762  5941  6008  6592  6694  6698
                            7141  7388  8162#
```

```
NOMX              00EC L   668   671#
NOPBLOCK          17E6 L  4524  4530#
NOSELECT          1A2E L  4894  4896#
NOSELECT0         1A37 L  4901# 6653
NOSELECT1         1A41 L  4905  4905#
NOSHELL           01D9 L   758   761   764#
NOSHELLERR        0381 L   970   975#
NOTCTLC           0133 L   700   704#
NOTERR            1608 L  4211  4215#
NOTFCB33          0189 L   751   753#
NOTFCB36          01C4 L   752   754   756#
NOTMODNUM         10D6 L  3321  3323#
NOTUSER0          110A L  3346  3352#
NOUPDATE          1912 L  4689  4703#
NOWRITE           0716 L  1662# 1718  2579  3637  5086  5099  5269  6419
NTLOG             0804 V  6122  6136  6166  8148#
NXTREC            0020 N    97#   98  1372  1433  1459  3629  4080  6353  7067  7152
                          7548  7669
OFF               0000 N    49#
OFFSETV           08C5 V  1252  8317#
OLAPTYPE          0819 V  5961  5966  6017  6456  6491  6524  8172#
ON                FFFF N    48#
OPEN              127D L  3612# 4146  6323  6785
OPEN1             1280 L  3615# 6345
OPENCNT           08FC V  5697  5706  5708  7213  8355#
OPENCOPY          1282 L  3618# 3996
OPENERR           14BB L  3964  3978  3981  4009#
OPENFILE          239B L  6275  6281  6284#
OPENFND           080A V  8155#
OPENMAX           008B V  5709  7214  8069#
OPENR0            1482 L  3954  3967#
OPENR1            14A8 L  3975  3989#
OPENR2            14B0 L  3988  4002# 4039
OPENR3            14D4 L  3943  4023#
OPENR4            14F4 L  4028  4030  4036#
OPENREEL          1449 L  3928# 4412  4483
OPENROOT          0905 V  5381  5458  5588  5610  5631  5698  6099  6126  6153  6526
                          7234  8365#
OPENUSERZERO      2400 L  6283  6339#
OPENVECTOR        0088 V  5467  5679  8065#
OPENX             2414 L  6324  6346  6349#
OPENX0            2457 L  6365  6370  6373  6375  6379  6387  6391#
OPENX00           2445 L  6368  6381#
OPENX01           24DF L  6461# 6499
OPENX02           24E6 L  6465# 6825  7207
OPENX03           24EB L  6464  6468# 6936
OPENX034          24F0 L  6471# 7230
OPENX035          24F5 L  6470  6474#
OPENX04           24FA L  6460  6477#
OPENX041          250A L  6479  6484#
OPENX042          2526 L  6486  6488  6492#
OPENX05           2544 L  6500# 6513
OPENX06           2549 L  5571  6481  6494  6503# 6512  6832
```

```
OPENX07             254E L  6495  6510#
OPENX08             2555 L  6476  6483  6502  6514#
OPENX09             2567 L  6517  6521#
OPENX09A            2578 L  6528# 6532
OPENX09B            2589 L  6523  6525  6530  6533#
OPENX0A             2492 L  6404  6407  6423#
OPENX0B             249C L  6413  6428#
OPENX1              24B7 L  6427  6430  6434  6439#
OPENX101            24D8 L  6453  6455  6458#
OPENX1A             24BD L  6402  6410  6417  6420  6422  6441  6444# 6940
OPENX2              2597 L  6543# 6916
OPENXA              2426 L  6355  6359#
PACKEDDCNT          08A1 V  5336  5337  5672  5735  6112  7195  7196  8282#
PACKSDCNT           1CF0 L  1595  5333# 5529  6104  6109  6447  6600  6819  6935
PARAMETERSEGMENT    092B V   718   772   795  1022  6641  6720  8414#
PARLG               080C V   767  1017  6520  7740  8160#
PARRET              03FF L   987  1025# 1046
PARSAVE             03EA L  1006  1012# 7739
PARSAVE33           03E4 L   752  1003# 6723
PARSAVE36           03E8 L   755  1008#
PATCH               2E37 L  7877  7886#
PBCBCNT             0879 V  2019  2050  2095  2141  8227#
PCMOD               0018 N   199# 1548
PCNS                0020 N   204#  884
PDADDR              08A4 V   712  1547  5513  5594  5623  5638  5703  5866  5917  5959
                            6026  8285#
PDCNT               089D V   742  1515  1579  5579  5583  8272#
PDCNTOK             01A0 L   742   745#
PDLEN               0030 N   123#
PDSK                0012 N   194#  714  6259  6262  7767  7772
PERERROR            0467 L  1122# 1921
PERMSG              0960 V  8427  8433#
PFCTLC              0080 N   242#  699   909  1211  1215
PFLAG               0006 N   191#  685   687   699   907   909  1211  1215
PFTEMPKEEP          0200 N   244#  685   687   906
PHYMSK              08C8 V  2085  2245  2360  4246  4585  7429  8319#
PHYOFF              0870 V  2065  2150  2363  2438  8219#
PHYSHF              08C7 V  1254  4370  4782  8318#
PLASTBCBA           0877 V  2018  2051  2144  8226#
PNAME               0008 N   192#  887
PNAMSIZ             0008 N   116#  117
PRFCB               0A5D V   860  8486#
PRFCB1              0A65 V   866  8488#
PRFX                0A48 V   878  8482#
PRFX1               0A5A V   843  8485#
PRTERM              1E34 L   653  5586#
PRTERM1             1E3A L  5590# 5596  5599
PRTERM2             1E51 L  5593  5600#
PRVPOS              08F9 V  5393  5436  5450  5491  5505  5659  6496  8352#
PUDA                0010 N   193# 1056
PUSER               0013 N   195# 7794  7798  7805  7809
PWERROR             1898 L  5094# 6437  6854  7616  7701
PWMODE              08DD V  5070  5075  5084  5098  6432  6852  6898  6911  7722  7731
```

```
                    8331#
QDIRFCB1        1C6E L  5244# 5264  6403  6841  6877
QFKEEP          0002 N   302# 8551
QFMX            0001 N   300# 8551
QNAMSIZ         0008 N   117#
QSTAMP          1C84 L  5259# 6544  6918
QSTAMP1         1C89 L  5266# 5313
RANCLOSE        157E L  4098  4104  4124#
RANDOP          205E L  5883  5885#
RANREC          0021 N    98#  999  1032  4057  4061  4067  4073  4749  5832  5930
                5931  6037  6038  6050  6060  6061  6071  6082  6519  7120
                7155  7156
RBLOK           1237 L  3536  3547  3550#
RBLOK0          1241 L  3530  3560#
RCOUNT          0913 V  1445  1462  4218  4286  4392  4653  5746  8388#
RDBUFF          07F1 L  1874# 2275  4442
RDDIR           08F7 L  2481# 2543  6676
RDDIR2          0C54 L  2538  2541#
RDDIRFLAG       0810 V  2381  2536  3124  3708  4904  8164#
RDIR            0C27 L  2474  2511# 4958
READDEBLOCK     0A5E L  2284# 4439
READDIR         0BE7 L  2472# 2791  2837  3252
RECCNT          000F N    94#  947  1435  1441  1460  1463  3651  3683  3687  3784
                3785  4217  4696  7128  7470  7550  7564
RECORDOK        174F L  4394  4417#
RECORDOK1       1758 L  4424  4430#
RECORDOK2       1774 L  4437  4443#
RECSIZ          0080 N    65#    66  6702
RELOG           0821 V  1853  1904  2475  2545  2592  4960  8173#
REMOVEDRIVE     1D87 L  2783  5471# 6171
REMOVEDRIVE1    1D8B L  5468  5476#
REMOVELOCK1     1DA6 L  5495# 5501  5503
REMOVELOCK2     1DB9 L  5498  5504#
REMOVELOCKS     1D95 L  5485# 5597  5616  5646  5927  6630
RENAMO          288E L  6996# 7004  7009
RENAME          284C L  6952#
REPORTERR       04C6 L  1148  1198#
REQUIREDTABLE   08DF V  5965  5967  8337#
RESEL           08DF V   797   930  4885  8163#
RESELECT        19F1 L   925  4865#
RESELECT0       1A12 L  4879  4882#
RESELECT1       1A18 L  4863  4885#
RESELECTX       19E7 L  4858# 6663  6771  7044  7456  7587  7645  7676
RESET37         2986 L  6216  7164#
RESET37X        2989 L  2599  6222  7169#
RESETALL        233D L  6215  6218#
RESETCHECKSUMFCB 06FB L 1625# 6297  6394  6616  6774  6788  6861
RESETRR         03CE L   990# 6010  6014
RESTOREDIRFCB   117D L  3426# 6414  6440  6906
RESTORERC       12CD L  3678# 3857  4037  4121
RESTORERC1      12D9 L  3685  3683#
RET10           0E2A L  2896  2899# 2906  2923  2925
RET11           128A L  3616  3664# 3668  3673
```

```
RET12           12E4 L  3698  3702# 3711
RET13           13FF L  3838  3843  3885# 3888
RET13A          1618 L  4204  4206  4222#
RET13B          1721 L  4362  4374#
RET19           1947 L  4741# 4750
RET19A          19C3 L  4829# 4839
RET20           1AC9 L  4920  4951  4976#
RET20A          1BBF L  5092  5093  5111# 5117  5119  5137
RET21A          1C6D L  5228  5233  5241# 5255  5265  5298  5295  5314  5317
RET30           23ED L  6329# 6331  6352
RET32           24BC L  6415  6442#
RET34           25DB L  6536  6545  6594# 6599
RET35           26B8 L  6695  6707# 6717
RET36           27A9 L  6866# 6907
RET37           292B L  7054  7083# 7097  7113
RET38           2986 L  7187  7189# 7201
RET39           2D31 L  7648  7686# 7696  7727  7730
RET4            199A L  4768  4795#
RET41           0638 L  1461  1464#
RET42           068E L  1538# 1554  1569  1574
RET45           06FA L  1607  1619  1623# 1629
RET48           2845 L  7431# 7450
RET5            0737 L  1702# 1713  1719
RET6            0797 L  1808# 1813  1825  1855
RET7            0CB1 L  2561  2578  2580  2605  2612#
RET71           0CC9 L  2540  2546  2650#
RET7A           087B L  1945  1959# 1966
RET8            077B L  1768  1782#
RET8A           0CFA L  2701# 2718
RET8D           0EE2 L  3017# 3025  3028  3030  3039
RET8E           0F51 L  3073  3076  3078#
RET8F           0FD2 L  3119  3123  3131  3137  3145# 3151  3182
RET9            1137 L  3378# 3409
RET9A           1157 L  3404# 3443
RETD1           0BE6 L  2469# 2476
RETERR0         0268 L   821   824#
RETERR1         029F L   844   848#  851
RETERR2         02A7 L   849   852#
RETERR3         02D0 L   864   877#
RETL1           1E33 L  5555  5567  5584#
RETL2           1E54 L  5602# 5608  5615  5637  5656
RETL25          1DBE L  5490  5506# 5530
RETL3           1F45 L  5710#
RETL4           1F57 L  5725#
RETL6           1FAE L  5763  5769  5777#
RETL7           206B L  5881  5891# 5914
RETL8           22A3 L  6094  6107  6127  6139#
RETMON          01DC L   763   766#
RETMON1         035B L   932   935#
RETMON3         0371 L   940   943#
RETMON4         0374 L   942   945#
RETMON5         037F L   931   950#
RETMON6         01F3 L   768   774#
```

```
SDL0            2C72 L  7591  7594  7596#
SDL1            2C9B L  7601  7610#
SDL2            2CBD L  7625# 7718
SDL3            2CC9 L  6905  7630  7633# 7709
SEARCH          1045 L  3211  3217# 6681  7590  7600
SEARCH1         1048 L  3222# 6977
SEARCH1NAME     1037 L  3202# 5321  7647  7724
SEARCHA         0883 V  3193  3259  3414  6724  6972  8242#
SEARCHABASE     088D V  6642  6715  6719  8260#
SEARCHAOFST     088B V  6644  6716  6721  8259#
SEARCHEXT       1043 L  3213# 3442  3479  5015  6986  7053  7124  7474  7684
SEARCHFIN       10A9 L  3249  3255  3269  3293#
SEARCHH0        0F06 L  3034  3036#
SEARCHH2        0F24 L  3050  3056# 3068
SEARCHH3        0F39 L  3066# 3085  3089  3091  3097  3099  3102  3107  3111
SEARCHH4        0F52 L  3063  3080#
SEARCHH41       0F61 L  3083  3087#
SEARCHH42       0F7C L  3086  3100#
SEARCHH43       0F82 L  3095  3103#
SEARCHH45       0F8D L  3105  3109#
SEARCHH5        0F93 L  3065  3108  3112#
SEARCHH7        0F96 L  3064  3090  3128#
SEARCHH8        0FC5 L  3133  3135#
SEARCHHASH      0EE3 L  3020# 3248
SEARCHI         1023 L  3190# 3221  3430  3880  6343  6685
SEARCHI1        102A L  3195# 3417  6976
SEARCHL         088A V  3026  3197  3273  6684  8255#
SEARCHLISTD     1D0A L  5381  5385  5391#
SEARCHLLIST     1D01 L  5383# 5688
SEARCHLOOP      10B8 L  3283  3286  3302# 3362
SEARCHN         1048 L  3227# 3289  3351  3371  3418  3431  3474  3513  3881  6344
                        6686  7003  7071  7140  7512
SEARCHN1        1058 L  3237  3241#
SEARCHNAME      103F L  1585  3208# 3614  3710  3974  5071  6885
SEARCHNEXT      1079 L  3263  3271#
SEARCHNJMP      1128 L  3344  3349  3368  3370# 3382  3384
SEARCHNLIST     1D06 L  5387# 5497  5551  5592  5615  5636  5701  5819  5980  6107
                        6157  6499  6530
SEARCHNM        1138 L  3326  3380#
SEARCHOK        110F L  3291  3310  3314  3327  3358# 3391
SEARCHOLIST     1CFC L  1597  2786  5379# 5530  5553  5666  6460  6603  6821  7378
                        7405  7517
SEARCHUSER0     0812 V  3357  3383  6316  6326  8166#
SECTOR          0870 V  1095  1256  2330  2340  8234#
SECTPT          0888 V  1251  4779  4783  8309# 8321
SEEK            04F4 L  1246# 2266  4441  4615  4638
SEEKERR         15E1 L  4078  4131  4189#
SEEKOK          15CD L  4150  4170#
SEEKOK1         15D3 L  4102  4123  4180#
SELO            19CB L  4833  4836#
SELDSK          087F V   715  1203  1902  4821  4825  4895  4966  6254  7024  7031
                        8238#
SELECT          19C4 L  4827  4831#
```

```
SELECT0          19C8 L  4813  4823  4834#
SELECT2          19E6 L  4849  4852#
SELECTDISK       1953 L  4754# 4811
SELERROR         0473 L  1137# 4835
SELMSG           0987 V  8427  8441#
SETA1            04C0 L  1192  1194#
SETALLOCBIT      0CCA L  2652# 2745
SETARET          0484 L  1186# 2908  4465  5096  6467  6473  6508  6576  6588
SETCDISK         070A L  1645# 2595  4844  4851  6167
SETCDISK1        070E L  1651# 7032
SETCDR           0798 L  1810# 2865  3911  5032
SETCHKSUMFCB     06EC L   934  1616#
SETDATA          0C1A L  2503# 4440  4637
SETDCNT          0783 L  1789# 3410  3428  3875  6337
SETENDDIR        077C L  1784# 2523  2788  2832  3223  3295  4954  6282
SETFC1           0624 L  1455  1457#
SETFCB           0619 L  1451# 2288  4444  4711
SETFWF           075D L  1745# 3620  3816  3823  3925
SETHASH          0E38 L  2911# 3198
SETHASH1         0E4E L  2929  2931#
SETHASH15        0E5C L  2940  2943#
SETHASH2         0E66 L  2927  2949#
SETLRET          04AE L  1175# 1612  1771  2880  3300  4006  4183  4214  4538  5730
                         5905  5919  5956  5992  7641  7667
SETLRET1         04AC L  1172# 4021  4034  4117  4447  5780
SETPW            1C13 L  5178# 6904  7632
SETPW0           1C19 L  5185# 7745
SETPW1           1C18 L  5187# 5199
SETPW2           1C28 L  5191  5193  5195#
SETPW3           1C35 L  5201  5203#
SETPW4           1C38 L  5206# 5211
SETRC            12A5 L  1439  3645# 3860  3866  4038  4122
SETRC1           12B8 L  3657  3662#
SETRC2           12BB L  3654  3666#
SETRC3           12BF L  3669# 3799  7554  7557
SETROFLAG        0900 V  6123  6150  7023  8358#
SETSIZE          2972 L  7125  7143#
SETUSRCODE       2DCA L  7793  7796# 7804  7807#
SHELL            0380 L   762   953#
SHELL03          03C5 L   974   984#
SHELL04          03C9 L   971   986#
SHELLRR          0926 V  1000  8409#
SHELLSI          0924 V   957   964  8408#
SINGLE           0912 V  1316  1385  2725  3719  4552  4793  7532  8386#
SPSAVE           083A V   707   786  8098# 8542#
SRCHL1           1FE4 L  5820  5823#
SRCHL2A          201C L  5844  5848#
SRCHL3           2020 L  5842  5851#
SRCHL3A          2028 L  5855  5858#
SRCHL4           202D L  5847  5850  5857  5861#
SRCHL6           204A L  5870  5873#
SRCHL7           204C L  5867  5872  5875#
SRCHLIST1        1D0E L  5410# 5437
```

```
SRCHLIST2          1D17 L  5412  5415#
SRCHLIST3          1D43 L  5435# 5439
SRCHLIST4          1D49 L  5420  5438#
SRCHLIST5          1D4B L  5416  5422  5434  5440#
SRCHLLIST          1FDA L  5808# 5829  5836  5838  5889  5950  6023
SS                 SREG V   706   708   785
SSSAVE             0838 V   706   785  8097# 8541#
STAMP1             1C96 L  5273# 6546  7609
STAMP2             1C9A L  5278# 5323  6920  7613
STAMP3             1C9C L  5276  5282#
STAMP4             1CA3 L  5289# 5308
STAMP5             1CB9 L  5300# 7608  7612
STHLRET            2927 L  7016  7081#
SUP                0001 N   365#
SUPERVISOR         0008 V   557# 1057
SUPMOD             0000 V  8050#
SWAP               100A L  3171# 3239  3388  6096  6103  6105  6334
SYOWNER            0002 N   346#
SYSDAT             0006 V   556#  708  8033
SYSFIL             000A N    89# 6385
TESTDIR1           0D7B L  2789# 2795
TESTDIRCS          0CA8 L  2574  2603#
TESTDISKFCB        15E9 L  4197# 4401  4425  4505  5771
TESTDISKFCB1       15FA L  4208# 4698
TESTFFFF           0778 L  1763  1779#
TESTLOCK           204D L  4433  4492  5878#
TESTLOCK1          2064 L  5888# 5897  5901
TESTLOCK2          206C L  5890  5893#
TESTLOCK3          2081 L  5895  5902#
TESTVECTOR         071A L  1573  1668# 1895  2780  4807
TESTVECTOR1        071E L  1675# 6145  6147  6149  7186
TLOG               0802 V  2779  2782  6137  6144  6231  8147#
TMPSELECT          1982 L  4817# 6253  7317  7372  7639
TOD                007E V  5290  7846  7855  7863  7869  8062# 8376#
TRACK              0878 V  1094  1253  2277  2309  2326  2329  2337  9233#
TRUE               FFFF N   112#  547   792   797   932  1385  1588  1606  1871  1887
                           1910  2495  2536  2836  3332  3345  3383  3385  3806  3849
                           3985  3992  4174  4379  4484  4490  4542  4886  5524  5555
                           5606  5644  5757  5758  5770  5896  5923  5954  5968  5998
                           6076  6158  6277  6534  6535  6539  6555  6578  6882  7095
                           7258  7296  7462  8087
TRUNC1             2B94 L  7482# 7513
TRUNC12            2BB4 L  7489  7493  7496#
TRUNC13            2BBB L  7497  7500#
TRUNC15            2BCD L  7503# 7570
TRUNC2             2BD0 L  7499  7511#
TRUNC25            2BE5 L  7518  7521#
TRUNC3             2BE8 L  7505  7527#
TSTINVFCB          076A L  1765# 4378  4474
TSTLOG01           07AB L  1827# 1839  1843
TSTLOGFXS          07A1 L  1820# 3075  4902
TSTOLIST           1DC6 L  3451  5517# 7689
TSTRELOG           07D2 L  1850# 2478
```

```
UA0                0827 V  8183  8196#
UA1                082D V  8186  8183#
UA2                0833 V  8188  8190#
UA3                0839 V  8190  8192#
UA4                083F V  8192  8194#
UA5                0845 V  8194  8196#
UABLOCK            0002 N  2183# 2202  2241
UADISCARD          09F1 L  2212# 2700
UADRV              0004 N  2184# 2201  2221  2223  2239
UAFL1              09F7 L  2218# 2226
UAFL2              0A02 L  2222  2224#
UAHWM              0005 N  2185# 2204  2243  2249
UA1N1              09CF L  2194# 2198
UALINK             0000 N  2182# 2196  2197  2206  2209  2220  2225  2238  2253
UALROOT            0825 V  2193  2208  2217  2235  8183#
UAPRO              0A12 L  2237# 2254
UAPR1              0A33 L  2244  2250#
UAPR2              0A34 L  2240  2242  2252#
UAPREREAD          0A08 L  2229# 2421  4608  4641
UASETUP            09CC L  2188# 4548
UCONCCB            0062 V  8026#
UDAUVLLEN          0019 N   729   777  7965#
UDASAVE            0929 V   723  1076  1100  5518  6241  8413#
UDMADFST           0002 V   727   776  6242  6246  7779  7840  7944# 7965
UDMASEG            0004 V  7833  7838  7946#
UERRORMODE         0010 V  7828  7959#
UFUNC              0006 V   841  7948#
ULEN               0100 N   122#
UMULTCNT           0011 V  7820  7960#
UPDATESTAMP        1CC4 L  4475  5310# 7514
UPDCNT             001A V  5619  5621  7964#
URETSEG            0030 V   719   781  7839  7992#
USER               0000 N   364#  398   447   449   453   472
USERZEROPASS       0923 V  3235  3245  3345  3385  8404#
USRCODE            0880 V  4899  6311  8239#
UWRKSEG            002E V   717  7850  7866  7991#
VRECORD            0915 V  1273  1287  1434  1458  4005  4268  4391  4405  4477  4652
                           5745  8390#
W                  0000 N    54# 1033  1781  2830  3697
WFLAG              090E V  4569  4578  4583  4642  4685  4683  8382#
WRBUFF             07E8 L  1863# 2273  4617  4648
WRDIR              0C0D L  2492# 3506  3593  3819  5088  5298  6421  6913  7070  7467
                           7510  7635  7733
WRDIR0             0C17 L  2490  2499#
WRITE              18AA L  4632  4636#
WRITE0             18C7 L  4645  4647#
WRITE1             18EF L  4668  4681#
WXFCB1             2D45 L  7693  7698#
WXFCB2             2D60 L  7704  7708  7711#
WXFCB3             2D67 L  7714  7716#
WXFCB4             2D6C L  7710  7719#
XCBCX              0001 N   533#  534
XCBDX              0003 N   534#  535
```

```
XCBFUNC          0000 N   532#  533
XCBLEN           0005 N   535#
XDCNT            0815 V  1791  3081  3101  3181  3187  3367  3403  3408  3423  3873
                         3968  4133  5057  6092  6097  6314  6330  6778  6933  7449
                         7481  7519  7599  7683  8169#
XDCNTEQDCNT      1176 L  3420# 6411  6424  6879
XE10             0A10 V  8429  8470#
XE11             0A29 V  8429  8475#
XE3              0995 V  8429  8445#
XE5              0983 V  8429  8452#
XE6              09C7 V  8429  8456#
XE7              09DC V  8429  8460#
XE8              09E8 V  8429  8463#
XE9              09FF V  8429  8467#
XERRLIST         0946 V   838  8427#
XFCBREADONLY     089F V   936  4459  4862  4871  6285  6438  6489  8277#
XIOS             0006 N   370#
XIOSIF           0429 L  1069# 4765  7361  7426
XIOSMOD          0028 V  1077  1101  8053#
XPRINT           0408 L   832   839   879   894   897  1038#
ZERODM1          28FC L  7533  7535#
ZEROLENGTH       0015 N   738  8176#
```