```
1
2  =                    include cpyright.def
3  =                ;*******************************************************
4  =                ;*
5  =                ;*              Concurrent CP/M-86 V2.1
6  =                ;*              =======================
7  =                ;*
8  =                ;*              Copyright (c) 1982
9  =                ;*
10 =                ;*                 Digital Research
11 =                ;*                   P.O.Box 579
12 =                ;*           Pacific Grove, California 93950
13 =                ;*
14 =                ;*                   (408) 649-3896
15 =                ;*                   TWX 9103605001
16 =                ;*
17 =                ;* All Information contained in this source listing is
18 =                ;*
19 =                ;*                     PROPRIETORY
20 =                ;*                     ===========
21 =                ;*
22 =                ;* All rights reserved.  No part of this document
23 =                ;* may be reproduced, transmitted, stored in a
24 =                ;* retrieval system, or translated into any language
25 =                ;* or computer language, in any form or by any means
26 =                ;* without the prior written permission of Digital
27 =                ;* Research, P.O Box 579, Pacific Grove, California.
28 =                ;*
29 =                ;*******************************************************
30
31
32                  ;*******************************************************
33                  ;*
34                  ;*      MP/M or CCP/M Supervisor Module
35                  ;*
36                  ;*******************************************************
```

```
                        eject ! include system.def
                    ;*****************************************************
                    ;*
                    ;*      SYSTEM DEFINITIONS
                    ;*
                    ;*****************************************************
                    
FFFF                true        equ  0ffffh     ; value of TRUE
0000                false       equ  0          ; value of FALSE
0000                unknown     equ  0          ; value to be filled in
0080                dskrecl     equ  128        ; log. disk record len
0020                fcblen      equ  32         ; size of file control block
0008                pnamsiz     equ  8          ; size of process name
0008                qnamsiz     equ  pnamsiz    ; size of queue name
0008                fnamsiz     equ  pnamsiz    ; size of file name
0003                ftypsiz     equ  3          ; size of file type
00E0                osint       equ  224        ; int vec for O.S. entry
00E1                debugint    equ  osint+1    ; int vec for debuggers
0100                ulen        equ  0100h      ; size of uda
015E                u8087len    equ  ulen + 94  ; size of uda when process uses 8087
0030                pdlen       equ  030h       ; size of Process Descriptor
0005                todlen      equ  5          ; size of Time of Day struct
0001                flag_tick   equ  1          ; flag 0 = tick flag
0002                flag_sec    equ  2          ; flag 1 = second flag
0003                flag_min    equ  3          ; flag 2 = minute flag
00AA                ldtabsiz    equ  0aah       ; ldtablen=11, 10 entries
                    
0000                mpm     equ     false       ; MP/M-86 or
FFFF                ccpm    equ     not mpm     ; CCP/M-86
                    
0000                BCPM    equ     false       ; CP/M-86 BDOS
FFFF                BMPM    equ     not BCPM    ; multi-process BDOS
```

```
                        eject ! include modfunc.def
                    ;*****************************************************
                    ;*
                    ;*      CCP/M-86, 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
0001                supmod_bit      equ     001h
0002                rtmmod_bit      equ     002h
0004                memmod_bit      equ     004h
0008                bdosmod_bit     equ     008h
0010                ciomod_bit      equ     010h
0020                xiosmod_bit     equ     020h
0040                netmod_bit      equ     040h

                        ; Supervisor Functions
                    ;f_sysreset     equ     (user * 0100h) + 0
                    ;f_conin        equ     (user * 0100h) + 1
0002                f_conout        equ     (user * 0100h) + 2
                    ;f_rconin       equ     (user * 0100h) + 3      ;supported internally
                    ;f_rconout      equ     (user * 0100h) + 4      ;under CCP/M
0005                f_lstout        equ     (user * 0100h) + 5
                    ;f_rawconio     equ     (user * 0100h) + 6
                    ;f_getiobyte    equ     (user * 0100h) + 7
                    ;f_setiobyte    equ     (user * 0100h) + 8
0009                f_conwrite      equ     (user * 0100h) + 9
000A                f_conread       equ     (user * 0100h) + 10
                    ;f_constat      equ     (user * 0100h) + 11
                    ;f_bdosversion  equ     (user * 0100h) + 12
                    ;f_diskreset    equ     (user * 0100h) + 13
                    ;f_diskselect   equ     (user * 0100h) + 14
000F                f_fopen         equ     (user * 0100h) + 15
0010                f_fclose        equ     (user * 0100h) + 16
                    ;f_searchfirst  equ     (user * 0100h) + 17
                    ;f_searchnext   equ     (user * 0100h) + 18
```

```
124
125  =                    ;f_fdelete       equ     (user * 0100h) + 19
126  =  0014              f_freadseq       equ     (user * 0100h) + 20
127  =                    ;f_fwriteseq     equ     (user * 0100h) + 21
128  =                    ;f_fmake         equ     (user * 0100h) + 22
129  =                    ;f_frename       equ     (user * 0100h) + 23
130  =                    ;f_loginvector   equ     (user * 0100h) + 24
131  =  0019              f_getdefdisk     equ     (user * 0100h) + 25
132  =  001A              f_setdma         equ     (user * 0100h) + 26
133  =                    ;f_getallocvec   equ     (user * 0100h) + 27
134  =                    ;f_writeprotect  equ     (user * 0100h) + 28
135  =                    ;f_getrovector   equ     (user * 0100h) + 29
136  =                    ;f_setfileattr   equ     (user * 0100h) + 30
137  =                    ;f_getdpb        equ     (user * 0100h) + 31
138  =  0020              f_usercode       equ     (user * 0100h) + 32
139  =  0021              f_freadrdm       equ     (user * 0100h) + 33
140  =                    ;f_fwriterdm     equ     (user * 0100h) + 34
141  =                    ;f_filesize      equ     (user * 0100h) + 35
142  =  0024              f_setrndrec      equ     (user * 0100h) + 36
143  =                    ;f_resetdrive    equ     (user * 0100h) + 37
144  =                    ;f_accessdrive   equ     (user * 0100h) + 38
145  =                    ;f_freedrive     equ     (user * 0100h) + 39
146  =                    ;f_writerndzero  equ     (user * 0100h) + 40
147  =                    ;f_chain         equ     (user * 0100h) + 47
148  =  0032              f_callbios       equ     (user * 0100h) + 50
149  =  0033              f_setdmab        equ     (user * 0100h) + 51
150  =                    ;f_getdma        equ     (user * 0100h) + 52
151  =                    ;f_getmaxmem     equ     (user * 0100h) + 53
152  =                    ;f_getabsmaxmem  equ     (user * 0100h) + 54
153  =                    ;f_allocmem      equ     (user * 0100h) + 55
154  =                    ;f_allocabsmem   equ     (user * 0100h) + 56
155  =  0039              f_freemem        equ     (user * 0100h) + 57
156  =  003A              f_freeall        equ     (user * 0100h) + 58
157  =                    ;f_userload      equ     (user * 0100h) + 59
158  =  006E              f_delim          equ     (user * 0100h) + 110
159  =  0080              f_malloc         equ     (user * 0100h) + 128
160  =  0082              f_memfree        equ     (user * 0100h) + 130
161  =                    ;f_polldev       equ     (user * 0100h) + 131
162  =  0084              f_flagwait       equ     (user * 0100h) + 132
163  =  0085              f_flagset        equ     (user * 0100h) + 133
164  =  0086              f_qmake          equ     (user * 0100h) + 134
165  =  0087              f_qopen          equ     (user * 0100h) + 135
166  =                    ;f_qdelete       equ     (user * 0100h) + 136
167  =  0089              f_qread          equ     (user * 0100h) + 137
168  =  008A              f_cqread         equ     (user * 0100h) + 138
169  =  008B              f_qwrite         equ     (user * 0100h) + 139
170  =  008C              f_cqwrite        equ     (user * 0100h) + 140
171  =                    ;f_delay         equ     (user * 0100h) + 141
172  =  008E              f_dispatch       equ     (user * 0100h) + 142
173  =  008F              f_terminate      equ     (user * 0100h) + 143
174  =  0090              f_createproc     equ     (user * 0100h) + 144
175  =  0091              f_setprior       equ     (user * 0100h) + 145
176  =  0092              f_conattach      equ     (user * 0100h) + 146
```

```
177
178  =  0093              f_condetach      equ     (user * 0100h) + 147
179  =                    ;f_setdefcon     equ     (user * 0100h) + 148
180  =  0095              f_conassign      equ     (user * 0100h) + 149
181  =                    ;f_clicmd        equ     (user * 0100h) + 150
182  =                    ;f_callrsp       equ     (user * 0100h) + 151
183  =  0098              f_parsefilename  equ     (user * 0100h) + 152
184  =                    ;f_getdefcon     equ     (user * 0100h) + 153
185  =                    ;f_sdataddr      equ     (user * 0100h) + 154
186  =                    ;f_timeofday     equ     (user * 0100h) + 155
187  =                    ;f_pdaddress     equ     (user * 0100h) + 156
188  =                    ;f_abortprocess  equ     (user * 0100h) + 157
189  =  009E              f_lstattach      equ     (user * 0100h) + 158
190  =  009F              f_lstdetach      equ     (user * 0100h) + 159
191  =                    ;f_setdeflst     equ     (user * 0100h) + 160
192  =                    ;f_clstattch     equ     (user * 0100h) + 161
193  =  00A2              f_cconattch      equ     (user * 0100h) + 162
194  =                    ;f_osvernum      equ     (user * 0100h) + 163
195  =                    ;f_getdeflst     equ     (user * 0100h) + 164
196  =
197  =
198  =                            ; Internal SUP functions
199  =  010A              f_load           equ     (sup * 0100h) + 10
200  =  010E              f_cload          equ     (sup * 0100h) + 14
201  =
202  =
203  =                            ; Internal RTM functions
204  =  0203              f_inflagset      equ     (rtm * 0100h) + 3
205  =  0212              f_sleep          equ     (rtm * 0100h) + 18
206  =  0213              f_wakeup         equ     (rtm * 0100h) + 19
207  =  0214              f_findpdname     equ     (rtm * 0100h) + 20
208  =  0215              f_sync           equ     (rtm * 0100h) + 21
209  =  0216              f_unsync         equ     (rtm * 0100h) + 22
210  =  0217              f_no_abort       equ     (rtm * 0100h) + 23
211  =  0218              f_ok_abort       equ     (rtm * 0100h) + 24
212  =  0219              f_no_abort_spec  equ     (rtm * 0100h) + 25
213  =
214  =
215  =                            ; Internal MEM functions
216  =  0308              f_share          equ     (mem * 0100h) + 8
217  =  0309              f_maualloc       equ     (mem * 0100h) + 9
218  =  030A              f_maufree        equ     (mem * 0100h) + 10
219  =  030B              f_mlalloc        equ     (mem * 0100h) + 11
220  =  030C              f_mlfree         equ     (mem * 0100h) + 12
221  =
222  =
223  =                            ; Internal CIO functions
224  =  040E              f_conprint       equ     (cio * 0100h) + 14
225  =  0417              f_cioterm        equ     (cio * 0100h) + 23
226  =  0418              f_ciostat        equ     (cio * 0100h) + 24
227  =  0402              f_rconin         equ     (cio * 0100h) + 2
228  =  0403              f_rconout        equ     (cio * 0100h) + 3
229  =
```

```
230
231  =
232  =                              ; Internal BDOS functions
233  =  0520                f_bdosterm      equ     (bdos * 0100h) + 45
234
```

```
                        eject 1 include modtab.def
                ;*******************************************************
                ;*
                ;*      Definition of Module Table Entry
                ;*
                ;*******************************************************
                =
0000            mod_entry       equ     0
0004            mod_init        equ     mod_entry + dword
0008            modlen          equ     mod_init + dword
```

```
                        eject 1 include xioscb.def
                    ;*******************************************************
                    ;*
                    ;*      XIOS function jump table offsets
                    ;*
                    ;*******************************************************
                    if ccpm
0000                io_const        equ     0               ;func 50, direct BIOS
0001                io_conin        equ     1               ;can access io_funcs 0-6
0002                io_conout       equ     2
0003                io_liststt      equ     3
0004                io_list         equ     4
0005                io_auxin        equ     5
0006                io_auxout       equ     6
0007                io_switch       equ     7
0008                io_statline     equ     8
0009                io_seldsk       equ     9
000A                io_read         equ     10
000B                io_write        equ     11
000C                io_flush        equ     12
000D                io_polldev      equ     13
000C                nxiosfuncs      equ     io_flush
                    endif
                    if mpm
                    io_const        equ     0
                    io_conin        equ     1
                    io_conout       equ     2
                    io_list         equ     3
                    ;io_punch       equ     4               ;not used
                    ;io_reader      equ     5               ;not used
                    io_home         equ     6
                    io_seldsk       equ     7
                    io_settrk       equ     8
                    io_setsec       equ     9
                    io_setdma       equ     10
                    io_read         equ     11
                    io_write        equ     12
                    ;io_liststt     equ     13              ;not used
                    io_sectran      equ     14
                    io_setdmab      equ     15
                    ;io_getsegt     equ     16              ;not used
                    io_polldev      equ     17
                    io_strtclk      equ     18
                    io_stopclk      equ     19
                    io_maxconsole   equ     20
                    io_maxlist      equ     21
                    io_selmemory    equ     22
                    io_idle         equ     23
```

```
300
301   =                       io_flush        equ     24
302   =                       nxiosfuncs      equ     io_flush
303   =                       endif
304   =
305   =                       ;*******************************************************
306   =                       ;*
307   =                       ;*      XIOS Parameter Block for CALL XIOS functions
308   =                       ;*
309   =                       ;*******************************************************
310   =
311   =  0000                 xcb_func        equ     0
312   =  0001                 xcb_cx          equ     word ptr xcb_func + byte
313   =  0003                 xcb_dx          equ     word ptr xcb_cx + word
314   =  0005                 xcblen          equ     xcb_dx + word
315
```

```
                        eject | include pd.def
                ;*******************************************************
                ;*
                ;*      Process Descriptor - with the UDA associated
                ;*              with the PD, describes the current
                ;*              state of a Process under MP/M-CCP/M
                ;*
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 00|   link    |  thread   |stat |prior|   flag    |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 08|                     Name                      |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 10|    uda    | dsk | user| ldsk|luser|    mem    |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 18| cmod|tkcnt|   wait    | reserved  |  parent   |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 20| cns |abort|  conmode  | lst |     reserved    |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 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
                ;*      cmod    - compatibility mode bits.
                ;*                80 -> F1 bit
                ;*                40 -> F2 bit
                ;*                20 -> F3 bit
                ;*                10 -> F4 bit
                ;*      tkcnt   - temp keep count, # times tempkeep has been
                ;*                turned on
                ;*      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     - default console #
                ;*      abort   - abort code
                ;*      lst     - default list #
                ;*      conmode - console mode for function 109
                ;*      reserved- not currently used
                ;*      pret    - return code at termination
                ;*      scratch - scratch word
```

```
                        ;*
                        ;*****************************************************
                        ;
0000                    p_link          equ     word ptr 0
0002                    p_thread        equ     word ptr p_link + word
0004                    p_stat          equ     byte ptr p_thread + word
0005                    p_prior         equ     byte ptr p_stat + byte
0006                    p_flag          equ     word ptr p_prior + byte
0008                    p_name          equ     byte ptr p_flag + word
0010                    p_uda           equ     word ptr p_name + pnamsiz
0012                    p_dsk           equ     byte ptr p_uda + word
0013                    p_user          equ     byte ptr p_dsk + byte
0014                    p_ldsk          equ     byte ptr p_user + byte
0015                    p_luser         equ     byte ptr p_ldsk + byte
0016                    p_mem           equ     word ptr p_luser + byte
0018                    p_cmod          equ     byte ptr p_mem + word
0019                    p_tkcnt         equ     byte ptr p_cmod + byte
001A                    p_wait          equ     word ptr p_tkcnt + byte
001E                    p_parent        equ     word ptr p_wait + 4
0020                    p_cns           equ     byte ptr p_parent + word
0021                    p_abort         equ     byte ptr p_cns + byte
0022                    p_conmode       equ     word ptr p_abort + byte
0024                    p_lst           equ     byte ptr p_conmode + word
002C                    p_pret          equ     word ptr p_lst + 8
002E                    p_scratch       equ     byte ptr p_pret + word
002E                    p_wscrtch       equ     word ptr p_scratch
                        ;
                        ;       Process descriptor pd_status values
                        ;
0000                    ps_run          equ     0       ; in ready list root
0001                    ps_poll         equ     1       ; in poll list
0002                    ps_delay        equ     2       ; in delay list
0003                    ps_swap         equ     3       ; in swap list
0004                    ps_term         equ     4       ; terminating
0005                    ps_sleep        equ     5       ; sleep processing
0006                    ps_dq           equ     6       ; in dq list
0007                    ps_nq           equ     7       ; in nq list
0008                    ps_flagwait     equ     8       ; in flag table
0009                    ps_clowait      equ     9       ; waiting for character
000A                    ps_sync         equ     10      ; waiting for sync structure
                        ;
                        ;       Process descriptor pd_flag bit values
                        ;
0001                    pf_sys          equ     00001h  ; system process
0002                    pf_keep         equ     00002h  ; do not terminate
0004                    pf_kernal       equ     00004h  ; resident in kernal
0008                    pf_pure         equ     00008h  ; pure memory descibed
0010                    pf_table        equ     00010h  ; from pd table
0020                    pf_resource     equ     00020h  ; waiting for resource
0040                    pf_raw          equ     00040h  ; raw console i/o
```

```
422
423  =  0080              pf_ctlc         equ     00080h  ; abort pending
424  =  0100              pf_active       equ     00100h  ; active tty
425  =  0200              pf_tempkeep     equ     00200h  ; don't terminate yet...
426  =  0400              pf_ctld         equ     00400h  ; explicit detach occured
427  =  0800              pf_childabort   equ     00800h  ; child terminated abnormally
428  =  1000              pf_noctls       equ     01000h  ; control S not allowed
429  =  2000              pf_dskld        equ     02000h  ; process was loaded from disk
430  =  4000              pf_nokbd        equ     04000h  ; ignore keyboard
431  =  8000              pf_8087         equ     08000h  ; process uses 8087
432  =                    ;
433  =                    ;       Process descriptor pcm_flag bit values
434  =                    ;               (console mode set by function 109)
435  =                    ;
436  =  0001              pcm_11          equ     00001h  ; control C status for function 11
437  =  0002              pcm_ctls        equ     00002h  ; disable control S-Q
438  =  0004              pcm_rout        equ     00004h  ; raw output, no tabs or control P
439  =  0008              pcm_ctlc        equ     00008h  ; disable control C
440  =                    ;pcm_rsrv       equ     00040h  ; used by CP/M 3.0
441  =  0080              pcm_ctlo        equ     00080h  ; disable control O
442  =  0300              pcm_rsx         equ     00300h  ; 2 bits used by RSX for status
443
```

```
                        eject 1 include err.def
                ;*******************************************************
                ;*
                ;*      Error definitions
                ;*
                ;*******************************************************

0001            e_not_implemented   equ  1   ; not implemented
0002            e_bad_entry         equ  2   ; illegal func. #
0003            e_no_memory         equ  3   ; cant find memory
0004            e_ill_flag          equ  4   ; illegal flag #
0005            e_flag_ovrrun       equ  5   ; flag over run in
0006            e_flag_underrun     equ  6   ; flag underrun in
0007            e_no_qd             equ  7   ; no unused qd's
0008            e_no_qbuf           equ  8   ; no free qbuffer
0009            e_no_queue          equ  9   ; cant find que on qlr
000A            e_q_inuse           equ  10  ; queue in use
                ;                   equ  11  ;
000C            e_no_pd             equ  12  ; no free pd's
000D            e_q_protected       equ  13  ; no que access
000E            e_q_empty           equ  14  ; empty queue
000F            e_q_full            equ  15  ; full queue
0010            e_ncliq             equ  16  ; Cli queue missing
                ;                   equ  17  ;
0012            e_no_umd            equ  18  ; no unused MD's
0013            e_ill_cns           equ  19  ; illegal cns num.
0014            e_no_pdname         equ  20  ; no PD match
0015            e_no_cnsmatch       equ  21  ; no cns match
0016            e_nclip             equ  22  ; no cli process
0017            e_illdisk           equ  23  ; illegal disk #
0018            e_badfname          equ  24  ; illegal filename
0019            e_badftype          equ  25  ; illegal filetype
001A            e_nochar            equ  26  ; char not ready
001B            e_ill_md            equ  27  ; illegal mem descriptor
001C            e_bad_load          equ  28  ; bad ret. from BDOS load
001D            e_bad_read          equ  29  ; bad ret. from BDOS read
001E            e_bad_open          equ  30  ; bad ret. from BDOS open
001F            e_nullcmd           equ  31  ; null command
0020            e_not_owner         equ  32  ; not owner of resource
0021            e_no_cseg           equ  33  ; no CSEG in load file
0022            e_active_pd         equ  34  ; PD exists on Thread Root
0023            e_pd_noterm         equ  35  ; could not terminate process
0024            e_no_attach         equ  36  ; could not attach to ciodev
0025            e_ill_lst           equ  37  ; illegal list device
0026            e_ill_passwd        equ  38  ; illegal password specified
0027            e_no_mimic          equ  39  ; can't mimic or unmimic
0028            e_abort             equ  40  ; external termination occured
0029            e_fixuprec          equ  41  ; fixup error
```

```
                        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)
                ;*
                ;*******************************************************
0000            q_link          equ     word ptr 0
0002            q_net           equ     byte ptr q_link + word
0003            q_org           equ     word ptr q_net + byte
0004            q_flags         equ     byte ptr q_org + byte
0006            q_name          equ     word ptr q_flags + word
000E            q_msglen        equ     word ptr q_name + qnamsiz
0010            q_nmsgs         equ     word ptr q_msglen + word
0012            q_dq            equ     word ptr q_nmsgs + word
0014            q_nq            equ     word ptr q_dq + word
0016            q_msgcnt        equ     word ptr q_nq + word
0018            q_msgout        equ     word ptr q_msgcnt + word
001A            q_buf           equ     word ptr q_msgout + word
001C            qdlen           equ     q_buf + word
                ;
                ;       Q_FLAGS values
                ;
0001            qf_mx           equ     001h    ; Mutual Exclusion
0002            qf_keep         equ     002h    ; NO DELETE
0004            qf_hide         equ     004h    ; Not User writable
```

```
                        qf_rsp          equ     008h    ; rsp queue
                        qf_table        equ     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 fcb.def
                ;*******************************************************
                ;*
                ;*      File Control Block Format
                ;*
                ;*******************************************************
                
0000            fcb_dr          equ     byte ptr 00h
0001            fcb_name        equu    byte ptr 01h
0009            fcb_type        equ     byte ptr 09h

0010            fcb_pwd         equ     byte ptr 010h
0018            fcb_pptr        equ     word ptr 018h
001A            fcb_plen        equ     byte ptr 01ah

0020            fcb_cr          equ     byte ptr 020h
0021            fcb_r0          equ     word ptr 021h

```

```
599
600   =                         eject 1 include acb.def
601   =                       ;*************************************************************
602   =                       ;*
603   =                       ;*      Assign Console Control Block
604   =                       ;*
605   =                       ;*************************************************************
606   =
607   =  0000                 acb_cns         equ     byte ptr 0
608   =  0001                 acb_match       equ     byte ptr acb_cns + byte
609   =  0002                 acb_pd          equ     word ptr acb_match + byte
610   =  0004                 acb_name        equ     byte ptr acb_pd + word
611   =  000C                 acblen          equ     acb_name + pnamsiz
612
```

```
                        eject 1 include enttab.def
                ;*******************************************************
                ;*
                ;*      System Entry Table entry format
                ;*
                ;*******************************************************

                enttab_entry    equ     word ptr 0
                entlen          equ     enttab_entry + word

                        ; Flags Values in ent_flag

                ef_network      equ     0f0h    ; network intercept

                ; see SYSTEM.DEF for ent_tab definition
```

```
                        eject 1 include mem.def
                ;*******************************************************
                ;*
                ;*      Memory Descriptor Formats
                ;*
                ;*******************************************************

                        ; format while on MFL or MAL

0000            m_link          equ     word ptr 0
0002            m_start         equ     word ptr m_link+word
0004            m_length        equ     word ptr m_start+word
0006            m_plist         equ     word ptr m_length+word
0008            m_unused        equ     word ptr m_plist+word
000A            mdlen           equ     m_unused+word

                        ; format while on PD as memory segment
                        ;   ms_flags uses same definitions as
                        ;   mpb_flags

0000            ms_link         equ     word ptr m_link
0002            ms_start        equ     word ptr m_start
0004            ms_length       equ     word ptr m_length
0006            ms_flags        equ     word ptr m_plist
0008            ms_mau          equ     word ptr m_unused

                        ; format of spb (share parameter block)

0000            spb_opd         equ     word ptr 0
0002            spb_rpd         equ     word ptr spb_opd + word
0004            spb_start       equ     word ptr spb_rpd + word
```

```
663
664  =                          eject ! include mpb.def
665  =                        ;*******************************************************
666  =                        ;*
667  =                        ;*      Memory Parameter Block
668  =                        ;*
669  =                        ;*******************************************************
670  =
671  =  0000                  mpb_start       equ     word ptr 0
672  =  0002                  mpb_min         equ     word ptr mpb_start + word
673  =  0004                  mpb_max         equ     word ptr mpb_min + word
674  =  0006                  mpb_pdadr       equ     word ptr mpb_max + word
675  =  0008                  mpb_flags       equ     word ptr mpb_pdadr + word
676  =
677  =  000A                  mpblen          equ     mpb_flags + word
678  =
679  =
680  =                                ; mpb_flags definition (also ms_flags)
681  =
682  =  0001                  mf_load         equ     00001h
683  =  0002                  mf_share        equ     00002h
684  =  0004                  mf_code         equ     00004h
685  =                        ; mf_data       equ     00008h
686  =                        ; mf_extra      equ     00010h
687  =                        ; mf_stack      equ     00020h
688  =  0040                  mf_udaonly      equ     00040h
689  =
690  =                        ;*******************************************************
691  =                        ;*
692  =                        ;*      Memory Free Parameter Block
693  =                        ;*
694  =                        ;*******************************************************
695  =
696  =  0000                  mfpb_start      equ     word ptr 0
697  =  0002                  mfpb_pd         equ     word ptr mfpb_start + word
698  =  0004                  mfpblen         equ     mfpb_pd + word
699  =
700  =                        ;*******************************************************
701  =                        ;*
702  =                        ;*      MAU Free Request
703  =                        ;*
704  =                        ;*      +----+----+----+----+----+----+
705  =                        ;*      |    mau    |    sat    |  start  |
706  =                        ;*      +----+----+----+----+----+----+
707  =                        ;*
708  =                        ;*******************************************************
709  =
710  =  0000                  maf_mau         equ     word ptr 0
711  =  0002                  maf_sat         equ     word ptr maf_mau + word
712  =  0004                  maf_start       equ     word ptr maf_sat + word
713  =  0006                  maflen          equ     maf_start + word
714
```

```
715
716  =                          eject 1 include clicb.def
717  =                        ;*********************************************************
718  =                        ;*
719  =                        ;*      Cli Command Control Block
720  =                        ;*
721  =                        ;*********************************************************
722  =
723  =  0000                  clicb_net       equ     byte ptr 0
724  =  0001                  clicb_cmd       equ     byte ptr clicb_net + byte
725  =
726  =  0082                  clicblen        equ     clicb_cmd + 129*byte
727
```

```
                        eject ! include rsp.def
                    ;*******************************************************
                    ;*
                    ;*      RSP.DEF - Describes the relative offsets of
                    ;*                data items in the Data Segment of a
                    ;*                Resident System Process.  GENSYS
                    ;*                links all RSP's together through the
                    ;*                rsp_link field (segment address ptrs).
                    ;*                Each RSP is started up as System
                    ;*                Initialization.
                    ;*
                    ;* Format:        +---+---+---+---+---+---+---+---+
                    ;*          00    | link  |sdatvar|ncp| reserved  |
                    ;*                +---+---+---+---+---+---+---+---+
                    ;*          08    |    reserved   |  CS   |reserve|
                    ;*                +---+---+---+---+---+---+---+---+
                    ;*          10    |  Process Descriptor           |
                    ;*                +---+---+---+---+---+---+---+---+
                    ;*          40    |   User Data Area              |
                    ;*                +---+---+---+---+---+---+---+---+
                    ;*          E0    | RSP data area                 |
                    ;*                +---+---+---+---+---+---+---+---+
                    ;*
                    ;*      link - GENSYS links all RSP's through this.
                    ;*                At system init, this value is filled
                    ;*                with the SYSDAT segment address
                    ;*      sdatvar - if non-zero, this is a variable address
                    ;*                in the SYSDAT area which indicates the
                    ;*                a value to put into ncopies (ncp).
                    ;*      ncp - Number of copies - Used by GENSYS to determine
                    ;*                how many copies of this RSP to generate.
                    ;*                This number is modified to be the specific
                    ;*                copy that was generated.
                    ;*      CS - Used by GENSYS to determine where a shared
                    ;*                code segment may exist in multiple copy
                    ;*                RSPs.
                    ;*
                    ;*******************************************************

0000                rsp_top       equ  0
0000                rsp_link      equ  word ptr rsp_top
0000                rsp_sysdat    equ  rsp_link
0002                rsp_sdatvar   equ  word ptr rsp_link + word
0004                rsp_ncopies   equ  byte ptr rsp_sdatvar + word
0005                rsp_reserved  equ  rsp_ncopies + byte
0008                rsp_md        equ  08h
0010                rsp_pd        equ  10h
0040                rsp_uda       equ  ((rsp_pd+pdlen+0fh)/10h)*10h
0140                rsp_bottom    equ  rsp_uda + ulen
```

```
779
780  =                          eject 1 include pcb.def
781  =                        ;*******************************************************
782  =                        ;*
783  =                        ;*      Parse Control Block
784  =                        ;*
785  =                        ;*******************************************************
786  =
787  =  0000                 pcb_flnmptr equ word ptr 0                    ; name ptr
788  =  0002                 pcb_fcbptr  equ word ptr pcb_flnmptr + word ; fcb ptr
789  =  0004                 pcblen      equ pcb_fcbptr + word
790
```

```
                        eject | include cmdh.def
                    ;*****************************************************
                    ;*
                    ;*      Command Header Format and Load Fixup Records
                    ;*
                    ;*****************************************************
                                                                ;group descriptor format
0000                ch_form         equ     byte ptr 0
0001                ch_length       equ     word ptr (ch_form + byte)
0003                ch_base         equ     word ptr (ch_length + word)
0005                ch_min          equ     word ptr (ch_base + word)
0007                ch_max          equ     word ptr (ch_min + word)
0009                chlen           equ     ch_max + word

0008                ch_entmax       equ     8                   ;max number of group
                                                                ;descriptors

007F                ch_lbyte        equ     byte ptr 07fh       ;MSB bit in CH_LBYTE
007D                ch_fixrec       equ     word ptr 07dh       ;signals fixup records
                                                                ;start at record number
                                                                ;in CH_FIXREC

0000                fix_grp         equ     byte ptr 0          ;format of fixup record
0001                fix_para        equ     word ptr fix_grp + byte
0003                fix_offs        equ     byte ptr fix_para + word
0004                fixlen          equ     fix_offs + byte
```

```
                        eject 1 include char.def
                ;*******************************************************
                ;*
                ;*      Character Definitions
                ;*
                ;*******************************************************

0003            ctlc    equ     003h
0004            ctld    equ     004h
0005            ctle    equ     005h
0007            bell    equ     007h
0008            ctlh    equ     008h
0009            tab     equ     009h
000A            lf      equ     00ah
000D            cr      equ     00dh
0010            ctlp    equ     010h
0011            ctlq    equ     011h
0012            ctlr    equ     012h
0013            ctls    equ     013h
0015            ctlu    equ     015h
0018            ctlx    equ     018h
001A            ctlz    equ     01ah
005E            ctl     equ     05eh
007F            rubout  equ     07fh

                if ccpm
```

```
                        eject | include vccb.def
                ;***********************************************
                ;*
                ;*      Virtual Console Control Block Definition
                ;*
                ;*      +---------+---------+---------+---------+
                ;*  00  |      attach       |       queue       |
                ;*      +---------+---------+---------+---------+
                ;*  04  |  flag   | startcol| column  |  nchar  |
                ;*      +---------+---------+---------+---------+
                ;*  08  |  mimic  | msource |   pc    |   vc    |
                ;*      +---------+---------+---------+---------+
                ;*  0C  |  btmp   |   rsvd  |       state       |
                ;*      +---------+---------+---------+---------+
                ;*  10  |     maxbufsiz     |       vinq        |
                ;*      +---------+---------+---------+---------+
                ;*  14  |       voutq       |       vcmxq       |
                ;*      +---------+---------+---------+---------+
                ;*  18  | qpbflags| qpbfill |     qpbqaddr      |
                ;*      +---------+---------+---------+---------+
                ;*  1C  |      qpbnmsgs     |     qpbbuffptr    |
                ;*      +---------+---------+---------+---------+
                ;*  20  |       qbuff       |       cosleep     |
                ;*      +---------+---------+---------+---------+
                ;*  24  |       usleep      |       vsleep      |
                ;*      +---------+---------+---------+---------+
                ;*  28  |               Reserved                |
                ;*      +---------+---------+---------+---------+
                ;*
                ;*
                ;*
                ;*      attach  - current owner of device
                ;*                if 0, no owner
                ;*                if 0ffffh, a mimic device
                ;*      queue   - linked list of PDs waiting to attach
                ;*      flag    - run-time flags
                ;*      startcol- used for line editing
                ;*      column  - used for line editing
                ;*      nchar   - 1 character read ahead for CTRL chars.
                ;*      mimic   - cio dev that mimics us.
                ;*                0ffh means no mimic device
                ;*      msource - if attach = 0ffffh, we are a
                ;*                mimic device and msource is the
                ;*                device we are mimicing.
                ;*      pc      - physical console number
                ;*      vc      - virtual console number
                ;*      btmp    - temporary line editing variable
                ;*      rsvd    - unused
                ;*      state   - current state of virtual console
                ;*      maxbufsiz - maximum file size for buffered mode
                ;*      vinq    - address of QPB for this virtual console's
                ;*                input.  written by PIN, created by the VOUT
```

```
                        ;*                process associated with this virtual console
                        ;*      voutq   - address of QPP for this virtual console's
                        ;*                output. written and created by the assoc. VOUT
                        ;*      vcmxq   - MX queue for changing of virtual console state
                        ;*      qpbflags- the 1st 8 bytes of a Queue Parameter Block, for
                        ;*                queue reads and writes, used only by reentrant
                        ;*                intercept code
                        ;*      qbuff   - buffer for queue writes
                        ;*      cosleep - temporary list for processes waiting for XIOS console output
                        ;*      usleep  - user process sleeps here
                        ;*      vsleep  - vout process sleeps here
                        ;*
0000                    c_attach        equ     word ptr 0
0002                    c_queue         equ     word ptr c_attach + word
0004                    c_flag          equ     byte ptr c_queue + word
0005                    c_strtcol       equ     byte ptr c_flag + byte
0006                    c_column        equ     byte ptr c_strtcol + byte
0007                    c_nchar         equ     byte ptr c_column + byte
0008                    c_mimic         equ     byte ptr c_nchar + byte
0009                    c_msource       equ     byte ptr c_mimic + byte
000A                    c_pc            equ     byte ptr c_msource + byte
000B                    c_vc            equ     byte ptr c_pc + byte
000C                    c_btmp          equ     byte ptr c_vc + byte
000D                    c_rsvd          equ     byte ptr c_btmp + byte
000E                    c_state         equ     word ptr c_rsvd + byte
0010                    c_maxbufsiz     equ     word ptr c_state + word
0012                    c_vinq          equ     word ptr c_maxbufsiz + word
0014                    c_voutq         equ     word ptr c_vinq + word
0016                    c_vcmxq         equ     word ptr c_voutq + word
0018                    c_qpbflags      equ     byte ptr c_vcmxq + word
0019                    c_qpbfill       equ     byte ptr c_qpbflags + byte
001A                    c_qpbqaddr      equ     word ptr c_qpbfill + byte
001C                    c_qpbnmsgs      equ     word ptr c_qpbqaddr + word
001E                    c_qpbbuffptr    equ     word ptr c_qpbnmsgs + word
0020                    c_qbuff         equ     word ptr c_qpbbuffptr + word
0022                    c_cosleep       equ     word ptr c_qbuff + word
0024                    c_usleep        equ     word ptr c_cosleep + word
0026                    c_vsleep        equ     word ptr c_usleep + word
002C                    ccblen          equ     c_vsleep + word + 4

                        ; Flags for c_flag

0001                    cf_listcp       equ     001h    ;control P toggle
0002                    cf_compc        equ     002h    ;suppress output
0004                    cf_switchs      equ     004h    ;XIOS supports switch screening
0008                    cf_conout       equ     008h    ;ownership flag of console output
0010                    cf_vout         equ     010h    ;process writing to VOUTQ
0020                    cf_bufp         equ     020h    ;just sent a char to a VOUTQ, don't
                                                        ;echo to list if csm_ctrlP is set.

                        ;CCP state flags

0001                    csm_buffered    equ     00001h
```

```
953
954  =  0002                csm_background  equ     00002h
955  =  0004                csm_purging     equ     00004h
956  =  0008                csm_noswitch    equ     00008h
957  =  0010                csm_suspend     equ     00010h
958  =  0020                csm_abort       equ     00020h
959  =  0040                csm_filefull    equ     00040h
960  =  0080                csm_ctrlS       equ     00080h
961  =  0100                csm_ctrlJ       equ     00100h
962  =  0200                csm_ctrlP       equ     00200h
963  =
964  =                      ;LCB - list control block is first ten bytes of VCCB
965  =  000A                lcblen          equ     10
966
967                         endif
```

```
 968
 969  =                              eject 1 include init.sup
 970  =                         ;*****************************************************
 971  =                         ;*
 972  =                         ;*       MP/M-86 Supervisor Initialization
 973  =                         ;*
 974  =                         ;*****************************************************
 975  =
 976  =                              cseg
 977  =                              org     0
 978  =
 979  =0000 E93F00        0042      jmp     init            ;system initialization
 980  =0003 E9E201        01E8      jmp     entry           ;intermodule entry pt.
 981  =
 982  =                                                      ;next 3 words are set by GENSYS
 983  =0006 0000              sysdat          dw      0       ;segment
 984  =0008 E801              supervisor      dw      entry   ;offset
 985  =000A 0000                              dw      0       ;segment
 986  =
 987  =                              org     0ch
 988  =000C 06                dev_ver         db      6       ;development system data version
 989  =                                                      ;set in sysdat.dat
 990  =
 991  =0000 434F50595249                      db      'COPYRIGHT (C) 1982,'
 992           474854202843
 993           292031393832
 994           2C
 995  =0020 204449474954                      db      ' DIGITAL RESEARCH '
 996           414C20524553
 997           454152434820
 998  =0032 585858582D30                      db      'XXXX-0000-'
 999           30303030
1000  =003C 363534333231        serial        db      '654321'
1001  =
1002  =                         ;====
1003  =                         init:
1004  =                         ;====
1005  =                         ; system initialization
1006  =                         ; DS set to Sysdat Segment by loader
1007  =
1008  =                                 ;make INIT a process:
1009  =                                 ;set up init stack
1010  =
1011  =0042 FAFC                       cli ! cld                               ;Loader enters here
1012  =0044 8CD8                       mov ax,ds                               ;with interrupts
1013  =0046 8ED0BCB005                 mov ss,ax ! mov sp,offset (init_tos)    ;possibly on
1014  =
1015  =                                 ;initialize init uda
1016  =
1017  =004B BB7704                     mov bx,offset initpd
1018  =004E B86004                     mov ax,offset inituda
1019  =0051 B104D3E8                   mov cl,4 ! shr ax,cl
1020  =0055 2E0306060089               add ax,sysdat ! mov p_uda[bx],ax
```

```
1021
1022          4710
1023  =005D 8EC0                     mov es,ax
1024  =005F 268C1E2E00               mov u_wrkseg,ds
1025  =
1026  =                                   ;we now look like we are in the O.S. as usual:
1027  =                                   ;DS=SYSDAT seg, ES=UDA, U_WRKSEG=PD's DS.
1028  =
1029  =                                   ;  set up mpm entry point
1030  =
1031  =0064 1E                       push ds
1032  =0065 33C08ED8                 xor ax,ax ! mov ds,ax
1033  =0069 C7068003F200             mov i_os_ip,offset user_entry
1034  =006F 8C0E8203                 mov i_os_cs,cs
1035  =0073 1F                       pop ds
1036  =
1037  =                                   ;initialize modules
1038  =
1039  =0074 BB0400                   mov bx,mod_init
1040  =0077 53FF5F085B               push bx ! callf dword ptr rtmmod[bx] ! pop bx          ; init RTM
1041  =007C 53FF5F105B               push bx ! callf dword ptr memmod[bx] ! pop bx          ; init MEM
1042  =0081 F60646000874 008D        test module_map,bdosmod_bit ! jz nbdo
1043          05         008D
1044  =0088 53FF5F205B                   push bx ! callf dword ptr bdosmod[bx] ! pop bx     ; init BDOS
1045  =
1046  =008D F60646001074 0099 nbdo:  test module_map,ciomod_bit ! jz ncio
1047          05         0099
1048  =0094 53FF5F185B                   push bx ! callf dword ptr ciomod[bx] ! pop bx      ; init CIO
1049  =
1050  =0099 F60646002074 00A7 ncio:  test module_map,xiosmod_bit ! jz nxio
1051          07         00A7
1052  =00A0 1E06                         push ds ! push es
1053  =00A2 FF5F28071F                   callf dword ptr xiosmod[bx] ! pop es ! pop ds      ; init XIOS
1054  =
1055  =                         nxio:
1056  =                               ;  reset interrupt vectors after XIOS INIT
1057  =
1058  =00A7 8CD826D8BEDB             mov ax,ds ! sub bx,bx ! mov ds,bx
1059  =00AD C7068003F200             mov i_os_ip,offset user_entry
1060  =00B3 8C0E8203                 mov i_os_cs,cs
1061  =00B7 8ED8                     mov ds,ax
1062  =
1063  =                                   ; get Character Dev Info from XIOS
1064  =
1065  =                         if mpm
1066  =                               mov ax,io_maxconsole ! call xiosif
1067  =                               mov ncondev,bl ! mov nciodev,bl
1068  =                               mov ax,io_maxlist ! call xiosif
1069  =                               mov nlstdev,bl ! add nciodev,bl
1070  =                               mov bx,offset initpd
1071  =                               mov al,ncondev
1072  =                               mov p_lst[bx],al
1073  =                         endif
```

```
1074
1075  =                                     ; init CCB and LCB adr in INIT UDA so
1076  =                                     ; child processes will inherit them
1077  =
1078  =0089 B192CDE0                  mov cl,f_conattach ! int osint
1079  =008D B1DECDE0                  mov cl,f_lstattach ! int osint
1080  =
1081  =                                     ; Start RSPs
1082  =                           nrsp:                                   ;loop til done
1083  =00C1 2E8E1E0600                mov ds,sysdat                       ;reset DS
1084  =00C6 833E4200E320 00EC         mov cx,rspseg ! jcxz rsp_o          ;?all done?
1085  =00CC 8EC1                          mov es,cx                       ;ES->RSP
1086  =00CE 26A10000                      mov ax,es:.rsp_link             ;save next RSP
1087  =00D2 A34200                        mov rspseg,ax
1088  =00D5 2689E0000                     mov es:.rsp_link,ds             ;give Sysdat to RSP
1089  =00DA 8E1000                        mov si,rsp_pd                   ;get PD
1090  =00DD 8ED9                          mov ds,cx                       ;DS = RSP Data Seg
1091  =00DF C744160000                    mov p_mem[si],0
1092  =00E4 B190                          mov cl,f_createproc             ;Create RSP Process(s)
1093  =00E6 8B56DE0                       mov dx,si ! int osint
1094  =00EA EB35          00C1            jmps nrsp                       ;Do another...
1095  =                           rsp_o:
1096  =                                     ; terminate init process
1097  =00EC B13F                      mov cl,f_terminate
1098  =00EE B2FFCDE0                  mov dl,0ffh ! int osint
1099
```

```
1100
1101  =                         eject 1 include supif.sup
1102  =                       ;*****************************************************
1103  =                       ;*
1104  =                       ;*      SUPERVISOR INTERFACE ROUTINES
1105  =                       ;*
1106  =                       ;*****************************************************
1107  =
1108  =                       ;=========
1109  =                       user_entry:
1110  =                       ;=========
1111  =                       ;       User Entry Point - enter here from a INT 224
1112  =                       ;
1113  =                       ;               REGISTER USAGE
1114  =                       ;       ENTRY                   EXIT
1115  =                       ;       -----                   ----
1116  =                       ;  CL - Function Code     AX - Copy of BX
1117  =                       ;  DX - Param             BX - Return
1118  =                       ;  DS - Seg Addr          CX - Error Code
1119  =                       ;                         ES - Segment Return
1120  =                       ;
1121  =                       ;               DS,SI,DI,BP preserved through call
1122  =                       ;
1123  =                       ;               SETUP FOR MPM ENVIRONMENT
1124  =                       ;               -------------------------
1125  =                       ;       contents of users stack
1126  =                       ;               Flags
1127  =                       ;               CS
1128  =                       ;               IP              <- u_stack_ss,_sp
1129  =                       ;       DS =  Sysdat Segment
1130  =                       ;       ES -> user_data_area (UDA)
1131  =                       ;       DX -> function parameter
1132  =                       ;       u_wrkseg == user's DS
1133  =                       ;       u_retseg == user's ES
1134  =                                                       ;interrupts are off
1135  =                                                       ;set up MPM,CCPM environment
1136  =00F2 FC8CD8                   cld ! mov ax,ds          ;AX = user's DS
1137  =00F5 2E8E1E0600               mov ds,sysdat
1138  =00FA 881E6800                 mov bx,rlr
1139  =00FE 8F5F10                   mov ds,p_uda[bx]         ;DS = UDA segment
1140  =0101 8C063000                 mov ds:u_retseg,es       ;save user's ES
1141  =
1142  =                                                       ;U_INSYS is count of times
1143  =                                                       ;through this entry point
1144  =0105 803E6000007501 0120      cmp ds:u_insys,0 ! jne noswt    ;change stacks to UDA stack
1145        14             0120
1146  =                                                       ;if U_INSYS = 0
1147  =                                                       ;otherwise leave stack alone
1148  =
1149  =                                                       ;U_WRKSEG where is user's entry DS
1150  =                                                       ;is saved
1151  =010C A32E00                     mov ds:u_wrkseg,ax     ;wipe out earlier u_wrkseg
1152  =010F 8C163600                   mov ds:u_stack_ss,ss
```

```
1153
1154  =0113 89263400                    mov ds:u_stack_sp,sp
1155  =0117 8CD8                        mov ax,ds
1156  =0119 8ED08BC0001                 mov ss,ax ! mov sp,ulen       ;stack starts at end of UDA
1157  =011E EB07          0127          jmps uecont
1158  =                   noswt:
1159  =0120 FF362E00                    push ds:u_wrkseg              ;save current u_wrkseg on UDA stack
1160  =0124 A32E00                      mov ds:u_wrkseg,ax
1161  =
1162  =                   uecont:
1163  =0127 FB                          sti
1164  =0128 FE066000                    inc ds:u_insys
1165  =012C 8CD88EC0                    mov ax,ds ! mov es,ax         ;register moves are faster than
1166  =0130 2E8E1E0600                  mov ds,sysdat                 ;push and pop, DS=SYSDAT,ES=UDA
1167  =0135 565755                      push si ! push di ! push bp
1168  =
1169  =0138 26883E0600                  mov u_func,cl                 ;record function number in UDA
1170  =013D 32ED                        xor ch,ch                     ;call function and do
1171  =013F E83900        0178          call netchk                   ;network chk, returns BX, ES, CX
1172  =
1173  =                   coret:
1174  =0142 5D5F5E                      pop bp ! pop di ! pop si
1175  =0145 8CC0                        mov ax,es                     ;setup user's environment and return
1176  =0147 8ED8                        mov ds,ax                     ;DS = UDA segment
1177  =0149 8E063000                    mov es,ds:u_retseg            ;restore ES from entry unless
1178  =                                                               ;function returns a segment value
1179  =014D A12E00                      mov ax,ds:u_wrkseg            ;AX = user's entry DS
1180  =0150 FE0E6000750B  0161          dec ds:u_insys ! jnz nstk     ;switch back to user's stack
1181  =                                                               ;if U_INSYS = 0
1182  =0156 FA                            cli
1183  =0157 8E163600                      mov ss,ds:u_stack_ss
1184  =015B 8B263400                      mov sp,ds:u_stack_sp
1185  =015F EB04          0165            jmps ueout
1186  =                   nstk:
1187  =0161 8F062E00                    pop ds:u_wrkseg               ;restore previous U_WRKSEG if not
1188  =                                                               ;going back to user's stack
1189  =0165 8ED8          ueout:  mov ds,ax                           ;DS = user's entry DS
1190  =0167 89C3                  mov ax,bx                           ;parameter return BX = AX
1191  =0169 40                    inc ax
1192  =016A 7402          016E    jz u_err1                           ;CX=error code if AX,BX=0FFFFH
1193  =016C 33C9                    xor cx,cx                         ;CX always 0 if no error
1194  =                   u_err1:
1195  =016E 48                    dec ax
1196  =016F CF                    iret                                ;back to user ...
1197  =                                                               ;restore interrupts as they
1198  =                                                               ;were on entry
1199  =
1200  =                   sysfunc:        ; call system function
1201  =                   ;-------        ----------------------
1202  =                   ;       entry:  if CH <> 0 then CH is module number
1203  =                   ;                    (usually 1:SUP, 2:RTM, 3:MEM, 4:CIO, 5:BDOS)
1204  =                   ;                    network check is bypassed
1205  =                   ;               if CH = 0 then network check is done
```

```
1206
1207  =                         ;               CL = function #
1208  =                         ;               DX = arg
1209  =                         ;               BX = arg
1210  =                         ;       exit:   BX = return value
1211  =                         ;               ES = segment return value
1212  =                         ;               CX = error code if BX = 0FFFFH
1213  =
1214  =0170 84ED7407      0178          test ch,ch ! jz netchk                  ;network or local
1215  =0174 88C1EB43      0188              mov ax,cx ! jmps localfunc          ;in local module
1216  =
1217  =
1218  =                         illfunc:                                        ;illegal function number
1219  =0178 E9EE00        0269          jmp i_ent
1220  =
1221  =                         netchk:
1222  =                         ;       enter here for network check
1223  =                         ;       check for function numbers represented in function table,
1224  =                         ;       ENTTAB_TABLE.  Table space is saved by making the functions
1225  =                         ;       contiguous as we check the range.
1226  =                         ;       The table has entries for functions 0-69, 98-112, 128-164.
1227  =                         ;       Numbers 70-97, 112-127, 165-255 are not in the table.
1228  =                         ;       Illegal functions in this table return the error codes
1229  =                         ;       not implemented (1) or illegal function number (2).
1230  =                         ;       Function not in the table return the illegal function number
1231  =                         ;       code.
1232  =                         ;       To add a new user function modify the files SYSDAT.DAT and
1233  =                         ;       MODFUNC.DEF as well as the ranges immediately
1234  =                         ;       below.  All of the modules SUP,RTM,MEM,CIO,BDOS,SYSDAT must
1235  =                         ;       then be reassembled.
1236  =
1237  =017B 80F94557615   0195          cmp cl,69 ! jbe okfunc
1238  =0180 80E91C80F946 0178              sub cl,28 ! cmp cl,70 ! jb illfunc  ;98-112 are now 70-84
1239        72F0         0178
1240  =0188 80F9547608    0195                  cmp cl,84 ! jbe okfunc          ;128-164 are now 100-136
1241  =018D 80E90F                          sub cl,15                         ;128-164 are now 85-121
1242  =0190 80F97977E3    0178                  cmp cl,121 ! ja illfunc
1243  =
1244  =                         okfunc:                                         ;CL is now 0 - num entries
1245  =                                                                         ;- 1 in entry table
1246  =0193 88F1D1E6              mov si,cx ! shl si,1                        ;times 2 for entry table
1247  =0199 81C6A000              add si,offset sysent                        ;AH high nibble flags
1248  =019D 8B04                  mov ax,enttab_entry[si]                     ;AH low nibble module
1249  =                                                                       ;AL function number
1250  =                                                                       ;within module
1251  =019F F6C4F07417    0188          test ah,ef_network ! jz localfunc       ;not a network function
1252  =01A4 80E40F                      and ah,not ef_network                   ;turn off network bit
1253  =01A7 A04600                      mov al,module_map
1254  =01AA A8407400      0188          test al,netmod_bit ! je localfunc       ;network module is not there
1255  =01AE 50                          push ax
1256  =01AF FF1E3000                    callf dword ptr .netmod                 ;call network CL = func
1257  =01B3 3CFF                        cmp al,true                             ;if AL=0FFH do func locally
1258  =01B5 58                          pop ax                                  ;in addition over network
```

```
1259
1260  =0186 7503          0188                jne localfunc
1261  =0188 8BC3C3                               mov ax,bx ! ret
1262  =
1263  =
1264  =                                 localfunc:                                  ;not over the network
1265  =018B 80FC017410    0100          cmp ah,sup ! je insup                       ;AH=module, AL=function
1266  =018D 8AEC                        mov cl,ah
1267  =018F 8A2E46000D2ED               mov ch,module_map ! shr ch,cl               ;does module exist ?
1268  =0195 7313          019A          jnc badmod
1269  =0197 32ED49                      xor ch,ch ! dec cx                          ;make module 0 relative
1270  =019A D1E1D1E1D1E1                shl cx,1 ! shl cx,1 ! shl cx,1              ;* 8
1271  =01A0 8BF18AC8                    mov si,cx ! mov cl,al                       ;func # in CL
1272  =01A4 FF1C                        callf dword ptr module_table[si]
1273  =01A6 C3                          ret
1274  =                                 badmod:
1275  =01A7 E9B500        025F          jmp n_imp
1276  =                                 insup:
1277  =01AA 32E4                        xor ah,ah
1278  =01AC D1E08BF0                    shl ax,1 ! mov si,ax                        ; mov cx,ax
1279  =01B0 2EFFA44402                  jmp cs:supfunc[si]
1280  =
1281  =                                 ;=====
1282  =                                 entry:          ; Supervisor module entry point
1283  =                                 ;=====          --------------------------------
1284  =                                 ;
1285  =                                 ;               Arrive here usually on a CALLF using address
1286  =                                 ;               at SYSDAT:4.  Note: flag set is handled specially.
1287  =                                 ;
1288  =                                 ;       entry:  if CH <> 0 then CH is module number
1289  =                                 ;                    (usually 1:SUP, 2:RTM, 3:MEM, 4:CIO, 5:BDOS)
1290  =                                 ;                    network check is bypassed
1291  =                                 ;               if CH = 0 then network check is done;
1292  =                                 ;               CL = function #. If CH = 0 then CL is treated
1293  =                                 ;                    as a function number used with an INT 224:
1294  =                                 ;                    it is a USER function.
1295  =                                 ;                    If CH is not 0 then CL is the function number
1296  =                                 ;                    within the module specified by CH.
1297  =                                 ;                    The file MODFUNC.DEF contains the equates
1298  =                                 ;                    used for accessing functions from within the O.S.
1299  =                                 ;                    These functions are INTERNAL functions.
1300  =                                 ;                    Note the INTERNAL functions are a superset of
1301  =                                 ;                    the USER functions.  The equates for USER
1302  =                                 ;                    functions in MODFUNC.DEF have a 0 for the high byte
1303  =                                 ;                    forcing network checking from this entry point.
1304  =                                 ;                    An example: if f_conin (from MODFUNC.DEF) is
1305  =                                 ;                    equal to 0001H.  A Console output call to this
1306  =                                 ;                    entry point thus result in CH = 0, CL = 1.
1307  =                                 ;               DX = arg
1308  =                                 ;               BX = arg
1309  =                                 ;
1310  =                                 ;       exit:   BX = return value
1311  =                                 ;               ES = segment return value
```

```
1312
1313  =                       ;               CX = error code if BX = 0FFFFH
1314  =
1315  =
1316  =01E8 81F98500750F 01FD         cmp cx,f_flagset | jne notfs    ;flag set is exception:
1317  =01EE B90302                        mov cx,f_inflagset          ;do not go over network,
1318  =01F1 E87CFF        0170            call sysfunc                ;make path shorter,
1319  =01F4 8BC3                          mov ax,bx                   ;parameter return BX = AX
1320  =01F6 40                            inc ax                      ;AX=0FFFFH if error
1321  =01F7 7402          01FB           jz u_err2                    ;CX= error code if AX,BX=0FFFFH
1322  =01F9 33C9                           xor cx,cx                  ;CX=0 if no error
1323  =                       u_err2:
1324  =01FB 48                            dec ax                      ;set AX back to 0 or 0FFFFH
1325  =01FC C8                            retf                        ;DS must equal system
1326  =                       notfs:                                  ;data segment
1327  =
1328  =01FD 26890E4000                    mov u_unused,cx             ;save 16 bit function number
1329  =0202 E80100C8      0206            call osif | retf
1330  =
1331  =                       osif:                                   ;point SUP uses to get
1332  =                                                               ;to other modules
1333  =0206 26FF362E00                    push u_wrkseg               ;save current u_wrkseg
1334  =020B 8CD8                          mov ax,ds
1335  =020D 26A32E00                      mov u_wrkseg,ax
1336  =0211 2EA10600                      mov ax,sysdat
1337  =0215 8ED8                          mov ds,ax
1338  =0217 E855FF        0170            call sysfunc
1339  =021A 26A12E00                      mov ax,u_wrkseg
1340  =021E 8ED3                          mov ds,ax
1341  =0220 268F062E00                    pop u_wrkseg
1342  =0225 8BC3                          mov ax,bx                   ;BX,CX are return codes
1343  =0227 40                            inc ax                      ;AX=0FFFFH if error
1344  =0228 7402          022C           jz u_err3                    ;CX= error code if AX,BX=0FFFFH
1345  =022A 33C9                           xor cx,cx                  ;CX=0 if no error
1346  =                       u_err3:
1347  =022C 48                            dec ax                      ;set AX back to 0 or 0FFFFH
1348  =022D C3                            ret
1349  =
1350  =
1351  =                       ;=========
1352  =                       user_retf:
1353  =                       ;=========
1354  =                       ;       if a user process does a RETF to terminate,
1355  =                       ;       the process ends here.  The Load function sets up
1356  =                       ;       the default stack to point here.
1357  =
1358  =022E 8B1E6800                      mov bx,rlr
1359  =0232 8167067CFD                    and p_flag[bx],not pf_keep+pf_tempkeep+pf_ctlc+pf_sys
1360  =0237 B98F00                        mov cx,f_terminate
1361  =023A 33D2                          xor dx,dx
1362  =023C CDE0                          int osint               ;make sure the terminate
1363  =                                                           ;succeed by reseting the
1364  =                                                           ;the keep,tempkeep,ctlc,sys flags
```

```
1365
1366  =
1367  =                         ;======
1368  =                         xiosif:
1369  =                         ;======
1370  =023E FF1E2800C3                  callf dword ptr .xiosmod ! ret
1371  =
1372  =                         ;*******************************************************
1373  =                         ;*
1374  =                         ;*       Supervisor function table
1375  =                         ;*
1376  =                         ;*******************************************************
1377  =
1378  =                                 org     ((offset $)+1) AND 0fffeh       ;Word Boundary
1379  =
1380  =0244 6202               supfunc dw      n_imp           ; 0-not implemented
1381  =0246 6902                       dw      i_ent           ; 1-illegal function number
1382  =0248 7002                       dw      bver_ent        ; 2-(12)get BDOS version
1383  =024A 7E02                       dw      cbios_ent       ; 3-(50)call bios
1384  =024C D909                       dw      load_ent        ; 4-(59)user load function
1385  =024E 3903                       dw      cli_ent         ; 5-(150)CLI
1386  =0250 6811                       dw      rpl_ent         ; 6-(151)Call RPL
1387  =0252 FF0F                       dw      parse_ent       ; 7-(152)parse filename
1388  =0254 F802                       dw      sdat_ent        ; 8-(154)get sysdat addr
1389  =0256 0503                       dw      tod_ent         ; 9-(155)get tod addr
1390  =0258 DD09                       dw      load            ; 10-load function
1391  =025A 7702                       dw      over_ent        ; 11-O.S. version number
1392  =025C 3703                       dw      chain_ent       ; 12-(47)Program Chain
1393  =025E 1803                       dw      ser_ent         ; 13-(107)return serial
1394  =0260 D509                       dw      cload_ent       ; 14-chain load function
1395  =
1396
```

```
1397
1398  =                         eject ! include sysfunc.sup
1399  =                       ;*******************************************************
1400  =                       ;*
1401  =                       ;*      SYSTEM ENTRY FUNCTIONS
1402  =                       ;*
1403  =                       ;*******************************************************
1404  =
1405  =                       ;=====          ==========================
1406  =                       n_imp:          ; Function not implemented
1407  =                       ;=====          ==========================
1408  =0262 B90100                    mov cx,e_not_implemented
1409  =0265 B8FFFFC3                  mov bx,0ffffh ! ret
1410  =
1411  =                       ;====           ==========================
1412  =                       i_ent:          ; Illegal System Function
1413  =                       ;====           ==========================
1414  =0269 B90200                    mov cx,e_bad_entry
1415  =026C B8FFFFC3                  mov bx,0ffffh ! ret
1416  =
1417  =                       ;=======        ====================
1418  =                       bver_ent:       ; Get BDOS Version #
1419  =                       ;=======        ====================
1420  =0270 881E7A0033C9              mov bx,bvernum ! xor cx,cx ! ret
1421        C3
1422  =
1423  =                       ;=======        ====================
1424  =                       over_ent:       ; Get O.S. Version #
1425  =                       ;=======        ====================
1426  =0277 881E7C0033C9              mov bx,osvernum ! xor cx,cx ! ret
1427        C3
1428  =
1429  =                       if mpm
1430  =
1431  =                       ;=========      ==================      ============
1432  =                       cbios_ent:      ; Direct BIOS call      MPM 2.x ONLY
1433  =                       ;=========      ==================      ============
1434  =
1435  =                               mov si,dx
1436  =                               push ds ! mov ds,u_wrkseg
1437  =                               mov al,[si] ! mov cx,1[si]
1438  =                               mov dx,3[si] ! pop ds
1439  =                               cmp al,1 ! ja goxios                    ;if BOOT,WBOOT: terminate
1440  =                                   mov cx,f_terminate
1441  =                                   mov dx,0 ! jmp osif
1442  =                       goxios:
1443  =                               cmp al,7 ! jbe gx1                      ;7=reader input
1444  =                                   cmp al,15 ! je gx1                  ;15=list status
1445  =                                       mov bx,0ffffh ! mov cx,e_bad_entry
1446  =                                       ret
1447  =                       gx1:    mov bx,rlr
1448  =                               cmp al,4 ! ja xlst                      ;4=console output
1449  =                                   mov dl,p_cnsl[bx]
```

```
1450
1451  =                                 cmp al,4 ! je jxio
1452  =                                     mov cl,dl ! jmps jxio
1453  =
1454  =                           xlst:   cmp al,6 ! je jxio
1455  =                                   cmp al,7 ! je jxio
1456  =                                       mov dl,p_lst[bx]
1457  =                                       cmp al,15 ! jne jxio
1458  =                                           mov cl,dl
1459  =                           jxio:   sub al,2 ! mov ah,0
1460  =                                   jmp xiosif
1461  =                           endif                                   ;end of MP/M direct BIOS call
1462  =
1463  =                           if ccpm                                 ;CCP/M direct BIOS call
1464  =
1465  =                           ;=========      ==================      =============
1466  =                           cbios_ent:      ; Direct BIOS call      CCPM 2.x ONLY
1467  =                           ;=========      ==================      =============
1468  =
1469  =                                           ;  DI = 0 if last call was also func 50.
1470  =                                           ;  DI = 0ffffh if it wasn't
1471  =
1472  =                           ;       xor di,di
1473  =                           ;       cmp u_func,50
1474  =                           ;       je c_next50
1475  =                           ;         dec di
1476  =                           c_next50:
1477  =                           ;       mov u_func,50
1478  =027E 8BF2                         mov si,dx
1479  =0280 8CDD                         mov bp,ds                       ;user register for speed
1480  =0282 268E1E2E00                   mov ds,u_wrkseg
1481  =0287 8A048B4C01                   mov al,[si] ! mov cx,1[si]
1482  =028C 885403                       mov dx,3[si]
1483  =028F 8EDD                         mov ds,bp                       ;DS=SYSDAT
1484  =
1485  =0291 3C02750E      02A3           cmp al,2 ! jne not_consts       ;optimize constat
1486  =                           ;         test di,di                    ;DI=0 if last call was func 50
1487  =                           ;         jnz go_cio
1488  =                           ;           mov si,rlr
1489  =                           ;           mov bl,ncns                 ;is it a virtual console ?
1490  =                           ;           cmp bl,p_cnst[si]
1491  =                           ;           jae go_cio
1492  =                           ;get_status:
1493  =                           ;             mov si,u_conccb           ;if it was U_CCB is valid
1494  =                           ;             xor bx,bx
1495  =                           ;             cmp c_nchar[si],0
1496  =                           ;             jnz s_gotchar
1497  =                           ;               mov bl,c_numchars[si]   ;number of chars in VINQ
1498  =                           ;               jmps s_quick
1499  =                           go_cio:
1500  =0295 B91804                           mov cx,f_ciostat            ;doesn't change console mode
1501  =0298 E865FF        0206               call osif
1502  =                           ;s_quick:
```

```
1503
1504  =0296 84D87403      02A2          test bl,bl ! jz x_cs          ;returns 1 or 0 (or char count)
1505  =                           ;s_gotchar:
1506  =029A B8FF00                       mov ax,0ffh                 ;BIOS returns 0ffh or 0
1507  =                           x_cs:
1508  =02A2 C3                          ret
1509  =
1510  =                           not_consts:
1511  =02A3 3C047508      02AF          cmp al,4 ! jne not_conout     ;console output
1512  =02A7 8AD1                        mov dl,cl                   ;character to send
1513  =02A9 B90304                      mov cx,f_rconout
1514  =02AC E957FF        0206          jmp osif
1515  =
1516  =                           not_conout:
1517  =02AF 3C017709      02BC          cmp al,1 ! ja goxios
1518  =02B3 B98F00                      mov cx,f_terminate          ;cold or warm boot
1519  =02B6 BA0000E94AFF  0206          mov dx,0 ! jmp osif
1520  =                           goxios:
1521  =02BC 3C077608      02CB          cmp al,7 ! jbe gx1          ;BIOS 2-7 and 15 are ok
1522  =02C0 3C0F7407      02CB          cmp al,15 ! je gx1
1523  =02C4 B8FFFFB90200                    mov bx,0ffffh ! mov cx,e_bad_entry
1524  =02CA C3                              ret
1525  =                           gx1:
1526  =02CB 3C037506      02D5          cmp al,3 ! jne not_conin      ;console input
1527  =02CF B90204                      mov cx,f_rconin
1528  =02D2 E931FF        0206          jmp osif                    ;BIOS return in AL and BL
1529  =                           not_conin:
1530  =02D5 3C057508      02E1          cmp al,5 ! jne not_listout
1531  =02D9 8AD1                        mov dl,cl
1532  =02DB B90500                      mov cx,f_lstout
1533  =02DE E925FF        0206          jmp osif
1534  =                           not_listout:
1535  =02E1 3C067506      02EB          cmp al,6 ! jne not_auxout
1536  =02E5 B80600                      mov ax,io_auxout
1537  =02E8 E953FF        023E          jmp xiosif
1538  =                           not_auxout:
1539  =02EB 3C077506      02F5          cmp al,7 ! jne not_auxin
1540  =02EF B80500                      mov ax,io_auxin
1541  =02F2 E949FF        023E          jmp xiosif
1542  =                           not_auxin:
1543  =02F5 B80300                      mov ax,io_listst            ;when we move this to CIO
1544  =02F8 E943FF        023E          jmp xiosif                  ;check for ownership
1545  =                           endif                             ;end of CCP/M direct BIOS
1546  =
1547  =
1548  =                           ;========       ==============================
1549  =                           sdat_ent:       ; Ret Addr of System Data Area
1550  =                           ;========       ==============================
1551  =
1552  =02FB 268C1E3000                  mov u_retseg,ds
1553  =0300 33DB8BCBC3                  xor bx,bx ! mov cx,bx ! ret
1554  =
1555  =                           ;=======        ============================
```

```
1556
1557  =                         tod_ent:        ; Return current Time of Day
1558  =                         ;=======        ============================
1559  =                                         ; copy tod struct into user area
1560  =0305 06268E062E00                push es ! mov es,u_wrkseg
1561  =030B 8BFA                        mov di,dx
1562  =030D BE7E00B90500                mov si,offset tod ! mov cx,todlen
1563  =0313 F3A4                        rep movsb
1564  =0315 0733C988D9C3                pop es ! xor cx,cx ! mov bx,cx ! ret
1565  =
1566  =                         ;=======        ======================
1567  =                         ser_ent:        ; Return Serial Number
1568  =                         ;=======        ======================
1569  =                                         ; copy serial field into user area
1570  =0318 06268E062E00                push es ! mov es,u_wrkseg
1571  =031E 8BFABE3C00                  mov di,dx ! mov si,offset serial
1572  =0323 1E8CC88ED8                  push ds ! mov ax,cs ! mov ds,ax
1573  =0328 B90300F3A5                  mov cx,3 ! rep movsw
1574  =032D 1F07                        pop ds ! pop es
1575  =032F 33C988D9C3                  xor cx,cx ! mov bx,cx ! ret
1576
```

```
1577
1578  =                         eject ! include command.sup
1579  =                       ;*********************************************************
1580  =                       ;*
1581  =                       ;*      Command Line Interpreter, Program Chain
1582  =                       ;*
1583  =                       ;*********************************************************
1584  =                       ;==========
1585  =                       chain_ent:
1586  =                       ;==========
1587  =
1588  =0337 33D2                        xor dx,dx
1589  =
1590  =                       ;=======
1591  =                       cli_ent:
1592  =                       ;=======
1593  =                       ; Create a process based on an Ascii Command Line.
1594  =                       ; The command Line is first parsed and an FCB is
1595  =                       ; initialized as a result.
1596  =                       ; An attempt is made to open a queue with the filename
1597  =                       ; of the FCB, and with the RSP flag on.
1598  =                       ; The command tail is written to the queue if it is found.
1599  =                       ; If the queue cannot be opened or is not and RSP type then
1600  =                       ; we try and load the command from disk.
1601  =                       ; If the write queue fails we return e_q_full error code and
1602  =                       ; do not look on the disk.
1603  =                       ; After the queue write, an console assign call attempted to
1604  =                       ; a process with the same name and the PF_DSKLD flag off.
1605  =                       ; Irregardless of the success of the assign we then return.
1606  =                       ;
1607  =                       ; If the RSP queue cannot be opened, or is not an RSP type queue then
1608  =                       ; we make the name type in the
1609  =                       ; FCB to be "CMD" and attempt to open the file.  If this fails,
1610  =                       ; so do we.
1611  =                       ; We then obtain a Process Descriptor from the
1612  =                       ; PD table.  Again we fail if it does.
1613  =                       ; On a successful open, we call the BDOS load function.
1614  =                       ; If the load fails, so do we.  The PD is
1615  =                       ; initialized, the default console is assigned to the
1616  =                       ; PD, the PF_DSKLD flag turned on,
1617  =                       ; and a create process call is made.
1618  =                       ;
1619  =                       ;       input:  DX -> command buffer in u_wrkseg
1620  =                       ;               it is assumed that the calling process
1621  =                       ;               is attached to its default console
1622  =                       ;               and is willing to lose it since it
1623  =                       ;               will be handed to the newly created
1624  =                       ;               process.
1625  =                       ;               if DX = 0, assume chain w/command in DMA
1626  =                       ;
1627  =                       ;       output: BX = 0 if successful
1628  =                       ;                  = 0ffffh if failure has occured
1629  =                       ;               CX = error code
```

```
1630
1631  =
1632  =0339 8B1E68008B47              mov bx,rlr ! mov ax,p_flag[bx]
1633        06
1634  =0340 50                        push ax                         ;save flags on stack
1635  =0341 000012                    or ax,pf_tempkeep + pf_noctls
1636  =0344 894706                    mov p_flag[bx],ax
1637  =0347 52                        push dx                         ;save parameter
1638  =0348 E85906        09A4        call cli_sync                   ;rentrancy stops with sync call
1639  =034B 5A                        pop dx                          ;we can save the flags
1640  =034C 8F052103                  pop cli_pflag                   ;and other variables in the CLI
1641  =                                                               ;data area
1642  =                                   ; we have CLI SYNC
1643  =                                   ; Check for Chain
1644  =
1645  =0350 C606230300                mov cli_chain,false
1646  =0355 83FA00750A    0364        cmp dx,0 ! jne cli_cli
1647  =035A C6062303FF                    mov cli_chain,true
1648  =035F C606240300                    mov cli_term,false
1649  =                         cli_cli:
1650  =                                   ; initialize defaults from parent PD
1651  =
1652  =0364 C606760400                mov cli_dfil,false
1653  =0369 803E4F00FF75 0387         cmp dayfile,0ffh ! jne nodf
1654        17           0387
1655  =0370 52B9A200                      push dx ! mov cx,f_cconattch
1656  =0374 E88FFE5A      0206            call osif ! pop dx
1657  =0378 83F900750A    0387            cmp cx,0 ! jne nodf
1658  =037D C6067604FF                        mov cli_dfil,true
1659  =0382 52                            push dx
1660  =0383 E83305        0889            call prtime
1661  =0386 5A                            pop dx
1662  =0387 8B1E6800            nodf:     mov bx,rlr
1663  =038B 891EA603                      mov cli_ppd,bx
1664  =038F 8A4F12                        mov cl,p_dsk[bx]
1665  =0392 880E7404                      mov cli_dsk,cl
1666  =0396 8A4F13                        mov cl,p_user[bx]
1667  =0399 880E7304                      mov cli_user,cl
1668  =039D 8A4F20                        mov cl,p_cns[bx]
1669  =03A0 880E7204                      mov cli_cns,cl
1670  =03A4 268A0E1000                    mov cl,u_error_mode
1671  =03A9 880E7504                      mov cli_err_mode,cl
1672  =03AD 268B0E0200                    mov cx,u_dma_ofst
1673  =03B2 890E1D03                      mov cli_dma_ofst,cx
1674  =03B6 26830E0400                    mov cx,u_dma_seg
1675  =03BB 890E1F03                      mov cli_dma_seg,cx
1676  =03BF C7066D040000                  mov clierr,0
1677  =
1678  =                                   ; copy command into local area
1679  =
1680  =03C5 803E2303FF75 03E8         cmp cli_chain,true ! jne cli_cpy
1681        1C           03E8
1682  =03CC 061E                          push es ! push ds
```

```
1683
1684  =03CE 2E8E060600                  mov es,sysdat
1685  =03D3 8B351D03                    mov si,cli_dma_ofst
1686  =03D7 BFA303                      mov di,offset cli_cmdtail
1687  =03DA 8E1E1F03                    mov ds,cli_dma_seg
1688  =03DE B94000                      mov cx,040H
1689  =03E1 F3A5                        rep movsw
1690  =03E3 1F07                        pop ds ! pop es
1691  =03E5 E91E00        0406          jmp cli_parse
1692  =
1693  =                           cli_cpy:
1694  =03E8 061E                        push es ! push ds
1695  =03EA 268E1E2E0007                mov ds,u_wrkseg ! pop es
1696  =                                     ; DS=Wrkseg, ES=Sysdat, SP->UDA
1697  =                                     ; copy clicb_net
1698  =03F0 8BF28BFA5B03                mov si,dx ! mov di,offset cli_net
1699  =03F5 A4                          movsb
1700  =                                     ; copy command
1701  =03F6 8BF283C601                  mov si,dx ! add si,clicb_cmd
1702  =03FB BFA803                      mov di,offset cli_cmdtail
1703  =03FE B98100                      mov cx,clicblen-clicb_cmd
1704  =0401 F3A4                        rep movsb
1705  =0403 061F07                      push es ! pop ds ! pop es
1706  =
1707  =                                     ;parse the command
1708  =                           cli_parse:
1709  =0406 E8A905E307    09B2          call pfn ! jcxz cli_gprs
1710  =040B 890E6D04                        mov clierr,cx
1711  =040F E9D603        07E8              jmp cli_exit
1712  =                           cli_gprs:
1713  =
1714  =0412 E85B04        087D          call shtal
1715  =
1716  =                                     ;fcb has parsed filename
1717  =                                     ;if not explicit disk then
1718  =                                     ;   if not RSP try CMD
1719  =                                     ;else try CMD
1720  =
1721  =0415 803E2A040075  0439          cmp cli_fcb,0 ! jne cli_ffload
1722        1D            0439
1723  =041C BB2A04                      mov bx,(offset cli_fcb)
1724  =041F 807F1A007514  0439          cmp fcb_plen[bx],0 ! jne cli_ffload
1725  =0425 E81C00750F    0444              call cli_checkque ! jnz cli_ffload
1726  =                                                       ;successful RSP access
1727  =                           cli_qful:
1728  =042A 803E2303FF75  0436                  cmp cli_chain,true ! jne cli_exit2
1729        05            0436
1730  =0431 C6052403FF                              mov cli_term,true
1731  =0436 E9AF03        07E8      cli_exit2:      jmp cli_exit
1732  =                           cli_ffload:
1733  =0439 83F90F7571    04AF          cmp cx,e_q_full ! jne cli_fload
1734  =043E 890E6D04EBE6  042A              mov clierr,cx ! jmps cli_qful
1735  =
```

```
1736
1737  =                         cli_checkque:
1738  =                         ;------------
1739  =                         ;       output: z flag on if successful
1740  =
1741  =                                 ;copy fcb.name to qpb.name
1742  =
1743  =0444 BE2804                     mov si,(offset cli_fcb)+fcb_name
1744  =0447 BF5304                     mov di,(offset cli_cuspqpb)+qpb_name
1745  =044A B90400061E07               mov cx,qnamsiz/2 ! push es ! push ds ! pop es
1746  =0450 56F3A5                     push si ! rep movsw
1747  =
1748  =                                 ;copy fcb.name to acb.name
1749  =
1750  =0453 5E830400                   pop si ! mov cx,qnamsiz/2
1751  =0457 BF5F04                     mov di,(offset cli_acb)+acb_name
1752  =045A F3A507                     rep movsw ! pop es
1753  =
1754  =                                 ;open queue
1755  =
1756  =045D B98700BA4804               mov cx,f_qopen ! mov dx,(offset cli_cuspqpb)
1757  =0463 E8A0FDE304    0206         call osif ! jcxz cli_goodq
1758  =0468 83F900C3            retcli1:    cmp cx,0 ! ret              ;CX = 0ffffh on error
1759  =
1760  =                                 ;we successfully opened the queue
1761  =                                 ;now check RSP flag
1762  =                         cli_goodq:
1763  =046C B34804                     mov bx,offset cli_cuspqpb
1764  =046F 8B5F02                     mov bx,qpb_qaddr[bx]
1765  =0472 F74704080075 047E          test q_flags[bx],qf_rsp ! jnz cli_gq
1766        05           047E
1767  =0479 B90900E8EA    0468             mov cx,e_no_queue ! jmps retcli1
1768  =
1769  =                                 ;write command tail to queue
1770  =
1771  =047E B98C00BA4804          cli_gq: mov cx,f_qwrite ! mov dx,offset cli_cuspqpb
1772  =0484 E87FFDE305    0206         call osif ! jcxz cli_qw
1773  =0489 B90F00E8DA    0468             mov cx,e_q_full ! jmps retcli1
1774  =
1775  =                                 ;successful queue write, assign console
1776  =
1777  =048E 803E7604FF75 0498 cli_qw: cmp cli_dfil,true ! jne noqm
1778        03           0498
1779  =0495 E8B004        0948             call prcusp
1780  =0498 BB5B04              noqm:   mov bx,offset cli_acb
1781  =049B A072048807                  mov al,cli_cns ! mov acb_cns[bx],al
1782  =04A0 C6470100                    mov acb_match[bx],false
1783  =04A4 C7470201 00                 mov acb_pd[bx],1                ;match on PD with DSKLD flag off
1784  =04A9 E80E0533C9C3 09BA           call conasn ! xor cx,cx ! ret
1785  =
1786  =                         cli_fload:
1787  =                         ;---------
1788  =                         ; Try to Load a file for execution
```

```
1789
1790  =                       ; The Command Line Parsed correctly and we have an FCB
1791  =                       ; set up.  We already know there isn't a queue and a
1792  =                       ; process by the same name  as the command.
1793  =
1794  =                               ; Obtain a Process Descriptor
1795  =
1796  =044F 9CFA8B1E5C00              pushf ! cli ! mov bx,pul
1797  =0485 83FB00750A    04C4        cmp bx,0 ! jne cli_gpd
1798  =048A 9DC7066D040C                  popf ! mov clierr,e_no_pd
1799        00
1800  =04C1 E92403        07E8            jmp cli_exit
1801  =                         cli_gpd:
1802  =04C4 8B3789365C00              mov si,p_link[bx] ! mov pul,si
1803  =04CA 9D891E6B04                popf ! mov cli_pd,bx
1804  =                                       ; zero PD
1805  =04CF 061E07                    push es ! push ds ! pop es
1806  =04D2 8BFBB91800                mov di,bx ! mov cx,pdlen/2
1807  =04D7 33C0F3AB                  xor ax,ax ! rep stosw
1808  =04DB 07                        pop es
1809  =
1810  =                               ; Initialize the PD for Load
1811  =
1812  =04DC 8B1E6B04                  mov bx,cli_pd
1813  =04E0 C747061000                mov p_flag[bx],pf_table
1814  =04E5 8BFB83C708                mov di,bx ! add di,p_name
1815  =04EA BE2A0483C601              mov si,offset cli_fcb ! add si,fcb_name
1816  =04F0 068CD88EC0                push es ! mov ax,ds ! mov es,ax
1817  =04F5 B90400F3A5                mov cx,pnamsiz/2 ! rep movsw
1818  =04FA 07                        pop es
1819  =04FB 8B366800                  mov si,rlr
1820  =04FF A07404884712              mov al,cli_dsk ! mov p_dsk[bx],al        ;inherit parents drive/user
1821  =0505 A07304884713              mov al,cli_user ! mov p_user[bx],al      ;even if we load off another
1822  =                                                                        ;drive or user
1823  =
1824  =                                                                        ;this should be in
1825  =                                                                        ;process create ?
1826  =
1827  =050B A07204884720              mov al,cli_cns ! mov p_cns[bx],al
1828  =0511 8A4424                    mov al,p_lst[si]
1829  =                         if mpm
1830  =                          sub al,ncondev
1831  =                         endif
1832  =0514 884724                    mov p_lst[bx],al
1833  =
1834  =                               ; 3. Open the file
1835  =
1836  =0517 8E3404                    mov si,(offset cli_fcb)+fcb_pwd
1837  =051A BF2503                    mov di,offset cli_dma
1838  =051D 062E8E060600              push es ! mov es,sysdat
1839  =0523 B90400F3A5                mov cx,4 ! rep movsw
1840  =0528 07                        pop es
1841  =0529 26C706020025              mov u_dma_ofst,offset cli_dma
```

```
1842
1843          03
1844  =0530 268C1E0400              mov u_dma_seg,ds
1845  =
1846  =0535 BE2A04                  mov si,offset cli_fcb
1847  =0538 C6440943                mov byte ptr fcb_type[si],'C'
1848  =053C C6440A4D                mov byte ptr fcb_type+1[si],'M'
1849  =0540 C6440B44                mov byte ptr fcb_type+2[si],'D'
1850  =
1851  =                                     ; Open the CMD file
1852  =
1853  =0544 26C606100FFE            mov u_error_mode,0feh
1854  =054A E84504          0992    call flopn
1855  =054D 80FBFF7562      05B4    cmp bl,0ffh ! jne cli_gopen
1856  =
1857  =                                     ; on failure,
1858  =                                     ; if default is not system disk
1859  =                                     ;    and not an explicit disk then
1860  =                                     ;       try CMD file on System disk
1861  =
1862  =0552 80FF00754F      05A6        cmp bh,0 ! jne cli_bo               ;extended error
1863  =0557 8A0E4300                    mov cl,srchdisk
1864  =055B 3A0E74047445    05A6        cmp cl,cli_dsk ! je cli_bo          ;already on system disk
1865  =0561 803E2A040075    05A6        cmp cli_fcb,0 ! jne cli_bo          ;check for explicit
1866          3F            05A6
1867  =                                                                     ;select
1868  =                                         ; try system disk
1869  =
1870  =                         ;           mov bx,rlr
1871  =                         ;           mov p_dsk[bx],cl
1872  =0568 FEC1                          inc cl
1873  =056A 880E2A04                      mov cli_fcb,cl                  ;set drive byte to
1874  =056E E82104          0992          call flopn                      ;system disk
1875  =0571 80FBFF7430      05A6          cmp bl,0ffh ! je cli_bo
1876  =
1877  =                                         ;make sure SYS attribute is on...
1878  =
1879  =0576 BE2A04                       mov bx,offset cli_fcb
1880  =0579 F6470A807535    05B4          test byte ptr fcb_type+1[bx],080h ! jnz cli_gopen
1881  =
1882  =                                       ;We opened a NON-SYS file.  Do explicit open
1883  =                                       ;on user zero if not already
1884  =
1885  =057F E80304          098A            call flclse
1886  =0582 881E6800                        mov bx,rlr
1887  =0586 807F1300741F    05AB            cmp p_user[bx],0 ! je cli_boe
1888  =058C C6471300                            mov p_user[bx],0
1889  =0590 E8FF03          0992                call flopn
1890  =0593 80FBFF740E      05A6                cmp bl,0ffh ! je cli_bo
1891  =0598 BE2A04                                  mov bx,offset cli_fcb
1892  =059B F6470A80                                test byte ptr fcb_type+1[bx],080h
1893  =059F 7513            05B4                    jnz cli_gopen
1894  =05A1 E8E503          098A                        call flclse
```

```
1895
1896  =05A4 EB05          05AB                                jmps cli_boe
1897  =
1898  =                                                   ;could not find CMD file
1899  =
1900  =05A6 80FF007506    05B1 cli_bo:            cmp bh,0 ! jne cli_rmpd2
1901  =05AB C7066D041E00       cli_boe:               mov clierr,e_bad_open
1902  =05B1 E90202        07B6 cli_rmpd2:         jmp cli_rmpd
1903  =                        cli_gopen:
1904  =
1905  =                                        ; 9. Call the load function
1906  =
1907  =05B4 8B1E6800                  mov bx,rlr
1908  =05B8 F74706800074 05C8          test p_flag[bx],pf_ctlc ! jz cli_ld1
1909        09           05C8
1910  =05BF BBFFFFB92800                   mov bx,0ffffh ! mov cx,e_abort
1911  =05C5 E99F00        0667             jmp cli_cl
1912  =                        cli_ld1:
1913  =05C8 803E9000FF75 0602          cmp cmod,true !  jne not_cmod
1914        33           0602
1915  =05CF 8B1E6B04                       mov bx,cli_pd
1916  =05D3 8E2A04                         mov si,offset cli_fcb
1917  =
1918  =                                        ;test F1 bit
1919  =05D6 C6471800                       mov p_cmod[bx],0
1920  =05DA F64401807404 05E4              test byte ptr fcb_name[si],080h ! jz not_f1
1921  =05E0 804F1880                           or p_cmod[bx],080h
1922  =
1923  =                                        ;test F2 bit
1924  =05E4 F64402807404 05EE not_f1:      test byte ptr fcb_name+1[si],080h ! jz not_f2
1925  =05EA 804F1840                           or p_cmod[bx],040h
1926  =
1927  =                                        ;test F3 bit
1928  =                        not_f2:
1929  =
1930  =                        ;if mpm         dave brown test
1931  =05EE F6440380                       test byte ptr fcb_name+2[si],080h
1932  =05F2 7404          05F8             jz not_f3
1933  =                        ;endif
1934  =
1935  =05F4 804F1820                           or p_cmod[bx],020h
1936  =
1937  =                                        ;test F4 bit
1938  =05F8 F64404807404 0602 not_f3:      test byte ptr fcb_name+3[si],80h ! jz not_cmod
1939  =05FE 804F1870                           or p_cmod[bx],070h
1940  =                        not_cmod:
1941  =
1942  =0602 803E2303FF75 0641          cmp cli_chain,true ! jne cli_kuda
1943        38           0641
1944  =0609 8B1E6B04                       mov bx,cli_pd
1945  =060D 8C4710                         mov p_uda[bx],es
1946  =0610 B8B004                         mov ax,offset inituda
1947  =0613 B104D3E8                       mov cl,4 ! shr ax,cl
```

```
1948
1949  =0617 2E03060600              add ax,sysdat
1950  =061C 8CC38EC08F00            mov bx,es ! mov es,ax ! mov di,0
1951        00
1952  =0623 8ED38BF7                mov ds,bx ! mov si,di
1953  =0627 B98000                  mov cx,ulen/2
1954  =062A F3A5                    rep movsw
1955  =062C 9CFA58                  pushf ! cli ! pop dx
1956  =062F 8CC0                    mov ax,es
1957  =0631 2E8E1E0600              mov ds,sysdat
1958  =0636 8ED0                    mov ss,ax
1959  =0638 8B1E6800                mov bx,rlr
1960  =063C 894710                  mov p_uda[bx],ax
1961  =063F 5292                    push dx ! popf
1962  =                       cli_kuda:
1963  =0641 803E7604FF75 064E       cmp cli_dfil,true ! jne noprfil
1964        06           0656
1965  =0648 E8A502       08F0       call prfilnam
1966  =064B E85002       089E       call crlf
1967  =
1968  =064E 8B1E6B04         noprfil:mov bx,cli_pd
1969  =0652 BA2A04                  mov dx,offset cli_fcb
1970  =0655 B90A01                  mov cx,f_load
1971  =0658 803E2303FF75 0664       cmp cli_chain,true ! jne cli_ld
1972        05           0664
1973  =065F C6052403FF              mov cli_term,true
1974  =0664 E89FFB       0206 cli_ld: call osif
1975  =0667 5351           cli_cl: push bx ! push cx
1976  =0669 26C606100000            mov u_error_mode,0
1977  =066F E81803       098A       call flclse
1978  =0672 595B                    pop cx ! pop bx
1979  =0674 E30F         0685       jcxz cli_gload
1980  =0676 83F9287503   067E           cmp cx,e_abort ! jne cli_lnab
1981  =067B E93801       07B6               jmp cli_rmpd
1982  =067E 890E6D04         cli_lnab:   mov clierr,cx
1983  =0682 E93101       07B6           jmp cli_rmpd
1984  =                       cli_gload:
1985  =0685 891E6F04                mov cli_bpage,bx
1986  =
1987  =                                 ; 9a. Parse Command Tail
1988  =
1989  =                                 ;       copy cmdtail to user DMA buffer
1990  =
1991  =0689 068E066F04             push es ! mov es,cli_bpage
1992  =068E BF8100                  mov di,offset bpg_dma+1
1993  =0691 BEA303                  mov si,offset cli_cmdtail
1994  =0694 B97F00                  mov cx,127
1995  =0697 F3A407                  rep movsb ! pop es
1996  =
1997  =                                 ;       count cmd length and convert
1998  =                                 ;       to upper case
1999  =
2000  =069A 1E8E1E6F04              push ds ! mov ds,cli_bpage
```

```
2001
2002  =069F B100BF8100              mov cl,0 ! mov di,offset bpg_dma+1
2003  =                       ncmdchar:
2004  =0644 803D007412   0688       cmp byte ptr [di],0 ! je endtail
2005  =
2006  =                                 ; convert CMD tail to UPPER CASE
2007  =
2008  =06A9 803D617208   0686           cmp byte ptr [di],'a' ! jb nlow
2009  =06AC 803D7A7703   0686               cmp byte ptr [di],'z' ! ja nlow
2010  =06B3 80255F                           and byte ptr [di],05fh
2011  =                       nlow:
2012  =
2013  =
2014  =06B6 47FEC1E8E9   06A4           inc di ! inc cl ! jmps ncmdchar
2015  =                       endtail:
2016  =06BB 880E80001F               mov bpg_dma,cl ! pop ds
2017  =
2018  =                                 ;       load disk init, location 50H
2019  =                                 ;       of base page is done in LOAD
2020  =
2021  =06C0 068E066F04               push es ! mov es,cli_bpage
2022  =
2023  =                                 ;       init default fcb
2024  =
2025  =06C5 BF5C00                   mov di,offset bpg_fcb0
2026  =06C8 33C0AA                   xor ax,ax ! stosb                       ;default disk
2027  =06CB B020                     mov al,' '
2028  =06CD B90B00F3AA               mov cx,11 ! rep stosb                   ;name,type
2029  =06D2 33C0                     xor ax,ax
2030  =06D4 B90200F3AB               mov cx,2 ! rep stosw                    ;other
2031  =06D9 1E061F                   push ds ! push es ! pop ds
2032  =06DC BE5C00                   mov si,offset bpg_fcb0
2033  =06DF B90800F3A5               mov cx,8 ! rep movsw
2034  =06E4 1F07                     pop ds ! pop es
2035  =
2036  =                                 ;       if cmdtail, parse
2037  =
2038  =06E6 803EA8030074 0759       cmp cli_cmdtail,0 ! je ctdone
2039        6C             0759
2040  =06ED E8C2D2         0982           call pfn
2041  =06F0 83F8FF7464     0759           cmp bx,0ffffh ! je ctdone
2042  =
2043  =                                 ;       copy fcb to user fcb front
2044  =
2045  =06F5 068E066F04                   push es ! mov es,cli_bpage
2046  =06FA BF5C00                       mov di,offset bpg_fcb0
2047  =06FD BE2A04                       mov si,offset cli_fcb
2048  =0700 8B4418                       mov ax,fcb_pptr[si]
2049  =                                         ; AX->password in CLI_CMDTAIL
2050  =0703 2DA803                       sub ax,offset cli_cmdtail
2051  =0706 05d100                       add ax,offset bpg_dma + 1
2052  =                                         ; AX->password in Base Page
2053  =0709 26A35100                     mov es:bpg_pwlptr,ax
```

```
2054
2055  =070D 8A441A                      mov al,fcb_plen[si]
2056  =0710 26A25300                    mov es:bpg_pwllen,al
2057  =0714 B90800F3A5C7                mov cx,8 ! rep movsw ! pop es
2058  =
2059  =                                 ;       if more cmdtail, parse
2060  =
2061  =071A 83FB00743A    0759         cmp bx,0 ! je ctdone
2062  =071F FF36570443                      push cli_pcb ! inc bx
2063  =0724 891E5704                        mov cli_pcb,bx
2064  =0728 E88702        0982              call pfn
2065  =072B 8F065704                        pop cli_pcb
2066  =072F 83FBFF7425    0759              cmp bx,0ffffh ! je ctdone
2067  =
2068  =                                 ;       copy 2nd fcb front
2069  =
2070  =0734 069E066F04                      push es ! mov es,cli_bpage
2071  =0739 BF5C00                          mov di,offset bpg_fcb1
2072  =073C BE2A04                          mov si,offset cli_fcb
2073  =073F 8B4418                          mov ax,fcb_pptr[si]
2074  =                                           ; AX->password in CLI_CMDTAIL
2075  =0742 2DA303                          sub ax,offset cli_cmdtail
2076  =0745 058100                          add ax,offset bpg_dma + 1
2077  =                                           ; AX->password in Base Page
2078  =0748 26A35400                        mov es:bpg_pw2ptr,ax
2079  =074C 8A441A                          mov al,fcb_plen[si]
2080  =074F 26A25600                        mov es:bpg_pw2len,al
2081  =0753 B90800                          mov cx,8
2082  =0756 F3A5                            rep movsw
2083  =0758 07                              pop es
2084  =                         ctdone:
2085  =
2086  =                                 ; 10. Create the process
2087  =
2088  =0759 803E2303FF75 0769          cmp cli_chain,true ! jne nprior
2089        09              0769
2090  =0760 B99100                          mov cx,f_setprior
2091  =0763 BA0100E89DFA 0206               mov dx,1 ! call osif
2092  =                         nprior:
2093  =0769 8B365B04814C                mov si,cli_pd ! or p_flag[si],pf_dskld  ;from disk, to differ
2094        060020
2095  =076E 8BD5B99000                  mov dx,si ! mov cx,f_createproc         ;from RSP with same name
2096  =0777 E88CFA        0206          call osif
2097  =
2098  =                                 ; 11. Assign Console to new process
2099  =
2100  =077A 881E6800                    mov bx,rlr
2101  =077E 8167067FFF                  and p_flag[bx],not pf_ctlc
2102  =
2103  =                                 ; Check to see if user hit CTRL D
2104  =0783 F74706000474 0798           test p_flag[bx],pf_ctld ! jz asgn
2105        11              0798
2106  =078A 816706FFFB                      and p_flag[bx],not pf_ctld
```

```
2107
2108  =078F 881E6804                    mov bx,cli_pd
2109  =0793 814F060004                  or p_flag[bx],pf_ctld
2110  =0798 E94D00        07E8          jmp cli_exit
2111  =079B 885804              asgn:   mov bx,offset cli_acb
2112  =079E A072048807                  mov al,cli_cns ! mov acb_cns[bx],al
2113  =07A3 A1580489470 2               mov ax,cli_pd ! mov acb_pd[bx],ax
2114  =07A9 C64701FF                    mov acb_match[bx],true
2115  =07AD E80A02        09BA          call conasn
2116  =07B0 890E6D04                    mov clierr,cx
2117  =
2118  =07B4 EB32          07E8          jmps cli_exit
2119  =
2120  =                                       ; 12. All Done
2121  =
2122  =                         cli_rmpd:     ; release PD
2123  =
2124  =07B6 88366804                    mov si,cli_pd
2125  =
2126  =                                       ; Release any memory that might still be
2127  =                                       ; associated with the PD.  This could
2128  =                                       ; happen from a CTRL C.
2129  =
2130  =07BA 837C1600741B 07D8          cmp p_mem[si],0 ! je rmpd1
2131  =07C0 1E06                            push ds ! push es
2132  =07C2 56                              push si
2133  =07C3 887416                          mov si,p_mem[si]
2134  =07C6 FF7402                          push ms_start[si]
2135  =07C9 B93200                          mov cx,f_memfree
2136  =07CC 8BD4                            mov dx,sp
2137  =07CE 8CD08ED8                        mov ax,ss ! mov ds,ax
2138  =07D2 E831FA        0206              call osif
2139  =07D5 585E                            pop ax ! pop si
2140  =07D7 071F                            pop es ! pop ds
2141  =07D9 EBDB          07B6              jmps cli_rmpd
2142  =
2143  =                                       ; Place empty PD on PUL.
2144  =
2145  =07DB 9CFA                rmpd1:  pushf ! cli
2146  =07DD 8B1E5C00                    mov bx,pul
2147  =07E1 891C89365C00                mov p_link[si],bx ! mov pul,si
2148  =07E7 90                          popf
2149  =
2150  =                                       ; Normal EXIT
2151  =
2152  =                         cli_exit:     ; close file and release CLI SYNC
2153  =07E8 891E6800                    mov bx,rlr
2154  =07EC 8A0E7404884F                mov cl,cli_dsk ! mov p_dsk[bx],cl
2155        12
2156  =07F3 8A0E7304884F                mov cl,cli_user ! mov p_user[bx],cl
2157        13
2158  =07FA 8A0E75042688                mov cl,cli_err_mode ! mov u_error_mode,cl
2159        0E1000
```

```
2160
2161  =0803 8B0E1D032689              mov cx,cli_dma_ofst ! mov u_dma_ofst,cx
2162        0F0200
2163  =080C 8B0E1F032689              mov cx,cli_dma_seg ! mov u_dma_seg,cx
2164        0E0400
2165  =0815 803E2303FF75 083C         cmp cli_chain,true ! jne clirls
2166        20           083C
2167  =081C 8B1E6B008B36                  mov bx,rlr ! mov si,cli_pd          ;inherit calling PD's
2168        6B04
2169  =0824 8B471E                        mov ax,p_parent[bx]                 ;parent if chaining
2170  =0827 89441E                        mov p_parent[si],ax
2171  =082A 803E2403FF75 083C             cmp cli_term,true ! jne clirls
2172        08           083C
2173  =0831 8167067CFD                        and p_flag[bx],not (pf_keep+pf_sys+pf_tempkeep+pf_ctlc)
2174  =0836 8B0F00                            mov cx,f_terminate              ;TERM_ACT in dispatcher
2175  =0839 E9CAF9        0206                jmp osif                        ;releases CLI_SYNC
2176  =                           clirls:
2177  =083C FF362103                  push cli_pflag
2178  =0840 FF366D04                  push cli_err
2179  =0844 E86201        09A9        call cli_unsync
2180  =0847 5A                        pop dx
2181  =0848 58250012                  pop ax ! and ax,pf_tempkeep+pf_noctls
2182  =084C 8B1E68008B4F              mov bx,rlr ! mov cx,p_flag[bx]
2183        06
2184  =0853 81E1FFED                  and cx,not pf_tempkeep+pf_noctls
2185  =0857 0BC8894F06                or cx,ax ! mov p_flag[bx],cx
2186  =085C F74706800074 0873         test p_flag[bx],pf_ctlc ! jz cli_nctl
2187        10           0873
2188  =0863 B98F0033D2                    mov cx,f_terminate ! xor dx,dx
2189  =0868 E89BF9        0206            call osif
2190  =086B 8167067FFF                    and p_flag[bx],not pf_ctlc
2191  =0870 8A2800                        mov dx,e_abort
2192  =
2193  =                                       ; setup error return if needed
2194  =                           cli_nctl:
2195  =
2196  =0873 8BCA                      mov cx,dx
2197  =0875 33DB                      xor bx,bx
2198  =0877 E303          087C        jcxz cli_gexit
2199  =0879 BBFFFF                        mov bx,0ffffh
2200  =
2201  =                           cli_gexit:
2202  =087C C3                        ret
2203  =
2204  =                           shtal:
2205  =                           ;---------
2206  =                           ; setup command tail to be parsed
2207  =                           ;       input:  AX = output of previous parsefilename
2208  =
2209  =0870 3D000007416   0898        cmp ax,0 ! je ntail
2210  =
2211  =                                   ;shift command tail to beginning
2212  =                                   ;of command buffer
```

```
2213
2214  =
2215  =0882 5020A803                    push ax ! sub ax,offset cli_cmdtail
2216  =0886 B980002BC8D1                mov cx,128 ! sub cx,ax ! shr cx,1
2217        E9
2218  =088D 5FBFA803                    pop si ! mov di,offset cli_cmdtail
2219  =0891 061E07                      push es ! push ds ! pop es
2220  =0894 F3A507                      rep movsw ! pop es
2221  =0897 C3                          ret
2222  =                           ntail:
2223  =0898 C606A80300                  mov cli_cmdtail,0
2224  =089D C3                          ret
2225  =
2226  =                           ;============================
2227  =                           ;  Various string subroutines
2228  =
2229  =089E B20DE80200    08A5 crlf:    mov dl,13 ! call prchar
2230  =08A3 B20A                        mov dl,10
2231  =                                 ;jmps prchar
2232  =
2233  =                           prchar:
2234  =08A5 268B366200                  mov si,w_conccb
2235  =08AA 883E6800                    mov di,rlr
2236  =08AE 393C7506      08B8          cmp [si],di ! jne prr
2237  =08B2 B90200E94EF9  0206              mov cx,f_conout ! jmp osif
2238  =08B8 C3                    prr:  ret
2239  =
2240  =08B9 8A168000E81C 08DC prtime: mov dl,tod_hr ! call prnum
2241        00            08DC
2242  =08C0 B23AE8E0FF    08A5          mov dl,':' ! call prchar
2243  =08C5 8A168100E810 08DC           mov dl,tod_min ! call prnum
2244        00            08DC
2245  =08CC B23AE8D4FF    08A5          mov dl,':' ! call prchar
2246  =08D1 8A168200E804 08DC           mov dl,tod_sec ! call prnum
2247        00            08DC
2248  =08D8 B220EBC9      08A5          mov dl,' ' ! jmps prchar
2249  =
2250  =08DC 52B104                prnum:  push dx ! mov cl,4
2251  =08DF D2EA80C230                    shr dl,cl ! add dl,'0'
2252  =08E4 E8BEFF        08A5            call prchar
2253  =08E7 5A80E20F                      pop dx ! and dl,0fh
2254  =08EB 80C230EBB5    08A5            add dl,'0' ! jmps prchar
2255  =
2256  =                           prfilnam:
2257  =08F0 E83600        0929            call prdisk
2258  =08F3 BA2B04                        mov dx,(offset cli_fcb)+fcb_name
2259  =08F6 E87B00        0974            call prnam
2260  =08F9 B22EE8A7FF    08A5            mov dl,'.' ! call prchar
2261  =08FE BA3304                        mov dx,(offset cli_fcb) + fcb_type
2262  =0901 E86C00        0970            call prtyp
2263  =0904 B220E89CFF    08A5            mov dl,' ' ! call prchar
2264  =0909 BB2A04                        mov bx,offset cli_fcb
2265  =090C F647080750 12 0924            test byte ptr fcb_name+7[bx],080h ! jnz pruser
```

```
2266
2267  =0912 803E730400740 0923          cmp cli_user,0 ! je pret
2268        0A            0923
2269  =0919 8B1E6800                        mov bx,rlr
2270  =091D 807F13007401 0924               cmp p_user[bx],0 ! je pruser
2271  =0923 C3                  pret:       ret
2272  =
2273  =0924 BACC09              pruser: mov dx,offset userstr
2274  =0927 EB2E          0957          jmps prcsm
2275  =
2276  =0929 8A162A04            prdisk: mov dl,cli_fcb
2277  =092D 80FA007404    0936          cmp dl,0 ! je prpddsk
2278  =0932 FECAEB07      0939          dec dl ! jmps prdsk1
2279  =                         prpddsk:
2280  =0936 8B1E6800                    mov bx,rlr
2281  =093A 8A5712                      mov dl,p_dsk[bx]
2282  =093D 80C241E862FF 08A5 prdsk1: add dl,'A' ! call prchar
2283  =0943 B23AE95DFF    08A5          mov dl,':' ! jmp prchar
2284  =
2285  =0948 BA5304              prcusp: mov dx,(offset cli_cuspqpb) + qpb_name
2286  =094B E82600BAC209 0974           call prnam ! mov dx,offset questr
2287  =0951 E80300E947FF 0957           call prcsm ! jmp crlf
2288  =
2289  =0957 26883662 00         prcsm:  mov si,u_conccb
2290  =095C 8B3E6800                    mov di,rlr
2291  =0960 393C7508      096F          cmp [si],di ! jne prr1
2292  =0964 33DB1F                      xor bx,bx ! push ds
2293  =0967 8CC88ED8                    mov ax,cs ! mov ds,ax
2294  =096B E817001F      0985          call cprnt1 ! pop ds
2295  =096F C3                  prr1:   ret
2296  =
2297  =0970 B703EB02      0976 prtyp:   mov bh,3 ! jmps prn1
2298  =0974 B708                prnam:  mov bh,8
2299  =0976 B320                prn1:   mov bl,' '
2300  =0978 26883662 00                 mov si,u_conccb
2301  =097D 8B3E6800                    mov di,rlr
2302  =0981 393C75EA      096F          cmp [si],di ! jne prr1
2303  =0985 B90E04EB25    09AF cprnt1:      mov cx,f_conprint ! jmps jos
2304  =
2305  =                         flclse:
2306  =098A B9100088A2A04               mov cx,f_fclose ! mov dx,offset cli_fcb
2307  =0990 EB0C          099E          jmps fo1
2308  =                         flopn:
2309  =0992 B90F00                      mov cx,f_fopen
2310  =0995 BE2A04                      mov si,offset cli_fcb
2311  =0998 804C0680                    or byte ptr fcb_name+5[si],080h          ;f6'=open read-only
2312  =099C 8BD6                        mov dx,si
2313  =099E 06E864F807C3 0206 fo1:      push es ! call osif ! pop es ! ret
2314  =
2315  =09A4 B91502              cli_sync:mov cx,f_sync
2316  =09A7 EB03          09AC          jmps mx1
2317  =
2318  =09A9 B91602              cli_unsync: mov cx,f_unsync
```

```
2319
2320  =09AC 8B5106              mxl:    mov bx,offset cli_spb
2321  =09AF E954F8        0206  jos:    jmp osif
2322  =
2323  =09B2 BA6704B99800        pfn:    mov dx,offset cli_pcb ! mov cx,f_parsefilename
2324  =09B8 EBF5          09AF          jmps jos
2325  =
2326  =09BA B99500BA5B04        conasn: mov cx,f_conassign ! mov dx,offset cli_acb
2327  =09C0 EBED          09AF          jmps jos
2328  =
2329  =09C2 204D73672051        questr          db      " Msg Qued",0
2330        75656400
2331  =09CC 285573657220        userstr         db      "(User 0)",0
2332        302900
2333
```

```
2334
2335  =                         eject 1 include load.sup
2336  =                       ;*********************************************************
2337  =                       ;*
2338  =                       ;*      Program Load
2339  =                       ;*
2340  =                       ;*********************************************************
2341  =                       ;
2342  =                       ;       LDTAB Entry Format:
2343  =                       ; 0           2           4           6           8
2344  =                       ; +-----+-----+-----+-----+-----+-----+-----+-----+
2345  =                       ; |   START   |    MIN    |    MAX    |    PD     |
2346  =                       ; +-----+-----+-----+-----+-----+-----+-----+-----+
2347  =                       ;
2348  =                       ; 8           10(A)       12(C)       14(E)       16(10) 17(11)
2349  =                       ; +-----+-----+-----+-----+-----+-----+-----+-----+-----+
2350  =                       ; |    ATR    |   FSTRT   |   FLEN    | TYPE|    ID     |
2351  =                       ; +-----+-----+-----+-----+-----+-----+-----+-----+-----+
2352  =                       ;
2353  =                       ;
2354  =                       ;       start           Absolute Address requested
2355  =                       ;       min             Min memory wanted
2356  =                       ;       max             Max memory wanted
2357  =                       ;       pd              PD to allocate to
2358  =                       ;       atr             Attributes, Memory Flags
2359  =                       ;       fstrt           Starting paragraph in File
2360  =                       ;       flen            # of paragraphs in file
2361  =                       ;       type            Group type
2362  =                       ;       id              Segment Address of this group
2363  =                       ;
2364  =                       ;       The Load Table contains 9 entries, One for each
2365  =                       ;       potential Group in Command File and One extra
2366  =                       ;       for Independent Memory Allocations.
2367  =
2368  =  0000                 ldt_start       equ     word ptr 0
2369  =  0002                 ldt_min         equ     word ptr ldt_start + word
2370  =  0004                 ldt_max         equ     word ptr ldt_min + word
2371  =  0006                 ldt_pd          equ     word ptr ldt_max + word
2372  =  0008                 ldt_atr         equ     word ptr ldt_pd + word
2373  =  000A                 ldt_fstrt       equ     word ptr ldt_atr + word
2374  =  000C                 ldt_flen        equ     word ptr ldt_fstrt + word
2375  =  000E                 ldt_type        equ     byte ptr ldt_flen + word
2376  =  000F                 ldt_id          equ     word ptr ldt_type + byte
2377  =  0011                 ldtlen          equ     ldt_id + word
2378  =
2379  =
2380  =                       ;==========
2381  =                       cload_ent:      ; entry point to load for a chain command
2382  =                       ;==========
2383  =                       ;       Assumes UDA is set in passed PD.
2384  =
2385  =0905 53BB0100                  push bx ! mov bx,1
2386  =0909 EB02          0900        jmps load
```

```
2387
2388  =
2389  =                         ;========
2390  =                         load_ent:       ; User entry point to load .CMD file for execution
2391  =                         ;========
2392  =                         ;       input:  DX = address of open FCB in u_wrkseg
2393  =                         ;       output: BX = segment addr of Base Page
2394  =                         ;                  = 0ffffh if error
2395  =                         ;               CX = Error Code
2396  =
2397  =09DB 29D3                       sub bx,bx
2398  =                                ;jmp load
2399  =                         ;====
2400  =                         load:           ; Intermodule entry point to load .CMD file
2401  =                         ;====
2402  =                         ;       input:  DX = addr of open FCB in u_wrkseg
2403  =                         ;               BX = addr of unused PD to initialize
2404  =                         ;                    0 - do not init PD
2405  =                         ;                    1 - chain load (PD addr on stack)
2406  =                         ;       output: BX = seg addr of Base Page
2407  =                         ;                  = 0ffffh if error
2408  =                         ;               CX = Error Code
2409  =
2410  =                                ; Get MXLoad Queue
2411  =
2412  =09DD 5253                       push dx ! push bx
2413  =09DF B989008AB001               mov cx,f_qread ! mov dx,offset mxloadqpb
2414  =09E5 E81EF8        0206        call osif
2415  =09E8 C6065E90100               mov lod_chain,false
2416  =09ED 8F06C0015E                pop lod_pd ! pop si
2417  =09F2 833EC0010175 0A02         cmp lod_pd,1 ! jne ld_cf
2418        09           0A02
2419  =09F9 8F06C001C606                  pop lod_pd ! mov lod_chain,true
2420        E901FF
2421  =
2422  =                                ; Copy FCB into lod_fcb
2423  =                                ; SI-> user FCB in u_wrkseg
2424  =
2425  =0A02 B91000BFC201        ld_cf:  mov cx,fcblen/2 ! mov di,offset lod_fcb
2426  =0A08 061E1E                      push es ! push ds ! push ds
2427  =0A0B 268E1E2E0007                mov ds,u_wrkseg ! pop es
2428  =0A11 F3A5                        rep movsw
2429  =0A13 1F07                        pop ds ! pop es
2430  =
2431  =
2432  =                                ; Read the Header.
2433  =
2434  =0A15 BBC201                      mov bx,offset lod_fcb
2435  =0A18 C6472300                    mov byte ptr fcb_r0+2[bx],0
2436  =0A1C B80000                      mov ax,0                        ;record #
2437  =0A1F BB0100                      mov bx,1                        ;# of sectors
2438  =0A22 BAF301                      mov dx,offset lod_dma           ;DMA offset
2439  =0A25 2E8B0F0600                  mov cx,sysdat                   ;DMA segment
```

```
2440
2441  =0A2A E8A405E303    0F77          call drd ! jcxz h_hdr
2442  =0A2F E97204        0EA4              jmp lod_exit
2443  =                           h_hdr:
2444  =0A32 C706E6010000                    mov lod_indma,0
2445  =
2446  =                                         ; initialize Load Disk and User
2447  =                                         ; from FCB
2448  =
2449  =0A38 8B1E6800                        mov bx,rlr
2450  =0A3C 8A4713                          mov al,p_user[bx]
2451  =0A3F A2EA01                          mov lod_user,al
2452  =0A42 8A4712                          mov al,p_dsk[bx]        ;default disk of calling PD
2453  =0A45 A2E801                          mov lod_disk,al         ;1-15 -> A-P
2454  =0A48 BBC201                          mov bx,offset lod_fcb
2455  =0A4B 8A07                            mov al,fcb_dr[bx]
2456  =0A4D A2EC01                          mov lod_fifty,al        ;base page address 50H, 0=default
2457  =0A50 84C07405      0A59              test al,al ! jz use_ddsk
2458  =0A54 FEC8                                dec al              ;1-15 -> A-P
2459  =0A56 A2E801                              mov lod_disk,al
2460  =                           use_ddsk:
2461  =0A59 F64708807405  0A64              test byte ptr fcb_name+7[bx],080h ! jz use_dusr
2462  =0A5F C606EA0100                          mov lod_user,0
2463  =                           use_dusr:
2464  =
2465  =                                         ; Initialize ldtab
2466  =
2467  =                                             ; Zero ldtab
2468  =0A64 B955002BC0                      mov cx,ldtabsiz/2 ! sub ax,ax
2469  =0A69 BF7302                          mov di,offset ldtab
2470  =0A6C 062E8E060600                    push es ! mov es,sysdat
2471  =0A72 F3AB07                          rep stos ax ! pop es
2472  =
2473  =                                             ; 1st ldtab entry is UDA and LSTK
2474  =                                             ; if a PD was specified...
2475  =
2476  =0A75 C7066E010000                    mov lod_nldt,0
2477  =0A7B BE7302                          mov si,offset ldtab
2478  =0A7E 833EC0010075  0A88              cmp lod_pd,0 ! jne form_uda
2479        03            0A88
2480  =0A83 E93E00        0AC6                  jmp gc_ifo
2481  =0A88 C744021600            form_uda:     mov ldt_min[si],(lstklen+ulen)/16   ;min=max=UDA+STK paragraphs
2482  =0A8D C744041600                          mov ldt_max[si],(lstklen+ulen)/16
2483  =0A92 A1C001894406                        mov ax,lod_pd ! mov ldt_pd[si],ax
2484  =0A98 C744080100                          mov ldt_atr[si],mf_load
2485  =0A9D 83C611                              add si,ldtlen
2486  =0AA0 FF06E01                             inc lod_nldt
2487  =0AA4 803EE901FF75  0AC6                  cmp lod_chain,true ! jne gc_ifo
2488        1B            0AC6
2489  =
2490  =                                             ;We are CHAINING.  Free all memory
2491  =                                             ; except UDA area and LDSTK.  This will keep
2492  =                                             ; the first partition for the chain
```

```
2493
2494  =                                       ; as well as not crash the system.
2495  =
2496  =0AAB 56                                push si
2497  =0AAC 893A00E854F7 0206                 mov cx,f_freeall ! call osif
2498  =
2499  =                                       ;transfer memory to new pd
2500  =
2501  =0AB2 8B1E6800                          mov bx,rlr
2502  =0AB6 8B4716C74716                      mov ax,p_mem[bx] ! mov p_mem[bx],0
2503        0000
2504  =0ABE 891EC0018947                      mov bx,lod_pd ! mov p_mem[bx],ax
2505        16
2506  =0AC5 5E                                pop si
2507  =
2508  =                         gc_ifo:
2509  =                                       ;go through CMD header and init
2510  =                                       ;a ldtab entry per header entry.
2511  =                                       ;alloc abs mem
2512  =
2513  =0AC6 BBF301                      mov bx,offset lod_dma
2514  =0AC9 8A477F                      mov al,ch_lbyte[bx]     ; save fixup flag
2515  =0ACC A2EE01                      mov lod_lbyte,al
2516  =0ACF 8B477D                      mov ax,ch_fixrec[bx]    ; save record # of fixups, if any
2517  =0AD2 A3EF01                      mov lod_fixrec,ax
2518  =0AD5 A3F101                      mov lod_fixrec1,ax
2519  =
2520  =0AD8 B90800                      mov cx,ch_entmax        ; BX = offset LOD_DMA
2521  =0ADB BA0800                      mov dx,8                ; DX = position in file
2522  =0ADE 803F007503    0AE6 ch_more:cmp ch_form[bx],0 ! jne ch_doit
2523  =0AE3 E98100        0B67         jmp ch_next
2524  =0AE6 8A0788440E         ch_doit:    mov al,ch_form[bx] ! mov ldt_type[si],al     ;type of seg
2525  =0AEB 8B4701894400                   mov ax,ch_length[bx] ! mov ldt_flen[si],ax  ;length
2526  =0AF1 89540A03D0                     mov ldt_fstrt[si],dx ! add dx,ax            ;pos in file
2527  =0AF6 8B4703894402                   mov ax,ch_base[bx] ! mov ldt_start[si],ax    ;abs seg
2528  =0AFB 8B4705894402                   mov ax,ch_min[bx] ! mov ldt_min[si],ax       ;min needed
2529  =0B01 8B4707                         mov ax,ch_max[bx]
2530  =0B04 3D0000750B    0B0C             cmp ax,0 ! jne setmax
2531  =0B09 8B4705                             mov ax,ch_min[bx]
2532  =0B0C 894404             setmax:     mov ldt_max[si],ax                  ;max wanted
2533  =0B0F A1C001894406                   mov ax,lod_pd ! mov ldt_pd[si],ax    ;pd to alloc to
2534  =0B15 3D000074 07   0B21             cmp ax,0 ! je not_load
2535  =0B1A B80100                             mov ax,mf_load
2536  =0B1D EB02          0B21                 jmps not_load
2537  =0B1F EB5D          0ADE skipjmp:jmps ch_more
2538  =
2539  =                         if mpm
2540  =
2541  =                         not_load:   cmp ch_form[bx],1 ! jne try_sh
2542  =                                     add ax,mf_code ! jmps s_atr
2543  =                         try_sh:     cmp ch_form[bx],9 ! jne s_atr
2544  =                                     add ax,mf_code+mf_share
2545  =                         s_atr:      mov ldt_atr[si],ax                  ;memory flags
```

```
2546
2547  =
2548  =                                       ;if abs, allocate memory
2549  =                                   cmp ldt_start[si],0 ! je ch_nabs    ;see if abs mem
2550  =                                       jmps ch_al
2551  =                           ch_nabs:    cmp ldt_type[si],9
2552  =                                   jne ch_nxt          ;see if shared code
2553  =                                     push cx
2554  =                                     push bx ! push dx ;save load DMA and position in file
2555  =                                     call get_sn
2556  =                                     pop dx ! pop bx
2557  =                                     cmp cx,0 ! pop cx
2558  =                                     je ch_nxt
2559  =                                       jmp ld_out
2560  =                           ch_al:      push bx ! push dx ! push cx ! push si
2561  =                                       mov cx,f_malloc ! mov dx,si
2562  =                                       call osif ! pop si
2563  =                                       mov ax,ldt_start[si] ! mov ldt_id[si],ax
2564  =                                       cmp cx,0 ! pop cx ! pop dx ! pop bx
2565  =                                       je ch_nxt
2566  =                                           ;couldn't find memory
2567  =                                         mov bx,0ffffh ! mov cx,e_no_memory
2568  =                                         jmp ld_out
2569  =
2570  =                           endif
2571  =
2572  =                           if ccpm
2573  =
2574  =0821 803F097507    082D not_load:   cmp ch_form[bx],9 ! jne try_code
2575  =0826 C60701                        mov ch_form[bx],1
2576  =0829 C6440E01                      mov ldt_type[si],1
2577  =082D 803F017503    0835 try_code:  cmp ch_form[bx],1 ! jne s_atr
2578  =0832 050400                        add ax,mf_code
2579  =0835 894408                s_atr:  mov ldt_atr[si],ax                  ;memory flags
2580  =
2581  =                                       ;if abs, allocate memory
2582  =0838 833C007423    0860           cmp ldt_start[si],0 ! je ch_nxt     ;see if abs mem
2583  =083D 53525156              ch_al:      push bx ! push dx ! push cx ! push si
2584  =0841 B980008BD6                        mov cx,f_malloc ! mov dx,si
2585  =0846 E83DF65E      0206                call osif ! pop si
2586  =084A 8B0489440F                        mov ax,ldt_start[si] ! mov ldt_id[si],ax
2587  =084F 833F00595A5B                      cmp cx,0 ! pop cx ! pop dx ! pop bx
2588  =0855 7409          0860                je ch_nxt
2589  =                                           ;couldn't find memory
2590  =0857 B8FFFF8B0300                      mov bx,0ffffh ! mov cx,e_no_memory
2591  =085D E97500        0BD6                jmp ld_out
2592  =
2593  =                           endif
2594  =
2595  =0860 83C611                ch_nxt: add si,ldtlen
2596  =0863 FF058E01                      inc lod_nldt
2597  =0867 83C309                ch_next:add bx,chlen
2598  =086A E2B3          081F            loop skipjmp
```

```
2599
2600  =
2601  =                                     ; alloc all other memory
2602  =                                     ; SI -> mpb for non_abs mem req.
2603  =                                         ;add all parts together for a single malloc
2604  =0A6C B37302                      mov bx,offset ldtab
2605  =0A6F 8B0E8E01                    mov cx,lod_nldt
2606  =0A73 C606E80100                  mov lod_nrels,0
2607  =0A78 837F02007427 0AA1 lt_more:cmp ldt_min[bx],0 ! je lt_next
2608  =0A7E 833F007528   0AA8           cmp ldt_start[bx],0 ! jne lt_next
2609  =0A83 8B4702                          mov ax,ldt_min[bx]
2610  =0A86 8BD0                            mov dx,ax
2611  =0A88 014402                          add ldt_min[si],ax
2612  =0A8B 3B54027605   0A95               cmp dx,ldt_min[si] ! jbe lt_m   ; check for ovrflw
2613  =0A90 C74402FFFF                          mov ldt_min[si],0ffffh
2614  =0A95 8B4704              lt_m:       mov ax,ldt_max[bx]
2615  =0A98 8BD0                            mov dx,ax
2616  =0A9A 014404                          add ldt_max[si],ax
2617  =0A9D 3B54047605   0AA7               cmp dx,ldt_max[si] ! jbe lt_m1  ; check for ovrflw
2618  =0AA2 C74404FFFF                          mov ldt_max[si],0ffffh
2619  =0AA7 FE06E801            lt_m1:      inc lod_nrels
2620  =0AAB 83C311E2C8   0A78 lt_next:add bx,ldtlen ! loop lt_more
2621  =
2622  =                                         ;malloc
2623  =0AB0 833EC0010074 0ABC           cmp lod_pd,0 ! je noloadf
2624        05           0ABC
2625  =0AB7 C744080100                      mov ldt_atr[si],mf_load
2626  =0ABC A1C001894406        noloadf:mov ax,lod_pd ! mov ldt_pd[si],ax
2627  =0AC2 56833D6898000               push si ! mov dx,si ! mov cx,f_malloc
2628  =0AC8 E838F65E     0206           call osif ! pop si
2629  =0ACC 8B0489440F                  mov ax,ldt_start[si] ! mov ldt_id[si],ax
2630  =0AD1 83F8FF7533   0C09           cmp bx,0ffffh ! jne lt_sprd
2631  =
2632  =                                         ;Not Enough Memory - release any
2633  =                                         ; memory already allocated
2634  =0AD6 51                  ld_out:     push cx
2635  =0AD7 BB7302                          mov bx,offset ldtab
2636  =0ADA 8B0E8E0141                      mov cx,lod_nldt ! inc cx
2637  =0ADF 837F0F007418 0AFD lg_more:      cmp ldt_id[bx],0 ! je lg_next
2638  =0AE5 51531E                              push cx ! push bx ! push ds
2639  =                                         ;push MFPB on stack
2640  =0AE8 FF7706                          push ldt_pd[bx]
2641  =0AEB FF770F                          push ldt_id[bx]
2642  =0AEE 8BD4161F                        mov dx,sp ! push ss ! pop ds
2643  =0AF2 B98200                          mov cx,f_memfree
2644  =0AF5 E80EF6       0206               call osif
2645  =0AF8 5959                            pop cx ! pop cx
2646  =
2647  =0AFA 1F5B59                              pop ds ! pop bx ! pop cx
2648  =0AFD 83C311E2DD   0ADF lg_next:      add bx,ldtlen ! loop lg_more
2649  =0C02 BBFFFF59E99B 0EA4               mov bx,0ffffh ! pop cx ! jmp lod_exit
2650        02           0EA4
2651  =
```

```
2652
2653  =                         lt_spru:
2654  =                                                 ;spread the memory allocated
2655  =                                                 ;amongst the nrels
2656  =                                                 ;1st give everyone the minimum
2657  =0C09 BB7302                        mov bx,offset ldtab
2658  =0C0C 8B0E6E01                      mov cx,lod_nldt
2659  =0C10 833F007515    0C2A ls_more:cmp ldt_start[bx],0 ! jne ls_next
2660  =0C15 8B4702                        mov ax,ldt_min[bx]
2661  =0C18 294402                        sub ldt_min[si],ax
2662  =0C1B 3B4704750A    0C2A            cmp ax,ldt_max[bx] ! jne ls_next
2663  =0C20 8B4489C7                          mov dx,ldt_start[si] ! mov ldt_start[bx],dx
2664  =0C24 0104                              add ldt_start[si],ax
2665  =0C26 FE0E6801                          dec lod_nrels
2666  =0C2A 83C311E2E1    0C10 ls_next:add bx,ldtlen ! loop ls_more
2667  =
2668  =                                                 ;spread whats left amongst those that need more
2669  =0C2F BB7302                        mov bx,offset ldtab
2670  =0C32 8B0E6E01                      mov cx,lod_nldt
2671  =0C36 833F007539    0C74 lsl_mre:cmp ldt_start[bx],0 ! jne lsl_nxt
2672  =0C3B 8B048907                      mov ax,ldt_start[si] ! mov ldt_start[bx],ax
2673  =0C3F 8B4402                        mov ax,ldt_min[si]
2674  =0C42 3B000074 24   0C6B            cmp ax,0 ! je adj_start
2675  =0C47 51 23C9                           push cx ! sub cx,cx
2676  =0C4A 8B D1 8B 0E E8 01                 mov dx,cx ! mov cl,lod_nrels
2677  =0C50 F7F1                              div cx ! pop cx
2678  =0C53 83FA007401    0C59                cmp dx,0 ! je evendiv
2679  =0C58 40                                    inc ax
2680  =0C59 8B5704 2B5702         evendiv:        mov dx,ldt_max[bx] ! sub dx,ldt_min[bx]
2681  =0C5F 3BC2 7602      0C65                   cmp ax,dx ! jbe nottoomuch
2682  =0C63 8BC2                                  mov ax,dx
2683  =0C65 014702 294402         nottoomuch:     add ldt_min[bx],ax ! sub ldt_min[si],ax
2684  =0C6B 8B4702 0104           adj_start:  mov ax,ldt_min[bx] ! add ldt_start[si],ax
2685  =0C70 FE0E6801                          dec lod_nrels
2686  =0C74 83C311E2BD    0C36 lsl_nxt:add bx,ldtlen ! loop lsl_mre
2687  =
2688  =                                           ; fill memory from file
2689  =
2690  =0C79 BE7302                        mov si,offset ldtab
2691  =0C7C 8B0E6E01                      mov cx,lod_nldt
2692  =0C80 837C0C007413  0C99 lf_mre: cmp ldt_flen[si],0 ! je lf_nxt
2693  =0C86 51FF3456                          push cx ! push ldt_start[si] ! push si
2694  =0C8A E82502        0EB2                call lod_group
2695  =0C8D 5E8F04                            pop si ! pop ldt_start[si]
2696  =0C90 83F90059                          cmp cx,0 ! pop cx
2697  =0C94 7403          0C99                je lf_nxt
2698  =
2699  =                         if ccpm
2700  =
2701  =0C96 E930FF        0BD6                   jmp ld_out
2702  =                         endif
2703  =
2704  =                         if mpm
```

```
2705
2706  =                                         ; error in lod_group
2707  =                                         ; if loading shared code we also
2708  =                                         ; have to release Shared Code from SCL.
2709  =                                         ; it will be the top item in the list.
2710  =
2711  =                                 cmp ldt_atr[si],mf_load+mf_code+mf_share
2712  =                                 je rm_sh
2713  =                                     jmp ld_out
2714  =                           rm_sh:  push cx                         ; remember Err Code
2715  =
2716  =                                         ; Remove PD from SCL
2717  =                                 pushf ! cli
2718  =                                 mov bx,scl
2719  =                                 mov ax,p_thread[bx]
2720  =                                 mov scl,ax
2721  =                                 popf
2722  =
2723  =                                         ; Release the memory
2724  =
2725  =                                 push ds ! push bx
2726  =                                 mov bx,p_mem[bx] ! push ms_start[bx]
2727  =                                 mov ax,ss ! mov ds,ax
2728  =                                 mov dx,sp ! mov cx,f_memfree
2729  =                                 call osif
2730  =                                 pop ax ! pop bx ! pop ds
2731  =
2732  =                                         ; Place PD on PUL
2733  =
2734  =                                 pushf ! cli
2735  =                                 mov ax,pul
2736  =                                 mov pul,bx
2737  =                                 mov p_link[bx],ax
2738  =                                 popf
2739  =
2740  =                                 pop cx
2741  =                                 jmp ld_out
2742  =                           endif
2743  =
2744  =0C99 83C611E2E2    0C80  lf_nxt: add si,ldtlen ! loop lf_mre
2745  =
2746  =
2747  =                           ;        Check for fixup records and do fixups
2748  =
2749  =0C9E F606EE0180                  test lod_lbyte,80h        ; hi bit of last byte in cmd header
2750  =0CA3 7475          0D1A          jz init_base              ; is set if fixups
2751  =                           fx_read:
2752  =0CA5 A1EF01                      mov ax,lod_fixrec         ; get record # of fixups in file
2753  =0CA8 BB0100                      mov bx,1                  ; read one record
2754  =0CAB 8CD9                        mov cx,ds
2755  =0CAD BAF301                      mov dx, offset lod_dma
2756  =0CB0 E8C402        0F77          call drd                  ; do random read
```

```
2758
2759  =0CB3 E30F          0CC4          jcxz fx_read_ok           ; 0=ok,CL=0ff if EOF
2760  =0CB5 FEC1                          inc cl                  ; EOF ?
2761  =0CB7 7556          0D14            jnz fx_err              ; some read error, not EOF
2762  =0CB9 A1F101                          mov ax,lod_fixrecl    ; make sure one fixup
2763  =0CBC 3B06EF01                        cmp ax,lod_fixrec     ; record was read since cmd header
2764  =0CC0 7452          0D14              je fx_err             ; said we needed them
2765  =0CC2 EB55          0D1A                jmps init_base
2766  =
2767  =                             fx_read_ok:                   ; go look for fixup records
2768  =0CC4 BBF301                          mov bx,offset lod_dma ; BX-> at fixup records
2769  =                             fx_chk:
2770  =0CC7 8A07                            mov al,fix_grp[bx]    ; any more fixups?
2771  =0CC9 84C0                            test al,al
2772  =0CCB 744D          0D1A              jz init_base
2773  =0CCD 240F                            and al,0fh            ; low nibble is target group
2774  =0CCF E82D00        0CFF              call tblsrch          ; find target group in load table
2775  =0CD2 8B15                            mov dx,ldt_start[di]  ; DX = target segment, this is
2776  =                                                           ; what we add to the load image
2777  =0CD4 8A07                            mov al,fix_grp[bx]    ; location group is high nibble
2778  =0CD6 B104                            mov cl,4              ; put in low nibble of AL
2779  =0CD8 D2E8                            shr al,cl
2780  =0CDA E82200        0CFF              call tblsrch          ; DI = location load table entry
2781  =0CDD 8B05                            mov ax,ldt_start[di]  ; AX = base segment of location
2782  =0CDF 034701                          add ax,fix_para[bx]   ; add paragraph offset
2783  =
2784  =0CE2 06                              push es               ; absolute paragraph in memory
2785  =0CE3 8EC0                            mov es,ax
2786  =0CE5 33C0                            xor ax,ax
2787  =0CE7 8A4703                          mov al,fix_offs[bx]   ; get offset of fixup location
2788  =0CEA 8BF8                            mov di,ax
2789  =0CEC 260115                          add es:[di],dx        ; make the fixup (finally)
2790  =0CEF 07                              pop es
2791  =
2792  =0CF0 83C304                          add bx,fixlen         ; do next fixup
2793  =0CF3 81FB7302                        cmp bx,offset lod_dma + dskrecl ; end of this record ?
2794  =0CF7 75CE          0CC7              jne fx_chk
2795  =0CF9 FF06EF01                          inc lod_fixrec      ; read another record
2796  =0CFD EBA5          0CA5                jmps fx_read        ; of fixups
2797  =
2798  =                             tblsrch:
2799  =                             ;-------
2800  =                             ;       Search for group in load table
2801  =                             ;       entry:  AL = group # to match on
2802  =                             ;       exit:   DI = load group entry that matches or
2803  =                             ;                    pop return and exit loader
2804  =                             ;                    BX,DX preserved
2805  =
2806  =0CFF 8B0EEF01                        mov cx,lod_nldt       ; # entries in table
2807  =0D03 BF7302                          mov di,offset ldtab
2808  =                             srchloop:
2809  =0D06 3A450E                          cmp al,ldt_type[di]
2810  =0D09 7408          0D13              je srchdn             ; found group's entry
```

```
2811
2812  =0DC6 83C711                      add di,ldtlen
2813  =0DC9 E2F5            0DC0        loop srchloop
2814  =0DCB 58                          pop ax
2815  =0DCC EB01            0DCF        jmps fx_err
2816  =                           srchdn:
2817  =0DCE C3                          ret
2818  =
2819  =                           fx_err:
2820  =0DCF B92900                      mov cx,e_fixuprec
2821  =0DD2 E9BCFE          0BD6        jmp ld_out
2822  =
2823  =
2824  =                                 ; init Base Page
2825  =                                 ;       1st Data Group has Base Page
2826  =                                 ;       if none then first nonshared
2827  =                                 ;       Code Group (8080 model)
2828  =                           init_base:
2829  =0DD5 C606ED0100                  mov lod_8080,0
2830  =0DDA BE7302                      mov si,offset ldtab
2831  =0DDD 8B0E0E01                    mov cx,lod_nldt
2832  =0DE1 807C0E027422 0E09 lb_more:cmp ldt_type[si],2 ! je lb_fnd
2833  =0DE7 83C611E2F5    0DE1          add si,ldtlen ! loop lb_more
2834  =0DEC BE7302                        mov si,offset ldtab
2835  =0DEF 8B0E0E01                      mov cx,lod_nldt
2836  =0DF3 807C0E017408 0DFE lbc_mre:    cmp ldt_type[si],1 ! je lb_fnd80
2837  =0DF8 83C611E2F5    0DF3              add si,ldtlen ! loop lbc_mre
2838  =0DFD B92100E98DFE  0BD6                mov cx,e_no_cseg ! jmp ld_out
2839  =0E03 C606ED0101            lb_fnd80:   mov lod_8080,1
2840  =                           lb_fnd:
2841  =0E08 068E04                      push es ! mov es,ldt_start[si]
2842  =0E0B 8C063C01                    mov lod_basep,es
2843  =0E0F 2BC03BF8                    sub ax,ax ! mov di,ax
2844  =0E13 B92000F3AB                  mov cx,05bh/2 ! rep stos ax
2845  =0E18 A0ED0126A205                mov al,lod_8080 ! mov es:.5,al
2846        00
2847  =0E1E BE7302                      mov si,offset ldtab
2848  =0E21 8B0E0E01                    mov cx,lod_nldt
2849  =0E25 807C0E007447 0E6E lbb_mre:cmp ldt_type[si],0 ! je lbb_nxt
2850  =0E2B B8050D8A5C0E                mov ax,6 ! mov bl,ldt_type[si]
2851  =0E31 80FB097502    0E38          cmp bl,9 ! jne lbb_nsh
2852  =0E36 B301                          mov bl,1
2853  =0E38 FEC3F6E38BD8          lbb_nsh:    dec bl ! mul bl ! mov bx,ax
2854  =
2855  =                                   ;calculate last byte (3 byte value)
2856  =                                   ;  =(paragraphs*16)-1
2857  =0E3E 518B5402                    push cx ! mov dx,ldt_min[si]
2858  =0E42 52B104D3E2                  push dx ! mov cl,4 ! shl dx,cl
2859  =0E47 524A268917                  push dx ! dec dx ! mov es:[bx],dx
2860  =0E4C 595A51                      pop cx ! pop dx ! push cx
2861  =0E4F 80CD3EA59                   mov cl,12 ! shr dx,cl ! pop cx
2862  =0E54 83F900750E    0E67          cmp cx,0 ! jne lbb_nzer
2863  =0E59 83FA007404    0E62              cmp dx,0 ! je lbb_zer
```

```
2864
2865  =0DA5 FECAE805      0DAE                    dec dl ! jmps lbb_nzer
2866  =0DA9 26C7070000        lbb_zer:        mov es:word ptr [bx],0
2867  =0DAE 26895702S9        lbb_nzer:   mov es:2[bx],dl ! pop cx
2868  =
2869  =                                       ;put starting paragraph in place
2870  =0DB3 8B0426894703                    mov ax,ldt_start[si] ! mov es:3[bx],ax
2871  =0DB9 83C611E2AE    0D6C lbb_nxt:add si,ldtlen ! loop lbb_mre
2872  =
2873  =                                       ;if 8080 model, copy CS info into DS info
2874  =
2875  =0DBE 803EE0010175 0DD4          cmp lod_8080,1 ! jne lnot80
2876        0F            0DD4
2877  =0DC3 1E061FBE0000                    push ds ! push es ! pop ds ! mov si,0
2878  =0DC8 BF0600B90300                    mov di,6 ! mov cx,3
2879  =0DD1 F3A51F                          rep movsw ! pop ds
2880  =                             lnot80:
2881  =0DD4 A0EC01                  mov al,lod_fifty        ;initialize base page load disk
2882  =0DD7 26A25000                mov es:.50H,al
2883  =0DDB 07                      pop es
2884  =
2885  =                                       ; init PD ,UDA and LOSTK
2886  =
2887  =0DDC 8B1EE0C01               mov bx,lod_basep
2888  =0DE0 833EC0010075 0DEA       cmp lod_pd,0 ! jne lip_1
2889        03            0DEA
2890  =0DE7 E9BA00        0EA4          jmp lod_exit
2891  =0DEA BE7302              lip_1:  mov si,offset ldtab
2892  =0DED 8B1EC001                mov bx,lod_pd
2893  =0DF1 8B04                    mov ax,ldt_start[si]
2894  =0DF3 894710                  mov p_uda[bx],ax
2895  =                                       ; remember where lstk,uda are
2896  =0DF6 A3B801                  mov lod_uda,ax
2897  =0DF9 068EC0                  push es ! mov es,ax
2898  =0DFC 051000A3BA01            add ax,(ulen/16) ! mov lod_lstk,ax
2899  =                                       ; initialize UDA,LDSTK with zeros
2900  =0E02 33FF8BC7                xor di,di ! mov ax,di
2901  =0E06 B9B000                  mov cx,(ulen + lstklen)/2
2902  =0E09 F3AB07                  rep stos ax ! pop es
2903  =
2904  =                                       ; setup p_uda for creat_proc
2905  =
2906  =0E0C 2EA10600                mov ax,sysdat
2907  =0E10 294710                  sub p_uda[bx],ax
2908  =0E13 C6470400                mov p_stat[bx],ps_run
2909  =0E17 C64705C8                mov p_prior[bx],200
2910  =
2911  =                                       ; init load disk/user
2912  =
2913  =0E1B A0E401884715            mov al,lod_user ! mov p_luser[bx],al
2914  =0E21 A0E301884714            mov al,lod_disk ! mov p_ldsk[bx],al
2915  =
2916  =                                           ; init UDA
```

```
2917
2918  =
2919  =0E27 068E06B401              push es ! mov es,lod_uda
2920  =0E2C C6C706020080            mov u_dma_ofst,(offset bpg_dma)
2921        00
2922  =0E33 8B1E3BC01               mov bx,lod_basep
2923  =0E37 26891E0400              mov u_dma_seg,bx
2924  =0E3C A13A01                  mov ax,lod_lstk
2925  =0E3F 1E8ED8                  push ds ! mov ds,bx
2926  =0E42 26891E5200              mov u_initds,bx
2927  =0E47 26891E5400              mov u_inites,bx
2928  =0E4C 26A35600                mov u_initss,ax
2929  =0E50 A10300                  mov ax,bpg_cseg
2930  =0E53 3D00007508    0E60      cmp ax,0 ! jne h_cs
2931  =0E58 1F07B92100              pop ds ! pop es ! mov cx,e_no_cseg
2932  =0E5D E975FD        0BD6      jmp ld_out
2933  =0E60 26A35000          h_cs: mov u_initcs,ax
2934  =0E64 A10F00                  mov ax,bpg_eseg
2935  =0E67 3D00007404    0E70      cmp ax,0 ! je noes
2936  =0E6C 26A35400                mov u_inites,ax
2937  =0E70 26C706340056        noes: mov u_stack_sp,(offset ls_sp)
2938        00
2939  =0E77 2BD2A00500              sub dx,dx ! mov al,bpg_8080
2940  =0E7C 3C097403      0E83      cmp al,0 ! je n80m
2941  =0E80 BA0001                  mov dx,0100h
2942  =                         n80m:
2943  =0E83 263E1E5600              mov ds,u_initss
2944  =0E88 89165600                mov ls_offset,dx                ;set up initial stack
2945  =0E8C C7065A000002            mov ls_flags,0200h              ;for a RETF from user
2946  =0E92 C7065C002E02            mov ls_roffset,offset user_retf ;process, see SUPIF.SUP
2947  =0E98 8C0E5E00                mov ls_rcseg,cs                 ;module for USER_RETF:
2948  =0E9C 1F07                    pop ds ! pop es                 ;code
2949  =0E9E 8B1E3BC01               mov bx,lod_basep
2950  =0EA2 2BC9                    sub cx,cx
2951  =                         lod_exit:
2952  =0EA4 5153                    push cx ! push bx
2953  =0EA6 B9890DBAB001            mov cx,f_qwrite ! mov dx,offset mxloadqpb
2954  =0EAC E857F35B59C3 0206       call osif ! pop bx ! pop cx ! ret
2955  =
2956  =
2957  =                         lod_group:              ;load a group described in ldtab
2958  =                         ;----------
2959  =                         ;       input:  SI = addr of ldtab entry
2960  =                         ;       output: AX = Error Code
2961  =
2962  =                                 ; see if first part already in DMA
2963  =0EB2 8BDE                    mov bx,si
2964  =0EB4 A1E601                  mov ax,lod_indma                ;starting paragraph in dma
2965  =0EB7 8B4F0A                  mov cx,ldt_fstrt[bx]
2966  =                                 ;AX = starting paragraph in local DMA
2967  =                                 ;CX = starting paragraph to transfer
2968  =                                 ;BX -> ldtab entry
2969  =0EBA 3BC87237      0EF5      cmp cx,ax ! jb rd_first
```

```
2970
2971  =                                     ;starts at or after the pp. in dma
2972  =0EBE 2BC8                        sub cx,ax
2973  =0EC0 83F9087330    0EF5          cmp cx,8 ! jae rd_first
2974  =                                     ;starts in the dma
2975  =0EC5 BA03002BD1                  mov dx,8 ! sub dx,cx
2976  =                                     ;CX = # of pp. to skip
2977  =                                     ;DX = length of remaining buffer
2978  =0ECA 3B570C7603    0ED2          cmp dx,ldt_flen[bx] ! jbe xfer
2979  =0ECF 8B570C                          mov dx,ldt_flen[bx]
2980  =0ED2 BEF301              xfer:       mov si,offset lod_dma
2981  =0ED5 8BC1B104D3E0                    mov ax,cx ! mov cl,4 ! shl ax,cl
2982  =0EDB 03F0                            add si,ax
2983  =                                     ;SI -> beginning of transfer area
2984  =                                     ;       in lod_dma
2985  =0EDD 8BC2B103D3E0                    mov ax,dx ! mov cl,3 ! shl ax,cl
2986  =0EE3 8BC8                            mov cx,ax
2987  =                                     ;CX = number of words to transfer
2988  =0EE5 33FF                            xor di,di
2989  =0EE7 068E07                          push es ! mov es,ldt_start[bx]
2990  =0EEA F3A507                          rep movsw ! pop es
2991  =0EED 0117                            add ldt_start[bx],dx
2992  =0EEF 2957UC01570A                    sub ldt_flen[bx],dx ! add ldt_fstrt[bx],dx
2993  =                         rd_first:
2994  =0EF5 837F0C007503 0EFE         cmp ldt_flen[bx],0 ! jne rd_1st
2995  =0EFA 2BC9C3                        sub cx,cx ! ret
2996  =                         rd_1st:
2997  =0EFE F7470A070075 0F0B         test ldt_fstrt[bx],07h ! jnz rd_indma
2998        06           0F0B
2999  =0F05 837F0C087333 0F3E         cmp ldt_flen[bx],8 ! jae xf_d
3000  =                         rd_indma:
3001  =0F0B 53                            push bx
3002  =0F0C 8B470A                        mov ax,ldt_fstrt[bx]
3003  =0F0F D1E8D1E8D1E8                  shr ax,1 ! shr ax,1 ! shr ax,1      ; Record #
3004  =0F15 BB0100                        mov bx,1                            ; read 1 record
3005  =0F18 BAF301                        mov dx,offset lod_dma               ; DMA offset
3006  =0F1B 2E8B0E0600                    mov cx,sysdat                       ; DMA segment
3007  =0F20 E85400        0F77            call drd
3008  =0F23 5B                            pop bx
3009  =0F24 E304          0F30            jcxz rd_agn
3010  =0F26 81F9FF007503 0F2F                cmp cx,0ffh ! jne rd_r3
3011  =0F2C B90000                            mov cx,0
3012  =0F2F C3                  rd_r3:        ret
3013  =0F30 8B470A              rd_agn:   mov ax,ldt_fstrt[bx]
3014  =0F33 25F8FF                        and ax,0fff8h                       ; note starting paragraph
3015  =0F36 A3E601                        mov lod_indma,ax                    ; in DMA
3016  =0F39 8BF3E974FF    0EB2            mov si,bx ! jmp lod_group
3017  =
3018  =                                     ; We are at a Sector Boundary with at least
3019  =                                     ; a sector to place into the user area
3020  =                         xf_d:
3021  =0F3C 53                        push bx
3022  =0F3F 2BD2                      sub dx,dx                             ; DMA offset
```

```
3023
3024  =0F41 8B0F                    mov cx,ldt_start[bx]                    ; DMA segment
3025  =0F43 8B470A                  mov ax,ldt_fstrt[bx]
3026  =0F46 D1E8D1E8D1E8            shr ax,1 ! shr ax,1 ! shr ax,1          ; Record #
3027  =0F4C 8B5F0C                  mov bx,ldt_flen[bx]
3028  =0F4F D1EBD1EBD1EB            shr bx,1 ! shr bx,1 ! shr bx,1          ; # of records
3029  =0F55 53                      push bx
3030  =0F56 E81E00        0F77      call drd
3031  =0F59 5B58                    pop ax ! pop bx
3032  =0F5B E309          0F66      jcxz xfer_n
3033  =0F5D 81F9FF007502  0F65          cmp cx,0ffh ! jne rd_r4
3034  =0F63 33C9                        xor cx,cx
3035  =0F65 C3                rd_r4:    ret
3036  =0F66 D1E0D1E0D1E0      xfer_n: shl ax,1 ! shl ax,1 ! shl ax,1
3037  =0F6C 0107                      add ldt_start[bx],ax
3038  =0F6E 01470A                    add ldt_fstrt[bx],ax
3039  =0F71 29470C                    sub ldt_flen[bx],ax
3040  =0F74 E97EFF        0EF5        jmp rd_first
3041  =
3042  =                       drd:
3043  =                       ;       input:  AX = Record Number
3044  =                       ;               BX = Number of Sectors
3045  =                       ;               CX = dma segment
3046  =                       ;               DX = dma offset
3047  =                       ;       output: CX = 0 if okay
3048  =                       ;               CX = 0FFH if End of File
3049  =                       ;                    else Error Code
3050  =
3051  =0F77 2689160200                mov u_dma_ofst,dx
3052  =0F7C 268C0E0400                mov u_dma_seg,cx
3053  =
3054  =                                   ; read BX sectors starting at Record AX
3055  =                       drd_lp:
3056  =0F81 53                        push bx                         ;old # sectors
3057  =0F82 50                        push ax                         ;old record #
3058  =0F83 81FB80007603  0F8C        cmp bx,128 ! jbe drd_r
3059  =0F89 BB8000                        mov bx,128
3060  =
3061  =                                   ; Max Mulit-sector count is 128
3062  =
3063  =0F8C 268A0E1100        drd_r:  mov cl,u_mult_cnt
3064  =0F91 51                        push cx
3065  =0F92 26881E1100                mov u_mult_cnt,bl
3066  =0F97 53                        push bx
3067  =0F98 BEC201                    mov si,offset lod_fcb
3068  =0F9B 894421                    mov fcb_r0[si],ax
3069  =0F9E B9210088D6                mov cx,f_freadrdm ! mov dx,si
3070  =0FA3 06E85FF207    0206        push es ! call osif ! pop es
3071  =0FA8 5A                        pop dx                          ;multi_cnt used
3072  =0FA9 5926880E1100              pop cx ! mov u_mult_cnt,cl
3073  =0FAF 80FB017609    0FBD        cmp bl,1 ! jbe dr_r2
3074  =0FB4 B9100088FFFF                  mov cx,e_bad_read ! mov bx,0ffffh
3075  =0FBA 5858C3                        pop ax ! pop ax ! ret
```

```
3076  =
3077  =0F90 84C8                dr_r2:  mov cl,al
3078  =0F92 5803C2                       pop ax ! add ax,dx              ;adjust record #
3079  =0F95 5A83DA                       pop dx ! sub bx,dx              ;adjust # sectors
3080  =0F98 D1E2D1E2D1E2                 shl dx,1 ! shl dx,1 ! shl dx,1
3081  =0F9E 2601160400                   add u_dma_seg,dx
3082  =
3083  =                                         ; check for CTRL C.  If hit while loading
3084  =                                         ; last set of characters, this is the place to
3085  =                                         ; clean things up...
3086  =
3087  =0FA3 515053                       push cx ! push ax ! push bx
3088  =0FA6 B91B04E82DF2 0206            mov cx,f_ciostat ! call osif
3089  =0FAC 585859                       pop bx ! pop ax ! pop cx
3090  =
3091  =                                         ; Now see if CTRL C was hit during the
3092  =                                         ; the load.
3093  =
3094  =0FAF 8B365800                     mov si,rlr
3095  =0FB3 F7440680007400FEE            test p_flag[si],pf_ctlc ! jz dr_r1
3096        07            0FEE
3097  =0FBC B92B00B3FFFF                     mov cx,e_abort ! mov bx,0ffffh
3098  =0FC2 C3                               ret
3099  =0FC3 80F9007404    0FC7 dr_r1: cmp cl,0 ! je dr_r5
3100  =0FC8 B9FF00C3                         mov cx,0ffh ! ret
3101  =0FCC 83FB007585    0F81 dr_r5: cmp bx,0 ! jne drd_lp
3102  =0FD1 33C9                             xor cx,cx
3103  =0FD3 C3                               ret
3104  =
3105  =                         if mpm
3106  =
3107  =                         get_sh:
3108  =                         ;------
3109  =                         ; Allocate memory for shared code.  If memory already
3110  =                         ; exists then mark LDTAB entry with FLEN=0 for no load.
3111  =                         ; START must be non-zero on success.
3112  =                         ;
3113  =                         ;       input:  SI = LDTAB Entry for shared code
3114  =                         ;       output: CX = 0 on success
3115  =                         ;                  = 0ffffh on failure
3116  =                         ;               SI is unchanged
3117  =                         ;
3118  =
3119  =                                 ; 1. Look for PD Name on SCL, making sure
3120  =                                 ;       LDSK and LUSER are the same.
3121  =
3122  =                                 mov bx,(offset scl)-p_thread
3123  =                         gs_nxt: push si
3124  =                                 mov dx,offset lod_fcb ! add dx,fcb_name
3125  =                                 mov cx,f_findpdname ! call osif
3126  =                                         ; BX=pd found or 0ffffh
3127  =                                 pop si
3128  =                                 cmp bx,0ffffh ! je no_sh
```

```
3129
3130  =                         mov al,lod_disk
3131  =                         cmp p_ldsk[bx],al ! jne gs_nxt
3132  =                             mov al,lod_user
3133  =                             cmp p_luser[bx],al ! jne gs_nxt
3134  =
3135  =                             ; 2. if (1.) then Share the Memory.
3136  =                             ; 2.1  Set FLEN=0
3137  =
3138  =                                 push bx ! push si
3139  =                                 call shmem
3140  =                                 pop si ! pop bx
3141  =                                 cmp cx,0ffffh ! je no_sh
3142  =                                     mov ldt_flen[si],0
3143  =
3144  =                                             ; Put SHARE PD on end of SCL
3145  =                                             ; BX = SHARE PD
3146  =
3147  =                                 pushf ! cli
3148  =                                 mov di,(offset SCL)-p_thread
3149  =                 sh_nin1:        cmp p_thread[di],bx ! je sh_rm  ;look for share PD
3150  =                                     mov di,p_thread[di] ! jmps sh_nin1
3151  =                 sh_rm:          mov ax,p_thread[bx]             ;take it off the list
3152  =                                 mov p_thread[di],ax
3153  =                 sh_in:          cmp p_thread[di],0 ! je sh_end  ;look for the end
3154  =                                     mov di,p_thread[di] ! jmps sh_in
3155  =                 sh_end:         mov p_thread[di],bx             ;insert share PD on end
3156  =                                 xor cx,cx
3157  =                                 mov p_thread[bx],cx
3158  =                                 popf ! ret                      ;success
3159  =
3160  =                             ; 3. if (NOT 1.) allocate memory to NEW PD
3161  =                 no_sh:
3162  =
3163  =                                 ; get new PD
3164  =
3165  =                     pushf ! cli
3166  =                     mov bx,pul
3167  =                     cmp bx,0 ! je sherr
3168  =                         mov ax,p_link[bx] ! mov pul,ax
3169  =                         popf
3170  =
3171  =                                 ; alloc memory for code segment
3172  =
3173  =                             push bx ! push si
3174  =                             mov cx,f_malloc
3175  =                             mov dx,si
3176  =                             call osif
3177  =                             pop si ! pop bx
3178  =                             cmp cx,0 ! jne merr
3179  =
3180  =                                     ; initialize new PD name
3181  =                                     ; BX = New PD
```

```
3182
3183  =                                 push si
3184  =                                 mov si,ldt_pd[si]               ;SI=old pd
3185  =                                 push si
3186  =                                 mov di,bx
3187  =                                 add si,p_name ! add di,p_name
3188  =                                 mov cx,4
3189  =                                 push es ! mov ax,ds ! mov es,ax
3190  =                                 rep movsw
3191  =                                 pop es ! pop di ! pop si
3192  =
3193  =                                         ; DI = old PD, BX=New PD
3194  =
3195  =                                 mov al,lod_user ! mov p_luser[bx],al
3196  =                                 mov al,lod_disk ! mov p_ldsk[bx],al
3197  =
3198  =                                         ; share w/new PD
3199  =
3200  =                                 mov ax,ldt_start[si]
3201  =                                 push bx ! push si ! push ds
3202  =                                 push ax ! push bx ! push di
3203  =                                 mov ax,ss ! mov ds,ax
3204  =                                 mov dx,sp ! mov cx,f_share
3205  =                                 call osif
3206  =                                 add sp,6
3207  =                                 pop ds ! pop si ! pop bx
3208  =
3209  =                                 ; put new PD on SCL
3210  =
3211  =                                 pushf ! cli
3212  =                                 mov di,(offset SCL)-p_thread
3213  =                         sh_nin: cmp p_thread[di],0 ! je sh_doit
3214  =                                     mov di,p_thread[di] ! jmps sh_nin
3215  =                         sh_doit: mov p_thread[di],bx            ;insert new share PD
3216  =                                 xor cx,cx
3217  =                                 mov p_thread[bx],cx
3218  =                                 popf ! ret                      ;success
3219  =                         merr:       pushf ! cli
3220  =                                     mov ax,pul
3221  =                                     mov p_link[bx],ax
3222  =                                     mov pul,bx
3223  =                         sherr:  popf
3224  =                                 mov cx,0ffffh ! ret
3225  =
3226  =                         shmem:
3227  =                         ;-----
3228  =                         ;input: ST = LDTAB
3229  =                         ;       BX = Owner PD
3230  =                         ;       Load_pd = Requestor
3231  =                         ;       have to set LDT_START
3232  =                         ;
3233  =
3234  =                                 lea di,(p_mem-ms_link)[bx]
```

```
3235
3236  =                         sm_nxt:
3237  =                                 mov di,ms_link[di]
3238  =                                 cmp di,0 ! je sm_no
3239  =                                     mov ax,ms_flags[di]
3240  =                                     and ax,mf_share+mf_code+mf_load
3241  =                                     cmp ax,mf_share+mf_code+mf_load ! jne sm_nxt
3242  =                                         push si ! push ds
3243  =                                         push ms_start[di]
3244  =                                         push ms_start[di]
3245  =                                         push lod_pd
3246  =                                         push bx
3247  =                                         mov ax,ss ! mov ds,ax
3248  =                                         mov dx,sp ! mov cx,f_share
3249  =                                         call osif
3250  =
3251  =                                                 ; This will always work.
3252  =
3253  =                                         pop ax ! pop ax ! pop ax
3254  =                                         pop dx ! pop ds ! pop si
3255  =                                         mov ldt_start[si],dx
3256  =                                         ret
3257  =                         sm_no:  mov cx,0ffffh ! ret
3258  =
3259  =                         endif
3260
```

```
3261
3262  =                         eject 1 include parse.sup
3263  =                       ;*****************************************************
3264  =                       ;*
3265  =                       ;*      Parse Filename Function
3266  =                       ;*
3267  =                       ;*****************************************************
3268  =
3269  =                       ;=========              ==========================
3270  =                       parse_ent:              ; Parse filename into FCB
3271  =                       ;=========              ==========================
3272  =                       ;       input:  DX -> FCB in u_wrkseg
3273  =                       ;       output: DX = 0ffffh on error
3274  =                       ;                  = 0      on last name in string
3275  =                       ;                  = offset of delimiter following name
3276  =                       ;               CX = Error Code
3277  =                       ;
3278  =                       ;       FCB:    +----+----+----+----+
3279  =                       ;               | NAMEADR | FCBADR  |
3280  =                       ;               +----+----+----+----+
3281  =                       ;
3282  =                       ;       NAMEADR = offset of String to parse
3283  =                       ;       FCBADR  = offset of FCB in u_wrkseg
3284  =                       ;
3285  =                       ;       Parsed FCB:
3286  =                       ;        0      => drive, 0=default, 1=A, 2=B, ...
3287  =                       ;        1- 8   => name, converted to upper case,
3288  =                       ;                  padded w/blanks
3289  =                       ;        9-11   => type, converted to upper case,
3290  =                       ;                  padded w/blanks
3291  =                       ;       12-15   => set to zero
3292  =                       ;       16-23   => password, convered to upper case,
3293  =                       ;                  padded w/blanks
3294  =                       ;       24-25   => address of password field in 'filename',
3295  =                       ;                  set to zero if password length=0
3296  =                       ;       26      => length of password (0-8)
3297  =                       ;       27-31   => left as is
3298  =
3299  =0FFF 26A12E00                  mov ax,u_wrkseg
3300  =1003 1E8ED8                    push ds | mov ds,ax                     ;DS->u_wrkseg
3301  =1006 068EC0                    push es | mov es,ax                     ;ES->u_wrkseg for string ops
3302  =1009 8BDA8B37                  mov bx,dx | mov si,pcb_flnmptr[bx]      ;SI->nxt char in parse string
3303  =100D 8B5F02                    mov bx,pcb_fcbptr[bx]                   ;BX -> FCB
3304  =1010 E83901E82701 1148         call fcbi | call debl                   ;init FCB, deblank parse str
3305  =1016 BD1800                    mov bp,e_badfname                       ;BAD FILENAME
3306  =                                                                       ;BP=error code if error
3307  =1019 E8C0007503    10FC        call delim | jnz prs1                   ;chk 1st char
3308  =101E E9B100        10D2            jmp pfn_endp
3309  =1021 8AE8468A04        prs1:   mov ch,al | inc si | mov al,[si]        ;see if Disk spec
3310  =1026 3C3A7529      1053        cmp al,':' | jne prs2
3311  =102A BD1700                        mov bp,e_illdisk                    ;drive specified
3312  =1020 8AC52C417235  1068            mov al,ch | sub al,'A' | jc pfn_err ;see if legal
3313  =1033 3C107D31      1068            cmp al,16 | jge pfn_err
```

```
3314
3315  =1037 FEC08807                    inc al ! mov fcb_dr[bx],al      ;good drv, set fcb_dr field
3316  =103B BD1900                      mov bp,e_badfname
3317  =103E 46E85A007510 10FC           inc si ! call delim ! jnz prs3  ;check nxt char
3318  =1044 3C2E7420     1068               cmp al,'.' ! je pfn_err             ;hit delim
3319  =1048 3C3A741C     1068               cmp al,':' ! je pfn_err             ;  see if legal
3320  =104C 3C337418     1068               cmp al,';' ! je pfn_err
3321  =1050 E90100       1054                   jmp prs3
3322  =1053 4E                  prs2:   dec si                          ;use default disk
3323  =
3324  =                         prs3:                                   ;parse filename
3325  =1054 8BFB8D5D01                  mov di,bx ! lea bx,fcb_name[di]
3326  =1059 B503E87C00   10DA           mov ch,8 ! call setfld          ;fill in FCB
3327  =105E 80FD00750D   1070           cmp ch,0 ! jne prs4
3328  =1063 E8960074O8   10FC               call delim ! jz prs4        ;see if more than 8 chars
3329  =1068 8BCD                 pfn_err:    mov cx,bp
3330  =106A BBFFFF071FC3                     mov bx,0ffffh ! pop es ! pop ds ! ret
3331  =1070 3C2E7519     108D prs4:  cmp al,'.' ! jnz prs5                ;see if filetype
3332  =1074 BD1900                       mov bp,e_badftype
3333  =1077 B503B05D09                   mov ch,3 ! lea bx,fcb_type[di]
3334  =107C 46E85A00     10DA            inc si ! call setfld            ;fill in FCB
3335  =1080 80FD007508   108D            cmp ch,0 ! jne prs5
3336  =1085 E874007403   10FC                call delim ! jz prs5
3337  =108A E9D3FF       1068                  jmp pfn_err
3338  =
3339  =                                  ;parse passwd
3340  =                         prs5:
3341  =108D 3C33752D     108E            cmp al,';' ! jnz prs8           ;see if password delim
3342  =1091 BD2500                       mov bp,e_ill_passwd
3343  =1094 B5088D5D10                   mov ch,8 ! lea bx,fcb_pwd[di]
3344  =1099 46                           inc si
3345  =109A 897518                       mov fcb_pptr[di],si             ;pointer to password if any
3346  =109D E83A00       10DA            call setfld                     ;yes
3347  =10A0 B1082ACD                     mov cl,8 ! sub cl,ch
3348  =10A4 884D1A                       mov fcb_plen[di],cl
3349  =10A7 80F9007505   10B1            cmp cl,0 ! jne prs51
3350  =10AC C745180000                       mov fcb_pptr[di],0          ;no password
3351  =10B1 80FD007508   10BE prs51:     cmp ch,0 ! jne prs8
3352  =10B6 E843007403   10FC                call delim ! jz prs8
3353  =10BB E9AAFF       1068                    jmp pfn_err
3354  =
3355  =10BE 8BDEE87A00   113D prs8:   mov bx,si ! call debl              ;see if more to parse
3356  =10C3 E835007508   10FC          call delim ! jnz pfn_out          ;if yes,exit
3357  =10C8 8BDE                         mov bx,si
3358  =10CA 3C007404     10D2            cmp al,0 ! je pfn_endp
3359  =10CE 3C0D7503     10D5            cmp al,cr ! jne pfn_out
3360  =10D2 BB0000               pfn_endp:   mov bx,0                    ;NOPE
3361  =10D5 2BC9071FC3           pfn_out:sub cx,cx ! pop es ! pop ds ! ret   ;exit
3362  =
3363  =                          setfld:     ; fill in a field until end of field or delim
3364  =                          ;------
3365  =10DA 5AE81E0052   10FC          pop dx ! call delim ! push dx
3366  =10DF 741A         10FB          jz setret
```

```
3367
3368  =10E1 3C2A750B      10F0              cmp al,'*' ! jnz setfld1
3369  =10E5 C6073F                          mov byte ptr [bx],'?'
3370  =10E8 43FEC975ED    10D8              inc bx ! dec ch ! jnz setfld
3371  =10ED E90500        10F5                  jmp setfld2
3372  =10F0 890743FEC9    10D8  setfld1:    mov [bx],al ! inc bx ! dec ch
3373  =10F5 4680FB0075DF  10D8  setfld2:    inc si ! cmp ch,0 ! jne setfld
3374  =10FB C3                  setret: ret
3375  =
3376  =                          delim:          ; see if char is delim, if not UPPER it, err if illegal
3377  =                          ;-----
3378  =                          ;       input:  SI -> next char in parse string
3379  =                          ;       output: 'z' flag on if delimiter found
3380  =                          ;               character converted to UPPER if 'a'-'z'
3381  =                          ;               AL = converted char
3382  =
3383  =10FC B10E8A04                    mov cl,ndelims ! mov al,[si]          ;See if Delimiter
3384  =1100 57BF2F11                    push di ! mov di,offset delims        ;look at delim string
3385  =1104 80F900740C    1115  delnxt: cmp cl,0 ! je no_del                  ;if end of delims,not delim
3386  =1109 2E3805000074  1120          cmp cs:[di],al ! je delret            ;if delim, ret w/ z set
3387        10            1120
3388  =1110 FEC947EBEF    1104              dec cl ! inc di ! jmps delnxt     ;try next delim
3389  =1115 3C207708      1121  no_del: cmp al,' ' ! ja del_up                ;not delim, check graphic
3390  =1119 5F58B91800                  pop di ! pop bx ! mov cx,e_badfname ;not graphic, err
3391  =111C E947FF        1068          jmp pfn_err                           ;go directly out, bypass ret
3392  =1121 3C617208      1120  del_up: cmp al,'a' ! jb delret                 ;if below 'a', no high bit,ret
3393  =1125 3C7A7702      1128          cmp al,'z' ! ja del_noup              ;if above 'z',
3394  =1129 245F                        and al,05fh                           ;make 'a'-'z' UPPER CASE
3395  =112B 247F                delnoup:  and al,07fh                         ;strip high bit
3396  =112D 5FC3                delret: pop di ! ret
3397  =
3398  =112F 00090D2E3A3B        delims  db      0,tab,cr,'.:;=,/[]<> '
3399        3D2C2F5B5D3C
3400        3E20
3401  =  000E                   ndelims equ     (offset $)-(offset delims)
3402  =
3403  =                          debl:                   ;strip leading blanks
3404  =                          ;-------
3405  =                          ;       input:  SI -> parse string
3406  =                          ;       output: SI -> first non-blank or tab char
3407  =
3408  =113D 803C207406    1148          cmp byte ptr [si],' ' ! je blnk
3409  =1142 803C097401    1148              cmp byte ptr [si],tab ! je blnk
3410  =1147 C3                              ret
3411  =1148 46EBF2        113D  blnk:   inc si ! jmps debl
3412  =
3413  =                          fcbi:           ; Initialize FCB
3414  =                          ;-------
3415  =114B 8BFB32C0                    mov di,bx ! sub ax,ax
3416  =114F 2BC9AA                      sub cx,cx ! stosb                     ; 0    =0
3417  =1152 B020B10BF3AA                mov al,' ' ! mov cl,11 ! rep stosb    ; 1-11=' '
3418  =1158 B000B102F3AB                mov al,0 ! mov cl,2 ! rep stosw       ;12-15=0
3419  =115E B020B108F3AA                mov al,' ' ! mov cl,8 ! rep stosb     ;16-23=' '
```

```
3420
3421  =1164 B000B103F3AA            mov al,0 ! mov cl,3 ! rep stosb        ;24-26=0
3422  =116A C3                      ret
3423
```

```
3424
3425  =                         eject ! include rpl.sup
3426  =                       ;*************************************************************
3427  =                       ;*
3428  =                       ;*      Call Resident System Procedure
3429  =                       ;*
3430  =                       ;*************************************************************
3431  =
3432  =  0000                 cpb_name        equ     word ptr 0
3433  =  0008                 cpb_param       equ     word ptr cpb_name + qnamsiz
3434  =
3435  =                       ;=======
3436  =                       rpl_ent:        ; Call Resident Procedure Library
3437  =                       ;=======
3438  =                       ;       input:  DX = CPB address in u_wrkseg
3439  =                       ;       output: BX = return from RPL (also u_retseg)
3440  =                       ;                    1 if error
3441  =                       ;               CX = error code
3442  =                       ;
3443  =                       ;               The stack is used like this:
3444  =                       ;
3445  =                       ; stack bottom  -------------------------               higher memory
3446  =                       ;            26 |   starting DS (sysdat)|
3447  =                       ;            24 |   starting ES (uda)   |
3448  =                       ;            22 |   seg of rpl_ret:     |
3449  =                       ;            20 |   offset of rpl_ret:  |
3450  =                       ;            18 |   seg of the RPL      |
3451  =                       ;            16 |   offset or RPL       |<-------
3452  =                       ;            14 | Q   /\                |        |
3453  =                       ;            12 | P   ||                |        |
3454  =                       ;            10 | |   ||                |        |
3455  =                       ;             8 | B qname               |        |
3456  =                       ;             6 | L buffptr             |------->|
3457  =                       ;             4 | D nmsgs               |
3458  =                       ;             2 | C qaddr               |
3459  =                       ;             0 | K flgs & net          |<------SP
3460  =                       ; stack top     -------------------------               lower memory
3461  =
3462  =
3463  =1168 1E06                       push ds ! push es
3464  =116D 268E1E2E00                 mov ds,u_wrkseg                 ; save ds
3465  =1172 8BF2                       mov si,dx
3466  =1174 0E                         push cs                         ; rpl_ret segment
3467  =1175 B8DA1150                   mov ax,offset rpl_ret ! push ax ; rpl_ret offset
3468  =1179 2BC05050                   sub ax,ax ! push ax ! push ax   ; QPB buffer
3469  =117D 8BFC                       mov di,sp                       ; DI -> buffer
3470  =117F FF7406                     push (cpb_name+6)[si]           ; qpb_name
3471  =1182 FF7404                     push (cpb_name+4)[si]
3472  =1185 FF7402                     push (cpb_name+2)[si]
3473  =1188 FF34                       push cpb_name[si]
3474  =118A 8B7408                     mov si,cpb_param[si]            ; SI=param
3475  =118D 57                         push di                         ; qpb_buffer address
3476  =118E 4050                       inc ax ! push ax                ; qpb_nmsgs
```

```
3477
3478  =1190 485050                      dec ax ! push ax ! push ax      ; qpb_addr,flg,net
3479  =1193 8BD48BFA                    mov dx,sp ! mov di,dx
3480  =1197 8CD08ED8                    mov ax,ss ! mov ds,ax
3481  =119B B89700                      mov cx,f_qopen
3482  =119E 9A57E863F0    0206          push si ! push di ! call osif
3483  =11A3 5F5E                        pop di ! pop si
3484  =11A5 83F9007539    11E3          cmp cx,0 ! jne rpl_err
3485  =11AA 8B5D02                          mov bx,qpb_qaddr[di]
3486  =11AD 1E2E8E1F0600                    push ds ! mov ds,sysdat
3487  =11B3 F7470420001F                    test q_flags[bx],qf_rpl ! pop ds
3488  =11B9 7428          11E3              jz rpl_err
3489  =11BB B989008B07                          mov cx,f_qread ! mov dx,di
3490  =11C0 56E842F05E    0206                  push si ! call osif ! pop si
3491  =11C5 83F9007519    11E3                  cmp cx,0 ! jne rpl_err
3492  =11CA 83C410                              add sp,16
3493  =11CD 8BD6                                mov dx,si
3494  =11CF 268E1E2E00                          mov ds,u_wrkseg
3495  =11D4 268E063000                          mov es,u_retseg
3496  =11D9 CB                                  retf
3497  =11DA 8CC0                rpl_ret:        mov ax,es
3498  =11DC 071F                                pop es ! pop ds
3499  =11DE 26A33000                            mov u_retseg,ax
3500  =11E2 C3                                  ret
3501  =                         rpl_err:
3502  =11E3 83C418                      add sp,24
3503  =11E6 071F                        pop es ! pop ds
3504  =11E8 BB0100B90900                mov bx,1 ! mov cx,e_no_queue
3505  =11EE C3                          ret
3506
```

```
3507
3508  =                         eject 1 include patch.cod
3509  =                     ;*****************************************************
3510  =                     ;*
3511  =                     ;*      PATCH AREA  --  128 bytes long
3512  =                     ;*
3513  =                     ;*****************************************************
3514  =
3515  =                     patch:
3516  =11EF 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;00-0f
3517  =11F5 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3518  =11FB 90909090                nop ! nop ! nop ! nop
3519  =
3520  =11FF 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;10-1f
3521  =1205 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3522  =120B 90909090                nop ! nop ! nop ! nop
3523  =
3524  =120F 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;20-2f
3525  =1215 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3526  =121B 90909090                nop ! nop ! nop ! nop
3527  =
3528  =121F 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;30-3f
3529  =1225 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3530  =122B 90909090                nop ! nop ! nop ! nop
3531  =
3532  =122F 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;40-4f
3533  =1235 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3534  =123B 90909090                nop ! nop ! nop ! nop
3535  =
3536  =123F 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;50-5f
3537  =1245 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3538  =124B 90909090                nop ! nop ! nop ! nop
3539  =
3540  =124F 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;60-6f
3541  =1255 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3542  =125B 90909090                nop ! nop ! nop ! nop
3543  =
3544  =125F 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;70-7f
3545  =1265 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3546  =126B 90909090                nop ! nop ! nop ! nop
3547  =
3548
```

```
3549
3550  =                       eject ! include basep.fmt
3551  =                     ;*****************************************************
3552  =                     ;*
3553  =                     ;*      base Page Format
3554  =                     ;*
3555  =                     ;*****************************************************
3556  =
3557  =                       DSEG
3558  =                       org     0
3559  =
3560  =0000                 bpg_clen        rb      3
3561  =0003                 bpg_cseg        rw      1
3562  =0005                 bpg_8080        rb      1
3563  =
3564  =0006                 bpg_dlen        rb      3
3565  =0009                 bpg_dseg        rw      1
3566  =000B                 bpg_dxxx        rb      1
3567  =
3568  =000C                 bpg_elen        rb      3
3569  =000F                 bpg_eseg        rw      1
3570  =0011                 bpg_exxx        rb      1
3571  =
3572  =0012                 bpg_slen        rb      3
3573  =0015                 bpg_sseg        rw      1
3574  =0017                 bpg_sxxx        rb      1
3575  =
3576  =                       org     050h
3577  =
3578  =0050                 bpg_lddsk       rb      1
3579  =0051                 bpg_pw1ptr      rw      1
3580  =0053                 bpg_pw1len      rb      1
3581  =0054                 bpg_pw2ptr      rw      1
3582  =0056                 bpg_pw2len      rb      1
3583  =
3584  =                       org     05ch
3585  =
3586  =005C                 bpg_fcb0        rb      0
3587  =
3588  =                       org     06ch
3589  =
3590  =006C                 bpg_fcb1        rb      0
3591  =
3592  =                       org     80h
3593  =
3594  =0080                 bpg_dma         rb      0
3595  =
3596  =                       org     100h
3597  =
3598  =0100                 bpg_udata       rb      0
3599
```

```
3600
3601  =                         eject 1 include lstk.fmt
3602  =                       ;*****************************************************
3603  =                       ;*
3604  =                       ;*      Load Stack Format
3605  =                       ;*
3606  =                       ;*****************************************************
3607  =
3608  =  0060                lstklen         equ     96 * byte
3609  =
3610  =                               DSEG
3611  =                               org     lstklen - 10
3612  =
3613  =0056                  ls_sp           rw      0
3614  =0056                  ls_offset       rw      1
3615  =0058                  ls_cseg         rw      1
3616  =005A                  ls_flags        rw      1
3617  =005C                  ls_roffset      rw      1
3618  =005E                  ls_rcseg        rw      1
3619
```

```
3620
3621  =                       eject 1 include vec.fmt
3622  =                     ;*******************************************************
3623  =                     ;*
3624  =                     ;* Interrupt Vectors - to fiddle with the interrupt
3625  =                     ;*       vectors, set the Data Segment Register to 0
3626  =                     ;*       and use the following variables.
3627  =                     ;*
3628  =                     ;*******************************************************
3629  =
3630  =                             DSEG
3631  =
3632  =0000                 i_divide_ip     rw      1       ; int 0
3633  =0002                 i_divide_cs     rw      1
3634  =0004                 i_trace_ip      rw      1       ; int 1
3635  =0006                 i_trace_cs      rw      1
3636  =0008                 i_nomask_ip     rw      1       ; int 2
3637  =000A                 i_nomask_cs     rw      1
3638  =000C                 i_break_ip      rw      1       ; int 3
3639  =000E                 i_break_cs      rw      1
3640  =0010                 i_ovrflw_ip     rw      1       ; int 4
3641  =0012                 i_ovrflw_cs     rw      1
3642  =0014                 i_interrupts    rw      ((osint-5)*2)
3643  =0380                 i_os_ip         rw      1
3644  =0382                 i_os_cs         rw      1
3645  =0384                 i_debug_ip      rw      1
3646  =0386                 i_debug_cs      rw      1
3647
```

```
3648
3649  =                         eject 1 include uda.fmt
3650  =                       ;*******************************************************
3651  =                       ;*
3652  =                       ;*      User Data Area - The User Data Area is an
3653  =                       ;*      extension of the process descriptor but it
3654  =                       ;*      travels with the user.  It contains info
3655  =                       ;*      that is needed only while in context.
3656  =                       ;*
3657  =                       ;*      While in the operating system, The Extra
3658  =                       ;*      Segment register points to the beginning
3659  =                       ;*      of the User Data Area.
3660  =                       ;*
3661  =                       ;*******************************************************
3662  =
3663  =  0060                 ud_insys        equ     60h
3664  =
3665  =                               eseg
3666  =                               org     0
3667  =
3668  =0000                   u_dparam        rw      1       ; arg to dispatch
3669  =
3670  =                       ;       this area overlays part of BDOS
3671  =0002                   u_dma_ofst      rw      1       ; BDOS dma offset
3672  =0004                   u_dma_seg       rw      1       ; BDOS dma segment
3673  =0006                   u_func          rb      1       ; actual function number
3674  =0007                   u_searchl       rb      1       ; BDOS search length
3675  =0008                   u_searcha       rw      1       ; BDOS search FCB offset
3676  =000A                   u_searchabase   rw      1       ; BDOS search user's segment
3677  =000C                   u_dcnt          rw      1       ; BDOS directory count
3678  =000E                   u_dblk          rw      1       ; BDOS directory block #
3679  =0010                   u_error_mode    rb      1       ; BDOS error mode
3680  =0011                   u_mult_cnt      rb      1       ; BDOS multi-sector count
3681  =0012                   u_df_password   rb      8       ; BDOS default password
3682  =001A                   u_pd_cnt        rb      1       ; BDOS process count
3683  =  0019                 uda_ovl_len     equ     (offset $)-(offset u_dma_ofst)
3684  =                       ;       end of overlay area
3685  =
3686  =
3687  =0018                   u_in_int        rb      1
3688  =001C                   u_sp            rw      1       ; save register area
3689  =001E                   u_ss            rw      1
3690  =0020                   u_ax            rw      1
3691  =0022                   u_bx            rw      1
3692  =0024                   u_cx            rw      1
3693  =0026                   u_dx            rw      1
3694  =0028                   u_di            rw      1
3695  =002A                   u_si            rw      1
3696  =002C                   u_bp            rw      1
3697  =002E                   u_wrkseg        rw      1       ; curr seg addr of buf
3698  =0030                   u_retseg        rw      1       ; usr ES return
3699  =0032                   u_ds_sav        rw      1       ;\
3700  =0034                   u_stack_sp      rw      1       ; usr stack segment
```

```
3701
3702  =0036                 u_stack_ss      rw      1       ; usr stack pointer
3703  =0038                 u_ivectors      rw      4       ; save int 0,1
3704  =0040                 u_unused        rw      2       ;
3705  =0044                 u_ivectors2     rw      4       ; save int 3,4
3706  =0048                 u_es_sav        rw      1       ; > Used during interrupts
3707  =004c                 u_flag_sav      rw      1       ;/
3708  =
3709  =0050                 u_initcs        rw      1
3710  =0052                 u_initus        rw      1
3711  =0054                 u_inites        rw      1
3712  =0056                 u_initss        rw      1
3713  =0058                 u_os_ip         rw      1       ; O.S. vec save
3714  =005A                 u_os_cs         rw      1
3715  =005C                 u_debug_ip      rw      1       ; RTS,Debug Vector Save
3716  =005E                 u_debug_cs      rw      1
3717  =0060                 u_insys         rb      1       ; # times through user_entry
3718  =0061                 u_stat_sav      rb      1       ; used during interrupts
3719  =0062                 u_conccb        rw      1       ; default console's CCB addr
3720  =0064                 u_lstccb        rw      1       ; default list devices CCB addr
3721  =0066                 u_delim         rb      1       ; delimiter for user function 9
3722  =
3723  =                     ;       org     ulen
3724  =                     ;u_8087         rw      47      ; 8087 save area
3725  =                     ;                               ; see dispatcher, loader and terminate
3726
```

```
3727
3728  =                         eject | include sysdat.dat
3729  =                 ;*********************************************************
3730  =                 ;*
3731  =                 ;*      System Data Area
3732  =                 ;*
3733  =                 ;*********************************************************
3734  =
3735  =                         CSEG
3736  =                         org     0ch
3737  =                 ;dev_ver
3738  =000C 06                         db      6       ;development system data version
3739  =                                                 ;SYSDAT.CON has 16 byte code segment
3740  =
3741  =                         DSEG
3742  =                         org     0
3743  =
3744  =                         ;
3745  =                         ;This data is initialized by GENCCPM
3746  =                         ;
3747  =
3748  =                         ;Module Table - contains the FAR CALL addresses
3749  =                         ;       of each module for their initialization
3750  =                         ;       and entry routines.
3751  =                         ;
3752  =                         ;       +---+---+---+---+---+---+---+---+
3753  =                         ;       |     entry     |   initialize  |
3754  =                         ;       +---+---+---+---+---+---+---+---+
3755  =                         ;
3756  =                         ;             entry    init
3757  =                         ;             -----    ----
3758  =
3759  =  0000                 module_table    equ     dword ptr (offset $)
3760  =  0000                 supmod          equ     (offset $)
3761  =0000 030000000000                      dw      3,0,    0,0     ;SUP
3762       0000
3763  =
3764  =  0006                 rtmmod          equ     (offset $)
3765  =0008 030000000000                      dw      3,0,    0,0     ;RTM
3766       0000
3767  =
3768  =  0010                 memmod          equ     (offset $)
3769  =0010 030000000000                      dw      3,0,    0,0     ;MEM
3770       0000
3771  =
3772  =  0018                 ciomod          equ     (offset $)
3773  =0018 030000000000                      dw      3,0,    0,0     ;CIO
3774       0000
3775  =
3776  =  0020                 bdosmod         equ     (offset $)
3777  =0020 030000000000                      dw      3,0,    0,0     ;BDOS
3778       0000
3779  =
```

```
3780
3781  =  0028                  xiosmod         equ     (offset $)
3782  =0028 030C0000000C                       dw      0C03H,0,        0C00H,0 ;XIOS
3783        0000
3784  =
3785  =  0030                  netmod          equu    (offset $)
3786  =0030 030000000000                       dw      3,0,    0,0     ;NET
3787        0000
3788  =
3789  =  0038                  dispatcher      equ     (offset $)
3790  =0038 00000000                           dw      0,0             ;far dispatcher (does IRET)
3791  =
3792  =  003C                  rtm_pdisp       equ     (offset $)
3793  =003C 00000000                           dw      0,0             ;far dispatcher (does RETF)
3794  =
3795  =                                ; location in memory of MP/M-86 or CCP/M-86
3796  =
3797  =0040 0810               osseg           dw      1008h   ;1st parag. of MP/M-86 or CCP/M
3798  =0042 0000               rspseg          dw      0       ;segment of first RSP
3799  =0044 0000               endseg          dw      0       ;1st parag. outside of MP/M or CCP/M
3800  =
3801  =0046 3F                 module_map      db      03fh    ;bit map of modules that exist
3802  =                                                        ; in this system.  low order bit
3803  =                                                        ; corresponds to 1st module in
3804  =                                                        ; module table.  If bit is on,then
3805  =                                                        ; module exists.
3806  =
3807  =                                ; some answers to GENCCPM questions
3808  =
3809  =0047 04                 ncns            db      4       ;# system console devices
3810  =0048 01                 nlst            db      1       ;# system list devices
3811  =0049 05                 nccb            db      5       ;# character control blocks
3812  =004A 20                 nflags          db      20h     ;# flags
3813  =004B 01                 srchdisk        db      1       ;system disk
3814  =004C 0040               mmp             dw      04000h  ;Max Memory per process
3815  =004E 00                 nslaves         db      0       ;Number of network requestors
3816  =004F 00                 dayfile         db      0       ;if 0ffh,display command info
3817  =0050 01                 tempdisk        db      1       ;Temporary disk
3818  =0051 3C                 tickspersec     db      60      ;Number of ticks per second
3819  =
3820  =                                ; data lists created by GENCCPM
3821  =
3822  =0052 0000               free_root       dw      0       ;locked unused list
3823  =0054 0000               ccb             dw      0       ;addr. Console Ctrl Blk Table
3824  =0056 0000               flags           dw      0       ;addr. Flag Table
3825  =0058 2000               mdul            dw      020h    ;Mem descr. Unused List
3826  =005A 0000               mfl             dw      0       ;Memory Free List
3827  =005C 1400               pul             dw      014h    ;Proc. descr. Unused List
3828  =005E 2000               qul             dw      020h    ;QCB Unused List
3829  =                                                        ;MAU for queue buffer info
3830  =0060 0000               qmau            dw      0       ;link
3831  =0062 0000                               dw      0       ;start segment
3832  =0064 0004                               dw      400h    ;length
```

```
3833
3834  =0066 0000                        dw      0       ;plist
3835  =
3836  =                         ;
3837  =                         ;This data is initialized at Assembly time
3838  =                         ;
3839  =
3840  =0068 7704       rlr             dw      initpd  ;Ready List Root
3841  =006A 0000       dlr             dw      0       ;Delay List Root
3842  =006C 0000       drl             dw      0       ;Dispatcher Ready List
3843  =006E 0000       plr             dw      0       ;Poll List Root
3844  =0070 0000       scl             dw      0       ;Shared Code List
3845  =0072 7704       thrdrt          dw      initpd  ;Process Thread Root
3846  =0074 9401       qlr             dw      mxloadqd;Queue List Root
3847  =0076 0000       mal             dw      0       ;Memory Alloc List
3848  =
3849  =                         ; Version information
3850  =
3851  =0078 0000       version         dw      unknown ;addr. version str in SUP code segment
3852  =                                         ;set by GENCCPM if CCP/M
3853  =                 if mpm
3854  =                 bvernum         dw      01130h  ;MPM-86 w/BDOS v3.0
3855  =                 osvernum        dw      01121h  ;MPM-86 V2.1
3856  =                 endif
3857  =
3858  =                 if ccpm
3859  =007A 3114        bvernum         dw      01431h  ;CCP/M w/BDOS 3.1
3860  =007C 2014        osvernum        dw      01420h  ;CCP/M V2.0
3861  =                 endif
3862  =
3863  =                         ; Time of Day Structure
3864  =
3865  =007E             tod             rw      0
3866  =007E 7E06        tod_day         dw      067EH   ;day since 1/1/78 (09 Jul 82)
3867  =0080 12          tod_hr          db      12h     ;hour of day
3868  =0081 00          tod_min         db      00h     ;minute of hour
3869  =0082 00          tod_sec         db      00h     ;second of minute
3870  =
3871  =                         ; info from XIOS
3872  =
3873  =0083 00          ncondev         db      0               ;# console devs in XIOS
3874  =0084 00          nlstdev         db      0               ;# character devs in XIOS
3875  =0085 00          nciodev         db      0               ;# character i/o devices
3876  =                                                         ; supported by XIOS.
3877  =0086 0000        lcb             dw      0               ; list control block address
3878  =0088 0000        openvec         dw      0               ; open file vector
3879  =008A 20          lock_max        db      20h             ; Max Locked Records/process
3880  =008B 20          open_max        db      20h             ; Max Open Files/process
3881  =008C 0000        owner8087       dw      0               ; no one owns it initially
3882  =008E             rsv             rw      1               ; RESERVED
3883  =0090 FF          cmod            db      0ffh            ; BDOS Compatibility
3884  =0091 00          ndp8087         db      false           ; true 8087 exits
3885  =                                                         ; (Numeric Data Processor)
```

```
3886
3887  =0092 00000000          err_intercept   dw      0,0             ; BDOS does a callf here
3888  =                                                               ; to print error msgs,
3889  =                                                               ; if second word is <> 0
3890  =0096 4106              slr             dw      offset mem_spb  ; Sync List Root
3891  =0098 000000000000              dw      0,0,0,0         ; RESERVED
3892        0000
3893  =
3894  =
3895  =                               ; SYSENT Table - MP/M-86, CCP/M-86 system function information
3896  =                               ;       The supervisor calls the appropriate module
3897  =                               ;       through this table.
3898  =                               ;
3899  =                               ;        Low Byte  High Byte
3900  =                               ;       +----+----+--------+
3901  =                               ;       |function |flgs|mod|
3902  =                               ;       +----+----+--------+
3903  =                               ;
3904  =                               ;       flgs    - 001h - network intercept
3905  =                               ;                 if on, the network module is called
3906  =                               ;                 first, on return, either the function
3907  =                               ;                 is called or it is considered complete
3908  =                               ;                 depending on the return.
3909  =                               ;       mod     - module number (0-15)
3910  =                               ;       function- function to call within module
3911  =
3912  =                               ;       note: sup function 0 returns not the
3913  =                               ;                 implemented error code to the caller,
3914  =                               ;                 and sup function 1 returns the illegal
3915  =                               ;                 function error code.
3916  =
3917  =                               ; standard CPM-2 functions
3918  =
3919  =                               ;       func,   module
3920  =
3921  =                               org     ((offset $) + 1) and 0fffeh
3922  =
3923  =00A0 0002              sysent  db      0,      rtm     ;  0-system reset
3924  =00A2 0004                      db      0,      cio     ;  1-conin
3925  =00A4 0104                      db      1,      cio     ;  2-conout
3926  =00A6 0001                      db      0,      sup     ;  3-raw conin/aux in
3927  =00A8 0001                      db      0,      sup     ;  4-raw conout/aux out
3928  =00AA 0404                      db      4,      cio     ;  5-list out
3929  =00AC 0504                      db      5,      cio     ;  6-raw conio
3930  =00AE 0001                      db      0,      sup     ;  7-getiobyte
3931  =00B0 0001                      db      0,      sup     ;  8-setiobyte
3932  =00B2 0604                      db      6,      cio     ;  9-conwrite
3933  =00B4 0704                      db      7,      cio     ; 10-conread
3934  =00B6 0804                      db      8,      cio     ; 11-constat
3935  =00B8 0201                      db      2,      sup     ; 12-get version
3936  =00BA 0005                      db      0,      bdos    ; 13-diskreset
3937  =00BC 0105                      db      1,      bdos    ; 14-diskselect
3938  =00BE 0205                      db      2,      bdos    ; 15-file open
```

```
3939
3940  =00C0 0305                db      3,      bdos    ; 16-file close
3941  =00C2 0405                db      4,      bdos    ; 17-search first
3942  =00C4 0505                db      5,      bdos    ; 18-search next
3943  =00C6 0605                db      6,      bdos    ; 19-file delete
3944  =00C8 0705                db      7,      bdos    ; 20-file read seq
3945  =00CA 0805                db      8,      bdos    ; 21-file write seq
3946  =00CC 0905                db      9,      bdos    ; 22-file make
3947  =00CE 0A05                db      10,     bdos    ; 23-file rename
3948  =00D0 0B05                db      11,     bdos    ; 24-login vector
3949  =00D2 0C05                db      12,     bdos    ; 25-get def disk
3950  =00D4 0D05                db      13,     bdos    ; 26-set dma
3951  =00D6 0E05                db      14,     bdos    ; 27-get alloc vector
3952  =00D8 0F05                db      15,     bdos    ; 28-write protect
3953  =00DA 1005                db      16,     bdos    ; 29-get r/O vector
3954  =00DC 1105                db      17,     bdos    ; 30-set file attr.
3955  =00DE 1205                db      18,     bdos    ; 31-get disk parm block
3956  =00E0 1305                db      19,     bdos    ; 32-user code
3957  =00E2 1405                db      20,     bdos    ; 33-file read random
3958  =00E4 1505                db      21,     bdos    ; 34-file write random
3959  =00E6 1605                db      22,     bdos    ; 35-file size
3960  =00E8 1705                db      23,     bdos    ; 36-set random record
3961  =00EA 1805                db      24,     bdos    ; 37-reset drive
3962  =00EC 1905                db      25,     bdos    ; 38-access drive
3963  =00EE 1A05                db      26,     bdos    ; 39-free drive
3964  =00F0 1B05                db      27,     bdos    ; 40-file write random w/zero fill
3965  =
3966  =                                         ;CPM-3 extensions
3967  =
3968  =00F2 0001                db      0,      sup     ; 41-Test and Write (NOT IMPLEMENTED)
3969  =                                         ; Would be BDOS func # 28
3970  =00F4 1C05                db      28,     bdos    ; 42-Lock Record
3971  =00F6 1D05                db      29,     bdos    ; 43-Unlock Record
3972  =00F8 1E05                db      30,     bdos    ; 44-Set Multi-sector
3973  =00FA 1F05                db      31,     bdos    ; 45-Set Bdos Error Mode
3974  =00FC 2005                db      32,     bdos    ; 46-Get Disk Free Space
3975  =00FE 0C01                db      12,     sup     ; 47-Chain to Program
3976  =                                         ; In CP/M-86 BDOS func # 34
3977  =0100 2105                db      33,     bdos    ; 48-Flush Buffers
3978  =0102 0101                db      1,      sup     ; 49-
3979  =
3980  =                                         ;CPM-86 extensions
3981  =
3982  =0104 0301                db      3,      sup     ; 50-call xios
3983  =0106 2205                db      34,     bdos    ; 51-set dma base
3984  =0108 2305                db      35,     bdos    ; 52-get dma
3985  =010A 0003                db      0,      mem     ; 53-get max mem
3986  =010C 0103                db      1,      mem     ; 54-get abs max mem
3987  =010E 0203                db      2,      mem     ; 55-alloc mem
3988  =0110 0303                db      3,      mem     ; 56-alloc abs mem
3989  =0112 0403                db      4,      mem     ; 57-free mem
3990  =0114 0503                db      5,      mem     ; 58-free all mem
3991  =0116 0401                db      4,      sup     ; 59-load
```

```
3992
3993  =0118 0101                db      1,      sup     ; 60-
3994  =011A 0101                db      1,      sup     ; 61-
3995  =011C 0101                db      1,      sup     ; 62-
3996  =011E 0101                db      1,      sup     ; 63-
3997  =
3998  =                                         ;CP/NET functions
3999  =
4000  =0120 4007                db      64,     net     ; 64-network login
4001  =0122 4107                db      65,     net     ; 65-network logoff
4002  =0124 4207                db      66,     net     ; 66-network send msg
4003  =0126 4307                db      67,     net     ; 67-network rcv msg
4004  =0128 4407                db      68,     net     ; 68-network status
4005  =012A 4507                db      69,     net     ; 69-get network config addr
4006  =
4007  =                                         ;CP/M-3 extensions
4008  =
4009  =012C 2405                db      36,     bdos    ; 98-Reset Alloc Vector
4010  =012E 2505                db      37,     bdos    ; 99-Truncate File
4011  =0130 2605                db      38,     bdos    ;100-Set Dir Label
4012  =0132 2705                db      39,     bdos    ;101-Return Dir Label
4013  =0134 2805                db      40,     bdos    ;102-Read File XFCB
4014  =0136 2905                db      41,     bdos    ;103-Write File XFCB
4015  =0138 2A05                db      42,     bdos    ;104-Set Date and Time
4016  =013A 2B05                db      43,     bdos    ;105-Get Date and Time
4017  =013C 2C05                db      44,     bdos    ;106-Set Default Password
4018  =013E 0D01                db      13,     sup     ;107-Return Serial Number
4019  =0140 0001                db      0,      sup     ;108-(not implemented)
4020  =0142 1904                db      25,     cio     ;109-Get/Set Console Mode
4021  =0144 1A04                db      26,     cio     ;110-Get/Set Output Delimiter
4022  =0146 1B04                db      27,     cio     ;111-Print Block
4023  =0148 1C04                db      28,     cio     ;112-List Block
4024  =
4025  =                                         ; MP/M functions
4026  =
4027  =014A 0603                db      6,      mem     ;128-mem req
4028  =014C 0603                db      6,      mem     ;129-(same function as 128)
4029  =014E 0703                db      7,      mem     ;130-mem free
4030  =0150 0102                db      1,      rtm     ;131-poll device
4031  =0152 0202                db      2,      rtm     ;132-flag wait
4032  =0154 0302                db      3,      rtm     ;133-flag set
4033  =0156 0402                db      4,      rtm     ;134-queue make
4034  =0158 0502                db      5,      rtm     ;135-queue open
4035  =015A 0602                db      6,      rtm     ;136-queue delete
4036  =015C 0702                db      7,      rtm     ;137-queue read
4037  =015E 0802                db      8,      rtm     ;138-cond. queue read
4038  =0160 0902                db      9,      rtm     ;139-queue write
4039  =0162 0A02                db      10,     rtm     ;140-cond. queue write
4040  =0164 0B02                db      11,     rtm     ;141-delay
4041  =0166 0C02                db      12,     rtm     ;142-dispatch
4042  =0168 0D02                db      13,     rtm     ;143-terminate
4043  =016A 0E02                db      14,     rtm     ;144-create process
4044  =016C 0F02                db      15,     rtm     ;145-set priority
```

```
                                db      9,      cio     ;146-console attach
                                db      10,     cio     ;147-console detach
                                db      11,     cio     ;148-set def console
                                db      12,     cio     ;149-console assign
                                db      5,      sup     ;150-CLI
                                db      6,      sup     ;151-call RPL
                                db      7,      sup     ;152-parse filename
                                db      13,     cio     ;153-get def console
                                db      8,      sup     ;154-sysdat addr
                                db      9,      sup     ;155-time of day
                                db      16,     rtm     ;156-get PD addr
                                db      17,     rtm     ;157-abort process
                        =
                                ; MPM II extensions
                        =
                                db      15,     cio     ;158-attach list
                                db      16,     cio     ;159-detach list
                                db      17,     cio     ;160-set list dev
                                db      18,     cio     ;161-Cond. Attach list
                                db      19,     cio     ;162-Cond. Attach Console
                                db      11,     sup     ;163-MP/M Version Number
                                db      20,     cio     ;164-get list dev
                        =
                        =
                                ; Initialized Queues
                        =
                                org     ((offset $) + 1) and 0fffeh
                        =
0194 7403       mxloadqd        dw      mxdiskqd
0196 0000                       db      0,0
0198 0300                       dw      qf_keep+qf_mx
019A 40584C6F6164               db      '@XLoad  '
        2020
01A2 000001000000               dw      0,1,0,0,1,0,0
        000001000000
        0000
01B0 0000       mxloadqpb       db      0,0
01B2 940101000000               dw      mxloadqd,1,0
                ;               db      '@XLoad  '
                        =
                                ; Data Used by Load Program
                        =
                                org     ((offset $) + 1) and 0fffeh
                        =
0188 0000       lod_uda         dw      0
018A 0000       lod_lstk        dw      0
018C 0000       lod_basep       dw      0
018E 0000       lod_nlst        dw      0
01C0 0000       lod_pd          dw      0
01C2            lod_fcb         rs      36
01E6 0000       lod_indma       dw      0
01E8 00         lod_nrels       db      0
```

```
4098
4099  =01E9 00              lod_chain       db      0
4100  =01EA 00              lod_user        db      0
4101  =01EB 00              lod_disk        db      0
4102  =01EC 00              lod_fifty       db      0
4103  =01ED 00              lod_8080        db      0
4104  =01EE 00              lod_lbyte       db      0
4105  =01EF 0000            lod_fixrec      dw      0
4106  =01F1 0000            lod_fixrecl     dw      0
4107  =01F3                 lod_dma         rb      dskrecl
4108  =0273                 ldtab           rb      ldtabsiz
4109  =
4110  =031D                 cli_dma_ofst    rw      1
4111  =031F                 cli_dma_seg     rw      1
4112  =0321                 cli_pflag       rw      1
4113  =0323                 cli_chain       rb      1
4114  =0324                 cli_term        rb      1
4115  =0325                 cli_dma         rb      dskrecl         ;dma buffer
4116  =
4117  =                                     ;copy of user's clicb
4118  =03A5                 cli_net         rb      1               ;net
4119  =03A6                 cli_ppd         rw      1               ;parent PD
4120  =03A8                 cli_cmdtail     rb      129             ;command sent
4121  =0429                                 rb      1
4122  =
4123  =042A                 cli_fcb         rb      fcblen+1        ;internal FCB
4124  =
4125  =0448 0000            cli_cuspqpb     db      0,0             ;QPB of command
4126  =044A 00000000                        dw      0,0
4127  =044E A603                            dw      offset cli_ppd
4128  =0450 242424242424                    db      '$$$$$$$$'
4129        2424
4130  =
4131  =0458 0000            cli_ucb         db      0,0             ;cns,match
4132  =045A 0000                            dw      0               ;pd
4133  =045C 242424242424                    db      '$$$$$$$$'      ;name
4134        2424
4135  =
4136  =0464 A803            cli_pcb         dw      offset cli_cmdtail      ;parse
4137  =0466 2A04                            dw      offset cli_fcb          ;ctl bk
4138  =
4139  =0468 0000            cli_pd          dw      0       ;pd of load prog
4140  =046A 0000            cli_err         dw      0       ;error return
4141  =046C 0000            cli_bpage       dw      0       ;base page
4142  =046E 01              cli_lddsk       db      1       ;load disk
4143  =
4144  =                             ;parent information
4145  =
4146  =0470 00              cli_cns         db      0       ;pd.p_cns save
4147  =0471 00              cli_user        db      0       ;pd.p_dsk save
4148  =0472 00              cli_dsk         db      0       ;pd.p_user save
4149  =0473 00              cli_err_mode    db      0       ;u_error_mode save
4150  =0474 00              cli_dfil        db      0       ;dayfile flag
```

```
4151
4152  =
4153  =                                ;
4154  =                                ;System Initialization Variables
4155  =                                ;
4156  =
4157  =0477 0000              initpd    dw      0               ;link
4158  =0479 0000                        dw      0               ;thread
4159  =047B 00                          db      ps_run          ;stat
4160  =047C 01                          db      1               ;prior
4161  =047D 0500                        dw      pf_syst+pf_kernal;flag
4162  =047F 496E69742020                db      'Init  '        ;name
4163        2020
4164  =0487 0000                        dw      unknown         ;uda segment
4165  =0489 00                          db      0               ;disk
4166  =048A 00                          db      0               ;user
4167  =048B 0000                        db      0,0             ;ldsk,luser
4168  =048D 0000                        dw      0               ;mem
4169  =048F 0000                        dw      0               ;dvract
4170  =0491 0000                        dw      0               ;wait
4171  =0493 0000                        db      0,0             ;org,net
4172  =0495 0000                        dw      0               ;parent
4173  =0497 00                          db      0               ;cns
4174  =0498 00                          db      0               ;abort
4175  =0499 0000                        db      0,0             ;cin,cout
4176  =049B 00                          db      0               ;lst
4177  =049C 000000                      db      0,0,0           ;sf3,4,5
4178  =049F                             rb      4               ;reserved
4179  =04A3 00000000                    dw      0,0             ;pret,scratch
4180  =
4181  =
4182  =                                ;User Data Area of Init process
4183  =                                ;paragraph aligned
4184  =
4185  =                                org     ((offset $)+0fh) AND 0fff0h
4186  =0480                   inituda   rb      ulen
4187  =0580                   init_tos  rw      0
4188  =                                org     offset inituda + ud_insys
4189  =0510 01                          db      1               ;keep the SUP from doing stack
4190  =                                org     offset init_tos ;switches
4191  =
4192  =
4193  =                                ; RTM data
4194  =                                ; is word aligned from init uda
4195  =
4196  =0580 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4197  =0586 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4198  =058C CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4199  =
4200  =05C2 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4201  =05C8 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4202  =05CE CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4203  =05D4 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
```

```
4204
4205  =
4206  =0604 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4207  =0610 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4208  =061C CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4209  =0628 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4210  =
4211  =0634 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4212  =0640 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4213  =064C CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4214  =0658 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4215  =
4216  =0664 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4217  =0670 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4218  =067C CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4219  =0688 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
4220  =
4221  =0694                 dsptchtos       rw      0
4222  =
4223  =0696 00              indisp          db      false   ;?currently in dispatch?
4224  =0697 00              intflag         db      0       ;if 0, interrupts not enabled -
4225  =                                                     ;not implemented
4226  =0698 0000            es_sav          dw      0       ;(staying word aligned)
4227  =069A 0000            bx_sav          dw      0
4228  =069C 0000            dx_sav          dw      0
4229  =
4230  =                             ; MEM Data
4231  =
4232  =069E 0000            beststart       dw      0
4233  =06A0 0000            bestlen         dw      0
4234  =06A2 0000            bestsi          dw      0
4235  =06A4 0000            bestmau         dw      0
4236  =06A6 0000            currmau         dw      0
4237  =06A8 0000            currsi          dw      0
4238  =06AA 000000000000    currmpb         dw      0,0,0,0,0
4239        00000000
4240  =
4241  =
4242  =                     ;       SYNC Parameter Blocks
4243  =
4244  =                     ;       The MEM ENTRY: point uses the following for
4245  =                     ;       mutual exclusion and recursion.
4246  =
4247  =06B4 00              mem_cnt         db      0       ;how many times a process has recursivly
4248  =                                                     ;called the memory manager
4249  =
4250  =06B5 4906            mem_spb         dw      q_spb   ;link   Mem Sync Parameter Block
4251  =06B7 0000                            dw      0       ;owner
4252  =06B9 0000                            dw      0       ;wait
4253  =06BB 0000                            dw      0       ;next
4254  =
4255  =                     ;       The queue sub-system in the RTM uses the following
4256  =                     ;       structure for mutual exclusion
```

```
4257
4258  =
4259  =0649 5106              q_spb           dw      cli_spb ;link   Queue Sync Parameter Block
4260  =064B 0000                              dw      0       ;owner
4261  =064D 0000                              dw      0       ;wait
4262  =064F 0000                              dw      0       ;next
4263  =
4264  =
4265  =                       ;       The CLI uses the CLI_SPB for mutual exclusion
4266  =
4267  =0651 5905              cli_spb         dw      thrd_spb;link   CLI Sync Parameter Block
4268  =0653 0000                              dw      0       ;owner
4269  =0655 0000                              dw      0       ;wait
4270  =0657 0000                              dw      0       ;next
4271  =
4272  =                       ;       When the thread is accessed, the THRD_SPB must be owned
4273  =                       ;       first.
4274  =
4275  =0659 A00B              thrd_spb        dw      msg_spb ;link   Thread Sync Parameter Block
4276  =065B 0000                              dw      0       ;owner
4277  =065D 0000                              dw      0       ;wait
4278  =065F 0000                              dw      0       ;next
4279  =
4280  =                       ; Currently the order in which the SYNCs must be obtained if
4281  =                       ; more than one is needed is:
4282  =
4283  =                       ;       CLI
4284  =                       ;       QUEUE                   ;called by CLI for RSPs
4285  =                       ;       MEM                     ;called by make queue
4286  =                       ;       THREAD                  ;used from the MEM module
4287  =
4288  =                       ; The SYNCs must be released in reverse order
4289  =                       ; MSG_SPB is used by the BDOS to protect the BDOS error message
4290  =                       ; buffer.  MSG_SPB is in DATA.BDO
4291
```

```
4292
4293  =                         eject 1 include data.bdo
4294  =
4295  =                       ;*********************************************************
4296  =                       ;*
4297  =                       ;*       bdos data area
4298  =                       ;*
4299  =                       ;*********************************************************
4300  =
4301  =  FFFF                 CCP40FF equ     true
4302  =
4303  =                       if BCPM
4304  =
4305  =                       ;
4306  =                       ;       8086 variables that must reside in code segment
4307  =                       ;
4308  =                       cseg    $
4309  =                       ;
4310  =                       axsave          dw      0       ; register saves
4311  =                       ss_save         dw      0
4312  =                       sp_save         dw      0
4313  =                       stack_begin     dw      endstack
4314  =                       ;
4315  =                       ;       variables in data segment:
4316  =                       ;
4317  =                       dseg    cpmsegment
4318  =                               org     bdosoffset+bdoscodesize
4319  =
4320  =                       header  rs      128
4321  =                               rs      72
4322  =
4323  =                       pag0    dw      0               ;address of user's page zero
4324  =                       ip0     db      0               ;initial page value for ip register
4325  =                       ;
4326  =                       ;     memory control block
4327  =                       ;
4328  =                       umembase        dw      0       ;user'sbase for memory request
4329  =                       umemlg  dw      0               ;length of memory req
4330  =                       contf   db      0               ;flag indicates added memory is avail
4331  =                       ;
4332  =                       ;
4333  =                       hold_info       dw      0       ;save info
4334  =                       hold_spsave     dw      0       ;save user sp during program load
4335  =                       hold_sssave     dw      0       ;save user ss during program load
4336  =
4337  =                       mod8080         db      0
4338  =                       ;
4339  =                       ;       byte i/o variables:
4340  =                       ;
4341  =                       compcol db      0               ;true if computing column position
4342  =                       strtcol db      0               ;starting column position after read
4343  =                       column  db      0               ;column position
4344  =                       listcp  db      0               ;listing toggle
```

```
4345
4346  =                       kbchar  db      0               ;initial key char = 00
4347  =
4348  =                       endif
4349  =
4350  =                               dseg
4351  =                       if BNPM
4352  =                               org     0c00h   ;org for MP/M system
4353  =                       endif
4354  =
4355  =                       if BNPM and CCPMOFF
4356  =                               org     0800h   ;org for CCP/M system
4357  =                       endif
4358  =
4359  =                       if BNPM
4360  =0800 0000              rlog            dw      0       ;removeable logged-in disks
4361  =0802 0000              tlog            dw      0       ;removeable disk test login vector
4362  =0804 0000              ntlog           dw      0       ;new tlog vector
4363  =
4364  =                       endif
4365  =
4366  =0806 0000              rodsk           dw      0       ;read only disk vector
4367  =0808 0000              dlog            dw      0       ;logged-in disks
4368  =
4369  =080A 00                open_fnd        db      0               ;open file found in srch_olist
4370  =
4371  =                       ;the following variables are set to zero upon entry to file system
4372  =
4373  =080B 00                fcbdsk          db      0               ;disk named in fcb
4374  =080C 00                parlg           db      0               ;length of parameter block copied
4375  =080D 0000              aret            dw      0               ;adr value to return
4376  =  080D                 lret            equ     byte ptr aret   ;low(aret)
4377  =080F 00                resel           db      0               ;reselection flag
4378  =0810 00                rd_dir_flag     db      0               ;must read/locate directory record
4379  =0811 00                comp_fcb_cks    db      0               ;compute fcb checksum flag
4380  =0812 00                search_user0    db      0               ;search user 0 for file (open)
4381  =0813 00                make_xfcb       db      0               ;make & search xfcb flag
4382  =0814 00                find_xfcb       db      0               ;search find xfcb flag
4383  =0815 0000              xdcnt           dw      0               ;empty directory dcnt
4384  =0817 00                DIR_CNT         DB      0               ;DIRECT I/O COUNT
4385  =0818 00                MULT_NUM        DB      0               ;MULTI-SECTOR COUNT used in bdos
4386  =0819 00                olap_type       db      0               ;record lock overlap type
4387  =081A 00                ff_flag         db      0               ;0ffh xios error return flag
4388  =081B 00                err_type        db      0               ;type of error to print if not zero
4389  =081C                   zerolength      rb      4               ;reserved bytes
4390  =  001b                 zerolength      equ     (offset $)-(offset fcbdsk)
4391  =0820 01                mult_sec        db      1               ;multi sector count passed to xios
4392  =0821 00                RELOG           DB      0               ;AUTOMATIC RELOG SWITCH
4393  =
4394  =0822 00                BLK_OFF         DB      0               ;RECORD OFFSET WITHIN BLOCK
4395  =0823 0000              blk_num         dw      0               ;block number
4396  =                       ;
4397  =0825 2703              ua_lroot        dw      offset ua0      ;unallocated block list root
```

```
4398
4399  =                       ;
4400  =0827 20030000          ua0     dw      offset ua1,0
4401  =082B FF00                      db      0ffh,0
4402  =082D 30030000          ua1     dw      offset ua2,0
4403  =0831 FF00                      db      0ffh,0
4404  =0833 39030000          ua2     dw      offset ua3,0
4405  =0837 FF00                      db      0ffh,0
4406  =0839 3F030000          ua3     dw      offset ua4,0
4407  =083D FF00                      db      0ffh,0
4408  =083F 45030000          ua4     dw      offset ua5,0
4409  =0843 FF00                      db      0ffh,0
4410  =0845 00000000          ua5     dw      0,0
4411  =0849 FF00                      db      0ffh,0
4412  =
4413  =084B                   HASH            RB      4               ;HASH CODE WORK AREA
4414  =084F 00                HASHL           DB      0               ;HASH SEARCH LENGTH
4415  =
4416  =0850 03                LOG_FXS         db      11              ;number of log_fxs entrys
4417  =                                       ;       fxi15,fxi17,fxi19,fxi22,fxi23
4418  =0851 020406090A                        db      02h  ,04h  ,06h  ,09h  ,0ah
4419  =                                       ;       fxi30,fxi35,fxi99,fxi100,fxi102,fxi103
4420  =0856 111625262829                      db      11h  ,16h  ,25h  ,26h   ,28h   ,29h
4421  =085C 00000000                          db      0,0,0,0         ;reserved
4422  =0860 07                rw_fxs          db      7               ;number of rw_fxs entrys
4423  =                                       ;       fxi20,fxi21,fxi33,fxi34,fxi40,fxi42,fxi43
4424  =0861 070814151B1C                      db      07h  ,08h  ,14h  ,15h  ,1bh  ,1ch  ,1dh
4425        1D
4426  =0868 0000                              db      0,0             ;reserved
4427  =086A 02                sc_fxs          db      2               ;number of sc_fxs entrys
4428  =                                       ;       fxi16,fxi18
4429  =086B 0305                              db      03h  ,05h
4430  =086D 0000                              db      0,0             ;reserved
4431  =
4432  =086F 00                DEBLOCK_FX      DB      0               ;DEBLOCK FUNCTION #
4433  =0870 00                PHY_OFF         DB      0               ;RECORD OFFSET WITHIN PHYSICAL RECORD
4434  =0871 0000              CUR_BCBA        DW      0               ;CURRENT BCB OFFSET
4435  =0873 0000              ROOT_BCBA       DW      0               ;ROOT BCB OFFSET
4436  =0875 0000              EMPTY_BCBA      DW      0               ;EMPTY BCB OFFSET
4437  =
4438  =                       if BNPN
4439  =
4440  =0877 0000              P_LAST_BCBA     DW      0               ;PROCESS'S LAST BCB OFFSET
4441  =0879 00                P_BCB_CNT       DB      0               ;PROCESS BCB COUNT IN BCB LIST
4442  =
4443  =                       endif
4444  =
4445  =087A 00                fx_intrn        db      0               ;internal BDOS function number
4446  =
4447  =087B 0000              TRACK           DW      0       ;BCB RECORD'S TRACK
4448  =087D 0000              SECTOR          DW      0       ;BCB RECORD'S SECTOR
4449  =
4450  =                       ;       seldsk,usrcode are initialized as a pair
```

```
4451
4452  =087F 00              seldsk          db      0       ;selected disk num
4453  =0880 00              usrcode         db      0       ;curr user num
4454  =
4455  =0881 0000            info            dw      0       ;info adr
4456  =0883 0000            searcha         dw      0       ;search adr
4457  =
4458  =                     ;the following variable order is critical
4459  =
4460  =                     ;variables copied from uda for mp/m                     x
4461  =
4462  =                     ;variables included in fcb checksum for mp/m and cp/m     x
4463  =
4464  =                     ;variables used to access system lock list for mp/m         x
4465  =
4466  =0885 0000            dma_ofst        dw      0       ;dma offset             1
4467  =0887 0000            dma_seg         dw      0       ;dma segment            2
4468  =0889 00              func            db      0       ;bdos function #        3
4469  =088A 00              searchl         db      0       ;search len             4
4470  =
4471  =                     if BMPM
4472  =
4473  =088B 0000            searchaofst     dw      0       ;search adr ofst        5
4474  =088D 0000            searchabase     dw      0       ;search adr base        6
4475  =
4476  =                     endif
4477  =
4478  =088F 0000            dcnt            dw      0       ;directory counter      7
4479  =0891 0000            dblk            dw      0       ;directory block        8 ?? - not used - ??
4480  =0893 00              error_mode      db      0       ;bdos error mode        9
4481  =0894 00              mult_cnt        db      0       ;bdos multi-sector cnt 10
4482  =0895                 df_password     rb      8       ;process default pw    11
4483  =
4484  =                     if BMPM
4485  =
4486  =089D 00              pd_cnt          db      0       ;bdos process cnt      12 1
4487  =
4488  =                     endif
4489  =
4490  =089E 00              high_ext        db      0       ;fcb high extent bits     2
4491  =089F 00              xfcb_read_only  db      0       ;xfcb read only flag      3
4492  =08A0 FF              curdsk          db      0ffh    ;current disk             4 1
4493  =
4494  =                     if BMPM
4495  =
4496  =08A1 00              packed_dcnt     db      0       ;packed dblk+dcnt           2
4497  =08A2 00                              db      0
4498  =08A3 00                              db      0
4499  =08A4 0000            pdaddr          dw      0       ;process descriptor addr    3
4500  =
4501  =                     endif
4502  =
4503  =                             org ((offset $) + 1) and 0fffeh
```

```
4504
4505  =
4506  =                         ;       curtrka - alloca are set upon disk select
4507  =                         ;       (data must be adjacent)
4508  =
4509  =03A6 0000               cdrmaxa         dw      0       ;ptr to cur dir max val
4510  =03A8 0000               drvlbla         dw      0       ;drive label data byte addr
4511  =03AA 0000               buffa           dw      0       ;ptr to dir dma addr
4512  =03AC 0000               dpbaddr         dw      0       ;curr disk param block addr
4513  =03AE 0000               checka          dw      0       ;curr checksum vector addr
4514  =03B0 0000               alloca          dw      0       ;curr alloc vector addr
4515  =03B2 0000               DIR_BCBA        DW      0       ;DIRECTORY BUFFER CONTROL BLOCK ADDR
4516  =03B4 0000               DAT_BCBA        dw      0       ;DATA BUFFER CONTROL BLOCK ADDR
4517  =03B6 0000               HASH_SEG        dw      0       ;HASH TABLE SEGMENT
4518  =  000C                  addlist         equ     12      ;"$-buffa" = addr list size
4519  =
4520  =                         ;       sectpt - offset obtained from disk parm block at dpbaddr
4521  =                         ;       (data must be adjacent)
4522  =
4523  =03B8 0000               sectpt          dw      0       ;sectors per track
4524  =03BA 00                 blkshf          db      0       ;block shift factor
4525  =03BB 00                 blkmsk          db      0       ;block mask
4526  =03BC 00                 extmsk          db      0       ;extent mask
4527  =03BD 0000               maxall          dw      0       ;max alloc num
4528  =03BF 0000               dirmax          dw      0       ;max dir num
4529  =03C1 0000               dirblk          dw      0       ;reserved alloc bits for dir
4530  =03C3 0000               chksiz          dw      0       ;size of checksum vector
4531  =03C5 0000               offsetv         dw      0       ;offset tracks at beginning
4532  =03C7 00                 PHYSHF          DB      0       ;PHYSICAL RECORD SHIFT FACTOR
4533  =03C8 00                 PHYMSK          DB      0       ;PHYSICAL RECORD MASK
4534  =03C9                    endlist         rs      0       ;end of list
4535  =  0011                  dpblist         equ     (offset endlist)-(offset sectpt)
4536  =                                                        ;size
4537  =
4538  =                         ;       local variables
4539  =
4540  =03C9                    common_dma      rb      16      ;copy of user's dma 1st 16 bytes
4541  =03D9 00                 make_flag       db      0       ;make function flag
4542  =03DA 00                 actual_rc       db      0       ;directory ext record count
4543  =03DB 00                 save_xfcb       db      0       ;search xfcb save flag
4544  =03DC 00                 save_mod        db      0       ;open_reel module save field
4545  =03DD 00                 pw_mode         db      0       ;password mode
4546  =03DE 00                 attributes      db      0       ;fcb interface attributes hold byte
4547  =
4548  =                         if BNPH
4549  =
4550  =                         ;       number of lock list items required for lock operation
4551  =03DF 010002020101       required_table  db      1,0,2,2,1,1,2,2,1,1,2,2,1,1,2,2
4552        020201010202
4553        01010202
4554  =
4555  =03EF 00                 chk_olist_flag  db      0       ;check | test olist flag
4556  =03F0 0000               lock_sp         dw      0       ;lock stack ptr
```

```
4557
4558  =08F2 00              lock_shell      db      0       ;lock shell flag
4559  =08F3 00              check_fcb_ret   db      0       ;check_fcb return switch
4560  =08F4 00              lock_unlock     db      0       ;lock | unlock function flag
4561  =08F5 00              incr_pdcnt      db      0       ;increment process_cnt flag ??
4562  =08F6 00              free_mode       db      0       ;free lock list entries flag ??
4563  =                                                     ;1=free entries for curdsk
4564  =                                                     ;0=free all entries
4565  =08F7 0000            cur_pos         dw      0       ;current position in lock list
4566  =08F9 0000            prv_pos         dw      0       ;previous position in lock list
4567  =
4568  =08FB 00              dont_close      db      0       ;inhibit actual close flag
4569  =08FC 00              open_cnt        db      0       ;process open file count
4570  =08FD 00              lock_cnt        db      0       ;process locked record count
4571  =08FE 0000            file_id         dw      0       ;address of file' lock list entry
4572  =0900 00              set_ro_flag     db      0       ;set drive r/o flag
4573  =0901 00              check_disk      db      0       ;disk reset open file check flag
4574  =0902 00              flushed         db      0       ;lock list open file flush flag
4575  =
4576  =                     ;free_root, lock_max, open_max initialized by sysgen
4577  =
4578  =0903 5200                            dw      offset free_root
4579  =0905 0000            open_root       dw      0       ;lock list open file list root
4580  =0907 0000            lock_root       dw      0       ;lock list locked record list root
4581  =
4582  =                     endif
4583  =
4584  =0909 0000            sdcnt           dw      0       ;saved dcnt of file's 1st fcb
4585  =090B 0000            sdcnt0          dw      0       ;saved dcnt (user 0 pass)
4586  =
4587  =                     if BCPM
4588  =
4589  =                     chain_flag      db      0       ;chain flag ??
4590  =                     tod             db      0ffh,0ffh,0ffh,0ffh ??
4591  =
4592  =                     endif
4593  =
4594  =
4595  =090D 00              rmf             db      0       ;read mode flag for open$reel
4596  =090E 00              wflag           db      0       ;xios/bios write flag
4597  =090F 00              dirloc          db      0       ;directory flag in rename, etc.
4598  =0910 00              linfo           db      0       ;low(info)
4599  =0911 00              dminx           db      0       ;local for diskwrite
4600  =0912 00              single          db      0       ;set true if single byte
4601  =                                                     ;alloc map
4602  =0913 00              rcount          db      0       ;record count in curr fcb
4603  =0914 00              extval          db      0       ;extent num and extmsk
4604  =0915 00              vrecord         db      0       ;curr virtual record
4605  =0916 00              ADRIVE          DB      0       ;CURRENT DISK - must preceed arecord
4606  =0917 0000            arecord         dw      0       ;curr actual record
4607  =0919 00                              db      0       ;curr actual record high byte
4608  =091A 0000            ARECORD1        dw      0       ;curr actual block# * blkmsk
4609  =091C 0000            drec            dw      0       ;curr actual directory record
```

```
4610
4611  =091E 0000            CUR_dma_seg     DW      0
4612  =0920 0000            CUR_DMA         DW      0
4613  =
4614  =                     ;       local variables for directory access
4615  =
4616  =0922 00              dptr            db      0       ;directory pointer 0,1,2,3
4617  =  003F               ldcnt           equ     byte ptr dcnt   ;low(dcnt)
4618  =0923 00              user_zero_pass  db      0       ;search user zero flag
4619  =
4620  =                     ;       shell variables
4621  =
4622  =0924 0000            shell_si        dw      0       ;bdos command offset
4623  =0926 000000          shell_rr        db      0,0,0   ;r0,r1,r2 save area
4624  =
4625  =                     ;       special 8086 variables:
4626  =
4627  =0929 0000            uda_save        dw      0       ;user data area saved value
4628  =092B 0000            parametersegment dw     0       ;user parameter segment
4629  =092D 0000            returnseg       dw      0       ;user return segment
4630  =
4631  =                     ;       error messages
4632  =
4633  =092F 00              err_drv         db      0
4634  =0930 0000            err_pd_addr     dw      0
4635  =
4636  =0932 0D0A43502F4D    dskmsg          db      13,10,'CP/M Error On '
4637        204572726F72
4638        204F6E20
4639  =0942 203A2000        dskerr          db      ' : ',0
4640  =
4641  =0946 000060096909    xerr_list       dw      0,permsg,rodmsg,rofmsg,selmsg
4642        78098709
4643  =0950 B309C7090C09              dw      xe5,xe6,xe7,xe8,xe9,xe10,xe11,xe3
4644        EB09FF09100A
4645        290A9509
4646  =
4647  =0960 4469736B2049    permsg          db      'Disk I/O',0
4648        2F4F00
4649  =0969 526561642F4F    rodmsg          db      'Read/Only Disk',0
4650        6E6C79204469
4651        736300
4652  =0978 526561642F4F    rofmsg          db      'Read/Only File',0
4653        6E6C79204669
4654        6C6500
4655  =0987 496E76616C69    selmsg          db      'Invalid Drive',0
4656        642044726976
4657        6500
4658  =
4659  =0995 46696C65204F    xe3             db      'File Opened in Read/Only Mode',0
4660        70656E656420
4661        696E20526561
4662        642F4F6C6C79
```

```
4663
4664            20436F646500
4665    =
4666    =09B5   46696C652043    xe5             db      'File Currently Open',0
4667            757272656E74
4668            6C79204F7065
4669            6E00
4670    =09C7   436C6F736520    xe6             db      'Close Checksum Error',0
4671            43686563756B
4672            756D20457272
4673            6F7200
4674    =09DC   50617373776F    xe7             db      'Password Error',0
4675            726420457272
4676            6F7200
4677    =09EB   46696C652041    xe8             db      'File Already Exists',0
4678            6C7265616479
4679            204578697374
4680            7300
4681    =09FF   496C6C656761    xe9             db      'Illegal ? in FCB',0
4682            6C203F20696E
4683            20464342 00
4684    =0A10   4F70656E2046    xe10            db      'Open File Limit Exceeded',0
4685            696C65204C69
4686            6D6974204578
4687            636565646564
4688            00
4689    =0A29   4E6F20526F6F    xe11            db      'No Room in System Lock List',0
4690            6D20696E2053
4691            797374656D20
4692            4C6F636B204C
4693            69737400
4694    =
4695    =0A45   0D0A00          crlf_str        db      13,10,0
4696    =0A48   0D0A42646F73    pr_fx           db      13,10,'Bdos Function = '
4697            2046756E6374
4698            696F6E203D20
4699    =0A5A   202020          pr_fx1          db      '   '
4700    =0A5D   2046696C6520    pr_fcb          db      ' File = '
4701            3D20
4702    =0A65                   pr_fcb1         rs      12
4703    =0A71   00                              db      0
4704    =
4705    =0A72   0D0A44697368    deniedmsg       db      13,10,'Disk reset denied, Drive '
4706            207265736574
4707            2064656E6965
4708            642C20447269
4709            766520
4710    =0A8D   003A            denieddrv       db      0,':'
4711    =0A8F   20436F6E736F                    db      ' Console '
4712            6C6520
4713    =0A98   00              deniedcns       db      0
4714    =0A99   2050726F6772                    db      ' Program '
4715            616D20
```

```
4716
4717  =0AA2 313233343536    deniedprc       db      '12345678',0
4718        373800
4719  =
4720  =                     if BNPM
4721  =
4722  =                     ;       bdos stack switch variables and stack
4723  =                     ;       used for all bdos disk functions
4724  =
4725  =                             org     ((offset $) + 1) and 0fffeh
4726  =
4727  =                             ; 69 word bdos stack
4728  =
4729  =0AAC CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4730  =0AB2 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4731  =0AB8 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4732  =0ABE CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4733  =0AC4 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4734  =0ACA CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4735  =0AD0 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4736  =0AD6 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4737  =0ADC CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4738  =0AE2 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4739  =0AE8 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4740  =0AEE CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4741  =0AF4 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4742  =0AFA CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4743  =0B00 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4744  =0B06 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4745  =0B0C CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4746  =0B12 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4747  =0B18 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4748  =0B1E CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4749  =0B24 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4750  =0B2A CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4751  =0B30 CCCCCCCCCCCC            dw      0cccch,0cccch,0cccch
4752  =0B36                 bdosstack       rw      0
4753  =
4754  =0B36                 save_sp         rw      1
4755  =0B38                 ss_save         rw      1
4756  =0B3A                 sp_save         rw      1
4757  =
4758  =                     ;       local buffer area:
4759  =
4760  =0B3C                 info_fcb        rb      40      ;local user FCB
4761  =0B64                 save_fcb        rb      16      ;fcb save area for xfcb search
4762  =
4763  =0B74 0000            mxdiskqd        dw      0       ;link
4764  =0B76 0000                            db      0,0     ;net,org
4765  =0B78 0300                            dw      qf_keep+qf_mx   ;flags (MX queue)
4766  =0B7A 4D5864697368    db      'MXdisk  '
4767        2020
4768  =0B82 00000100                        dw      0,1     ;msglen,nmsgs
```

```
4769
4770  =0886 00000000                    dw      0,0     ;nq,dq
4771  =088A 01000000                    dw      1,0     ;msgcnt,out
4772  =088E 0000                        dw      0       ;buffer ptr
4773  =
4774  =0890 00                 mxdiskqpb  db      0       ;flgs
4775  =0891 00                           db      0       ;net
4776  =0892 7402                         dw      mxdiskqd        ;qaddr
4777  =0894 0100                         dw      1       ;nmsgs
4778  =0896 0000                         dw      0       ;buffer
4779  =0898 40>864697368                 db      'MXdisk  '
4780        2020
4781  =
4782  =                         ;       Error Message sync param block for mutual exclusion
4783  =
4784  =08A0 0000                msg_spb  dw      0       ;link   Error Message sync parameter block
4785  =08A2 0000                         dw      0       ;owner
4786  =08A4 0000                         dw      0       ;wait
4787  =08A6 0000                         dw      0       ;next
4788  =                         endif
4789  =
4790  =                         if BCPM
4791  =
4792  =                         ;
4793  =                         ;       special 8086 variables:
4794  =                         ;
4795  =                         ioloc           db      0       ;iobyte
4796  =                         user_parm_seg   dw      0       ;holds user parameter seg during load
4797  =                         nallocmem       db      0       ;no. of allocated memory segments
4798  =                         ncrmem          db      0       ;no. of available memory segments
4799  =                         crmem           dw      0,0     ;memory table (16 elements)
4800  =                                         dw      0,0,0,0,0,0,0,0
4801  =                                         dw      0,0,0,0,0,0,0,0
4802  =                                         dw      0,0,0,0,0,0,0,0
4803  =                                         dw      0,0,0,0,0,0,0,0
4804  =                         ;
4805  =                         mem_stack_length        equ     40
4806  =                         memstack        rs      mem_stack_length
4807  =                                         ;8 possible allocations
4808  =                         stbase  equ     word ptr 0
4809  =                         stlen   equ     word ptr 2
4810  =                         ccpflag equ     byte ptr 4
4811  =                         nccpalloc       db      0       ;number of current ccp allocations
4812  =                         mem_stk_ptr     dw      0       ;current memory stack location
4813  =
4814  =                         stackarea       rw      ssize   ;stack size
4815  =                         endstack        rb      0       ;top of stack
4816  =                         ;
4817  =                         endif
4818  =                                         org     0fffh
4819  =                         if CCPMOFF
4820  =                                         org     0bfffh
4821  =                         endif
```

```
4822
4823  =0FF 00                           db      0
4824
4825                          if ccpm
```

```
4826
4827  =                         eject 1 include xiosdat.fmt
4828  =                       ;*************************************************
4829  =                       ;*
4830  =                       ;*      Concurrent CP/M XIOS Data Area
4831  =                       ;*
4832  =                       ;*************************************************
4833  =
4834  =                         DSEG
4835  =
4836  =                       org     0C0CH
4837  =
4838  =0C0C                   tick            rb      1
4839  =                                                   ;see XIOS for format of the rest of the
4840  =                                                   ;XIOS header, most of the variables
4841  =                                                   ;are copied to the System Data Segment
4842  =                                                   ;by GENSYS.
4843
4844                          endif
```

```
4845
4846                                    eject 1 end
4847
4848
4849  END OF ASSEMBLY.  NUMBER OF ERRORS:  0.  USE FACTOR: 71%
```

```
BPBPW2PTR        0054 V  2078  3581#
BPBSLEN          0012 V  3572#
BPBSSEG          0015 V  3573#
BPBSXXX          0017 V  3574#
BPBUDATA         0100 V  3598#
BUFFA            03AA V  4511#
BVERENT          0270 L  1382  1418#
BVERNUM          007A V  1420  3854# 3859#
BXSAV            0626 V  4227#
CATTACH          0000 N   913#  914
CBIOSENT         027E L  1383  1432# 1466#
CBTMP            000C N   923#  924
CCB              0054 V  3823#
CCBLEN           002C N   939#
CCOLUMN          0006 N   917#  918
CCOSLEEP         0022 N   936#  937
CCPM             FFFF N    66#  255   846  1463  2572  2699  3858  4825
CCPMOFF          FFFF N  4301# 4355  4819
CDRMAXA          08A6 V  4509#
CFBUFP           0020 N   948#
CFCOMPC          0002 N   944#
CFCONOUT         0008 N   946#
CFLAG            0004 N   915#  916
CFLTSTOP         0001 N   943#
CFSWITCHS        0004 N   945#
CFVOUT           0010 N   947#
CHATMENT         0337 L  1392  1585#
CHAL             0B3D L  2550  2560# 2583#
CHBASE           0003 N   802#  803  2527
CHDOIT           0AE6 L  2522  2524#
CHECKA           08AE V  4513#
CHECKDISK        0901 V  4573#
CHECKFCBRET      08F3 V  4559#
CHENTMAX         0008 N   807# 2520
CHFTXREC         0070 N   811# 2516
CHFORM           0000 N   800#  801  2522  2524  2541  2543  2574  2575  2577
CHKOLISTFLAG     08EF V  4555#
CHKSIZ           08C3 V  4530#
CHLBYTE          007F N   810# 2514
CHLEN            0009 N   805# 2597
CHLENGTH         0001 N   801#  802  2525
CHMAX            0007 N   804#  805  2529
CHMIN            0005 N   803#  804  2528  2531
CHMORE           0ADE L  2522# 2537
CHNEXT           0B67 L  2523  2597#
CHNXT            0660 L  2552  2558  2565  2582  2588  2595#
CIO              0004 N    89#  224   225   226   227   228  3924  3925  3928  3929
                       3932  3933  3934  4020  4021  4022  4023  4046  4047  4048
                       4049  4053  4061  4062  4063  4064  4065  4067
CIOMOD           0018 N  1048  3772#
CIOMODBIT        0010 N   100# 1046
CLIACB           045B V  1751  1780  2111  2326  4131#
CLIBD            05A6 L  1862  1864  1865  1875  1890  1900#
```

```
CL1BDE          05AE L  1887  1896  1901#
CL1BPAGE        046F V  1985  1991  2000  2021  2045  2070  4141#
CL1CCCMD        0001 N   724#  726  1701  1703
CL1CCLEN        0002 N   726# 1703
CL1CCNET        0000 N   723#  724
CL1CHAIN        0323 V  1645  1647  1680  1728  1942  1971  2088  2165  4113#
CL1CHECKQUE     0444 L  1725  1737#
CL1CL           0667 L  1911  1975#
CL1CLT          0364 L  1646  1649#
CL1CMDTAIL      03A8 V  1686  1702  1993  2038  2050  2075  2215  2218  2223  4120#
                        4136
CL1CNS          0472 V  1669  1781  1827  2112  4146#
CL1CPY          03E3 L  1680  1693#
CL1CUSPORB      044B V  1744  1756  1763  1771  2285  4125#
CL1DFTL         0476 V  1652  1658  1777  1963  4150#
CL1DMA          0325 V  1837  1841  4115#
CL1DMAOFST      031D V  1673  1685  2161  4110#
CL1DMASEG       031F V  1675  1687  2163  4111#
CL1DSK          0474 V  1665  1820  1864  2154  4148#
CL1ENT          0339 L  1385  1591#
CL1ERR          046D V  1676  1710  1734  1798  1901  1982  2116  2178  4140#
CL1ERRMODE      0475 V  1671  2158  4149#
CL1EXIT         07E8 L  1711  1731  1800  2110  2118  2152#
CL1EXIT2        0436 L  1728  1731#
CL1FCB          042A V  1721  1723  1743  1815  1836  1846  1865  1873  1879  1891
                        1916  1969  2047  2072  2258  2261  2264  2276  2306  2310
                        4123# 4137
CL1FFLOAD       0439 L  1721  1724  1725  1732#
CL1FLOAD        04AF L  1733  1786#
CL1GETAT        087C L  2198  2201#
CL1GLOAD        0685 L  1979  1984#
CL1GOODO        046C L  1757  1762#
CL1GOPEN        0504 L  1855  1890  1893  1903#
CL1GPD          04C4 L  1797  1801#
CL1GPRS         0412 L  1709  1712#
CL1GW           047E L  1765  1771#
CL1KUDA         0641 L  1942  1962#
CL1LD           0664 L  1971  1974#
CL1LD1          05C8 L  1908  1912#
CL1LDDSK        0471 V  4142#
CL1LNAB         067E L  1980  1982#
CL1NCTL         0373 L  2186  2194#
CL1NET          03A5 V  1698  4118#
CL1PARSE        0406 L  1691  1708#
CL1PCB          0467 V  2062  2063  2065  2323  4136#
CL1PD           0468 V  1803  1812  1915  1944  1968  2093  2108  2113  2124  2167
                        4139#
CL1PFLAG        0321 V  1640  2177  4112#
CL1PPD          03A6 V  1663  4119# 4127
CL1QFUL         042A L  1727# 1734
CL1QW           048E L  1772  1777#
CL1RLS          083C L  2165  2171  2176#
CL1RMPD         07B6 L  1902  1981  1985  2122# 2141
```

```
CLIRMPD2          0581 L  1900  1902#
CLISPB            0651 V  2320  4259  4267#
CLISYNC           09A4 L  1638  2315#
CLITERM           0324 V  1648  1730  1973  2171  4114#
CLIUNSYNC         09A9 L  2179  2316#
CLIUSER           0473 V  1667  1821  2156  2267  4147#
CLUPDENT          09D5 L  1594  2381#
CMAXBUFS17        0010 N   926#  927
CMIMIC            0008 N   919#  920
CMOD              0090 V  1913  3883#
CMSOURCE          0009 N   920#  921
CNCHAR            0007 N   918#  919
CNEXT50           027E L  1476#
COMMONDMA         08C9 V  4540#
COMPFLBCKS        0811 V  4379#
CONASM            098A L  1784  2115  2326#
CORET             0142 L  1173#
CPBNAME           0000 N  3432# 3433  3470  3471  3472  3473
CPBPARAM          0008 N  3433# 3474
CPC               060A N   921#  922
CPRNIT            0985 L  2294  2303#
CQBUFF            0020 N   935#  936
CQPBUFFPTR        001E N   934#  935
CQPBFILL          0019 N   931#  932
CQPBFLAGS         0018 N   930#  931
CQPBNMSGS         001C N   933#  934
CQPBQADDR         001A N   932#  933
CQUEUE            0002 N   914#  915
CR                000D N   835# 3359  3398
CRLF              089E L  1966  2229# 2287
CRLFSTR           0A45 V  4695#
CRSVD             000D N   924#  925
CS                SREG V  1034  1060  1279  1572  2293  2947  3386  3466
CSNABORT          0020 N   958#
CSNBACKGROUND     0002 N   954#
CSNBUFFERED       0001 N   952#
CSNCIRLD          0100 N   961#
CSNCIRLP          0200 N   962#
CSNCIRLS          0080 N   960#
CSNFILEFULL       0040 N   959#
CSNNOSWITCH       0008 N   956#
CSNPURGING        0004 N   955#
CSNSUSPEND        0010 N   957#
CSTATE            000E N   925#  926
CSTRICOL          0005 N   916#  917
CTDONE            0759 L  2038  2041  2061  2066  2084#
CTL               005E N   843#
CTLC              0003 N   828#
CTLD              0004 N   829#
CTLE              0005 N   830#
CTLH              0008 N   832#
CTLP              0010 N   836#
CTLQ              0011 N   837#
```

```
CTLR            001C N   838#
CTLS            0013 N   839#
CTLU            0015 N   840#
CTLX            0018 N   841#
CTLZ            001A N   842#
CURBCBA         0871 V  4434#
CURDMA          0920 V  4612#
CURDMASEG       091E V  4611#
CURDSK          08A0 V  4492#
CURPOS          08F7 V  4565#
CURRMAU         0632 V  4236#
CURRMPB         0636 V  4238#
CURRST          0634 V  4237#
CUSLEEP         0024 N   937#  938
CVC             000B N   922#  923
CVCMXQ          0016 N   929#  930
CVINQ           0012 N   927#  928
CVOUTQ          0014 N   928#  929
CVSLEEP         0026 N   938#  939
DATBCBA         0834 V  4516#
DAYFILE         004F V  1653  3816#
DBLK            0891 V  4479#
DCNT            088F V  4478#  4617
DEL             113D L  3304  3355  3403#  3411
DEBLOCKFX       086F V  4432#
DEBUGINT        00E1 N    55#
DELIM           10FC L  3307  3317  3328  3336  3352  3356  3365  3376#
DELTAS          112F V  3384  3398#  3401
DELNOUP         1128 L  3393  3395#
DELNXT          1104 L  3385#  3388
DELREI          112D L  3386  3392  3396#
DELUP           1121 L  3389  3392#
DENIEDCNS       0A98 V  4713#
DENIEDDRV       0A8D V  4710#
DENIEDMSG       0A72 V  4705#
DENIEDPRC       0AA2 V  4717#
DEVVER          000C V   988#
DFPASSWORD      0895 V  4482#
DIRBCBA         0882 V  4515#
DIRBLK          08C1 V  4529#
DIRCNT          0817 V  4384#
DIRLOC          090F V  4597#
DIRMAX          088F V  4528#
DISPATCHER      0038 N  3789#
DLOG            0808 V  4367#
DLR             006A V  3841#
DMADEST         0885 V  4466#
DMASEG          0887 V  4467#
DMINX           0911 V  4599#
DONTCLOSE       08F8 V  4568#
DPBADDR         08AC V  4512#
DPBLIST         0011 N  4535#
DPTR            0922 V  4616#
```

```
DRD                0F77 L  2441  2757  3007  3030  3042#
DRDLP              0F81 L  3055# 3101
DRDR               0F8C L  3058  3063#
DREC               091C V  4609#
DRL                006C V  3842#
DRR1               0FEE L  3095  3099#
DRR2               0FED L  3073  3077#
DRR5               0FF7 L  3099  3101#
DRVLOLA            0848 V  4510#
DS                 SREG V  1012  1024  1031  1032  1035  1052  1053  1058  1058  1061
                           1083  1088  1090  1136  1137  1139  1140  1144  1151  1152
                           1154  1155  1159  1160  1164  1165  1166  1176  1177  1179
                           1180  1183  1184  1187  1189  1334  1337  1340  1436  1436
                           1438  1479  1480  1483  1552  1572  1572  1574  1682  1687
                           1690  1694  1695  1705  1745  1805  1816  1844  1952  1957
                           2000  2000  2015  2031  2031  2034  2131  2137  2140  2219
                           2292  2293  2294  2426  2426  2427  2429  2638  2642  2647
                           2726  2728  2731  2755  2877  2877  2879  2925  2925  2931
                           2943  2948  3189  3201  3203  3207  3242  3247  3254  3300
                           3300  3330  3361  3463  3464  3480  3486  3486  3487  3494
                           3498  3503
DSKERR             0942 V  4639#
DSKMSG             0932 V  4636#
DSKRCL             0080 N    48# 2793  4107  4115
DSPTCHIOS          0622 V  4221#
EABORT             0028 N   491# 1910  1980  2191  3097
EACTIVEPD          0022 N   485#
EBADENTRY          0002 N   453# 1414  1445  1523
EBADFNAME          0018 N   475# 3305  3315  3390
EBADFTYPE          0019 N   476# 3332
EBADLOAD           001C N   479#
EBADOPEN           001E N   481# 1901
EBADREAD           001D N   480# 3074
EFIXUPREC          0029 N   492# 2820
EFLAGOVRRUN        0005 N   456#
EFLAGUNDERRUN      0006 N   457#
EFNETWORK          00F0 N   626# 1251  1252
EILLCMS            0013 N   470#
EILLDISK           0017 N   474# 3311
EILLFLAG           0004 N   455#
EILLLST            0025 N   488#
EILLMD             001B N   478#
EILLPASSWD         0026 N   489# 3342
EMPTYSCBA          0875 V  4436#
ENCLIP             0016 N   473#
ENCLIQ             0010 N   467#
ENDLIST            08C9 V  4534# 4535
ENDSEG             0044 V  3795#
ENDTAIL            0668 L  2004  2015#
ENOATTACH          0024 N   487#
ENOCHAR            001A N   477#
ENOCNSMATCH        0015 N   472#
ENOCSEG            0021 N   484# 2838  2931
```

```
ENOMEMORY          0003 N   454# 2567  2590
ENOMIMIC           0027 N   490#
ENOPD              000C N   463# 1796
ENOPDNAME          0014 N   471#
ENOQEUE            0008 N   459#
ENOSD              0007 N   458#
ENOSEGDESC         0009 N   460# 1767  3504
ENOTIMPLEMENTED    0001 N   452# 1408
ENOTOWNER          0020 N   485#
ENOUSD             0012 N   469#
ENTLEN             0002 N   622#
ENTRY              01E8 L   980   984  1282#
ENTRYADENTRY       0000 N   621#  622  1248
ENULLCMD           001F N   482#
EPDNOTERM          0023 N   486#
EQEMPTY            000E N   465#
EQFULL             000F N   466# 1733  1773
EQINUSE            000A N   461#
EQPROTECTED        0000 N   464#
ERRORV             092F V  4633#
ERRINTERCEPT       0092 V  3887#
ERRORMODE          0893 V  4480#
ERRPDADDR          0930 V  4634#
ERRTYPE            0818 V  4388#
ES                 SREG V  1023  1052  1053  1085  1086  1088  1140  1165  1175  1177
                           1560  1560  1564  1570  1570  1574  1682  1684  1690  1694
                           1695  1705  1705  1745  1745  1752  1805  1805  1808  1816
                           1816  1818  1838  1838  1840  1945  1950  1950  1956  1991
                           1991  1995  2021  2021  2031  2034  2045  2045  2053  2056
                           2057  2070  2070  2078  2080  2083  2131  2140  2219  2219
                           2220  2313  2313  2426  2427  2429  2470  2470  2471  2784
                           2785  2789  2790  2841  2841  2842  2845  2859  2866  2867
                           2870  2877  2882  2883  2897  2897  2902  2919  2919  2931
                           2948  2989  2989  2990  3070  3070  3189  3189  3191  3301
                           3301  3330  3361  3463  3495  3497  3498  3503
ESSAV              0624 V  4226#
EVENDIV            0C59 L  2678  2680#
EXTMSK             088C V  4526#
EXTVAL             0914 V  4603#
FALSE              0000 N    46#   65    68  1645  1648  1652  1782  2415  3884  4223
FBUOSTERM          052D N   233#
FCALLMIOS          0032 N   148#
FCBCR              0020 N   596#
FCBDR              0000 N   588# 2455  3315
FCBDSK             0808 V  4373# 4390
FCBT               1148 L  3304  3413#
FCBLEN             0020 N    49# 2425  4123
FCBNAME            0001 N   589# 1743  1815  1920  1924  1931  1938  2258  2265  2311
                           2461  3324  3325
FCBPLEN            001A N   594# 1724  2055  2079  3348
FCBPTR             0018 N   593# 2048  2073  3345  3350
FCBPWD             0010 N   592# 1836  3343
FCBPW              0021 N   597# 2435  3068
```

```
FCBTYPE          0009 N   590# 1847  1848  1849  1880  1892  2261  3333
FCCONATTCH       00A2 N   193# 1655
FCLOSTAT         0418 N   226# 1500  3088
FCTDTERM         0417 N   225#
FCLOAD           010E N   200#
FCONASSIGN       0095 N   180# 2326
FCONATTACH       0092 N   176# 1078
FCONDETACH       0093 N   178#
FCONOUT          0002 N   107# 2237
FCONPRINT        040E N   224# 2303
FCONREAD         000A N   115#
FCONWRITE        0009 N   114#
FCQREAD          008A N   168#
FCQWRITE         008C N   170# 1771
FCREATLPROC      0090 N   174# 1092  2095
FDELIM           0062 N   158#
FDISPATCH        008E N   172#
FFCLOSE          0010 N   121# 2306
FFFLAG           031A V  4387#
FFINDPDNAME      0214 N   207# 3125
FFLAGSET         0085 N   163# 1316
FFLAGWAIT        0084 N   162#
FFOPEN           000F N   120# 2309
FFREADRDM        0021 N   139# 3069
FFREADSEQ        0014 N   126#
FFRECALL         003A N   156# 2497
FFRECREM         0039 N   155#
FGETDEFDISK      0019 N   131#
FILEID           08FE V  4571#
FINDXFCB         0314 V  4382#
FINFLAGSET       0203 N   204# 1317
FIXGRP           0000 N   815#  816  2770  2777
FIXLEN           0004 N   818# 2792
FIXOFFS          0003 N   817#  818  2787
FIXPARA          0001 N   816#  817  2782
FLAGMIN          0003 N    62#
FLAGS            0056 V  3824#
FLAGSET          0002 N    61#
FLAGTICK         0001 N    60#
FLCLSE           099A L  1885  1894  1977  2305#
FLOAD            010A N   199# 1970
FLOPN            0992 L  1854  1874  1889  2308#
FLSTATTACH       009E N   189# 1079
FLSTDETACH       009F N   190#
FLSTOUT          0005 N   110# 1532
FLUSHED          0902 V  4574#
FMALLOC          0080 N   159# 2561  2584  2627  3174
FMAUALLOC        0309 N   217#
FMAUFREE         030A N   218#
FMEMFREE         0082 N   160# 2135  2643  2729
FMLALLOC         0308 N   219#
FMLFREE          030C N   220#
FNAMSIZ          0008 N    52#
```

```
FNOABORT          0217 N   210#
FNOABORTSPEC      0219 N   212#
FOI               099E L  2307  2313#
FOKABORT          0218 N   211#
FORMODA           0A88 L  2478  2481#
FPARSEFILENAME    0098 N   183# 2323
FQMAKE            0086 N   164#
FQOPEN            0087 N   165# 1756  3481
FQREAD            0089 N   167# 2413  3499
FQWRITE           0088 N   169# 2953
FRCONIN           0402 N   227# 1527
FRCONOUT          0403 N   228# 1513
FREEMODE          03F6 V  4562#
FREEROUT          0052 V  3822# 4573
FSETDMA           001A N   132#
FSETDMAB          0033 N   149#
FSETPRIOR         0091 N   175# 2090
FSETRNDREC        0024 N   142#
FSHARE            0308 N   216# 3204  3248
FSLEEP            0212 N   205#
FSYNC             0215 N   208# 2315
FTERMINATE        008F N   173# 1097  1360  1440  1518  2174  2188
FTYPSIZ           0003 N    53#
FUNC              0889 V  4468#
FUNSYNC           0216 N   209# 2318
FUSERCODE         0020 N   138#
FWAKEUP           0213 N   206#
FXCHK             0CC7 L  2769# 2794
FXERR             0D14 L  2761  2764  2815  2819#
FXINTRA           087A V  4445#
FXREAD            0CA5 L  2752# 2796
FXREADOK          0CC4 L  2759  2767#
GCIFO             0AC6 L  2480  2487  2508#
GOCIO             0295 L  1499#
GOXIOS            02BC L  1439  1442# 1517  1520#
GXI               02CB L  1443  1444  1447# 1521  1522  1525#
HASH              0848 V  4413#
HASHL             084F V  4414#
HASHSEG           0886 V  4517#
HCS               0E60 L  2930  2933#
HHDR              0A32 L  2441  2443#
HIGHEXT           089E V  4490#
IBREAKCS          0002 V  3639#
IBREAKIP          0000 V  3638#
IDEBUGCS          0386 V  3646#
IDEBUGIP          0384 V  3645#
IDIVIDECS         0002 V  3633#
IDIVIDEIP         0000 V  3632#
IENT              0269 L  1219  1381  1412#
IINTERRUPTS       0014 V  3642#
ILLFUNC           0178 L  1218# 1238  1242
INCRPDCNT         09F5 V  4561#
INDISP            0622 V  4223#
```

```
INFO                0881 V  4455#
INFOFCB             083C V  4760#
INIT                0042 L   969   979  1003#
INITBASE            001A L  2751  2765  2772  2328#
INITPD              0477 V  1017  1070  3840  3845  4157#
INITIOS             05E0 V  1013  4137# 4190
INITUDA             04P0 V  1018  1946  4186# 4188
INUMASKCS           000A V  3637#
INUMASKIP           0008 V  3636#
INSUP               0100 L  1265  1276#
INTFLAG             0623 V  4224#
IOAUXIN             0005 N   262# 1540
IOAUXOUT            0006 N   263# 1536
IOCONIN             0001 N   258#  277#
IOCONOUT            0002 N   259#  278#
IOCONST             0000 N   257#  276#
IOFLUSH             000C N   269#  272   301#  302
IOLIST              0004 N   261#  279#
IOLISTST            0003 N   260# 1543
IOPGLLDEV           000D N   270#  293#
IOREAD              000A N   267#  287#
IOSCS               0382 V  1034  1060  3644#
IOSELDSK            0009 N   266#  283#
IOSIP               0380 V  1033  1059  3643#
IOSTATETNE          0008 N   265#
IOSWITCH            0007 N   264#
IOVRFLWCS           0012 V  3641#
IOVRFLWIP           0010 V  3640#
IOWRITE             000B N   268#  288#
ITRACECS            0006 V  3635#
ITRACEIP            0004 V  3634#
JOS                 09AF L  2303  2321# 2324  2327
LBBMRE              0D6C L  2849# 2871
LBBNSH              0D7F L  2851  2853#
LBBNXT              0DB9 L  2849  2871#
LBBNZER             0DAE L  2862  2865  2867#
LBBZER              0DA9 L  2863  2866#
LBCMRE              0D38 L  2836# 2837
LBEND               0D4E L  2832  2840#
LBEND80             0D49 L  2836  2839#
LBMORE              0D26 L  2832# 2833
LCB                 0086 V  3877#
LCBLEN              000A N   965#
LDCF                0A02 L  2417  2425#
LDCNT               088F V  4617#
LDOUT               0BD6 L  2559  2568  2591  2634# 2701  2714  2742  2821  2838  2932
LDTAB               0273 V  2469  2477  2604  2635  2657  2669  2690  2807  2830  2834
                            2847  2891  4108#
LDTABSIZ            00AA N    63# 2463  4108
LDTATR              0008 N  2372# 2373  2484  2545  2579  2625  2712
LDTFLEN             000C N  2374# 2375  2525  2692  2978  2979  2992  2994  2999  3027
                            3039  3142
LDTFSTRT            000A N  2373# 2374  2526  2965  2992  2997  3002  3013  3025  3038
```

```
LDTID           000F N  2376#  2377  2563  2586  2629  2637  2641
LDTLEN          0011 N  2377#  2485  2505  2620  2648  2566  2686  2745  2812  2833
                        2837  2871
LDTMAX          0004 N  2370#  2371  2482  2532  2614  2616  2617  2618  2662  2680
LDTMIN          0002 N  2369#  2370  2481  2528  2607  2609  2611  2612  2613  2660
                        2661  2673  2680  2683  2683  2684  2857
LDTPD           0006 N  2371#  2372  2483  2533  2626  2640  3184
LDTSTART        0000 N  2368#  2369  2527  2549  2563  2582  2586  2608  2629  2659
                        2663  2663  2664  2671  2672  2672  2684  2695  2695  2775
                        2781  2841  2870  2893  2989  2991  3024  3037  3200  3255
LDTTYPE         000E N  2375#  2376  2524  2551  2576  2609  2832  2836  2849  2850
LF              000A N   834#
LFMRE           0C80 L  2692#  2745
LFNXT           0C99 L  2692  2697  2745#
LGMORE          06DF L  2637#  2648
LGNEXT          06FD L  2637  2648#
LINFO           0910 V  4598#
LIP1            0DEA L  2888  2891#
LNOT80          0D04 L  2875  2880#
LOAD            090D L  1390  2335  2386  2400#
LOADENT         0905 L  1384  2390#
LOCALFUNC       01BB L  1215  1251  1254  1260  1264#
LOCKCNT         03FD V  4570#
LOCKMAX         008A V  3879#
LOCKROUT        0907 V  4580#
LOCKSHELL       03F2 V  4558#
LOCKSP          03F0 V  4556#
LOCKUNLOCK      03F4 V  4560#
LODBOBO         01ED V  2829  2839  2845  2875  4103#
LODBASEP        01FC V  2842  2887  2922  2949  4092#
LODCHAIN        01E9 V  2415  2419  2487  4099#
LODDISK         01EB V  2453  2459  2914  3130  3196  4101#
LODDMA          01F3 V  2433  2513  2755  2768  2793  2980  3005  4107#
LODEXIT         0EA4 L  2442  2649  2890  2951#
LODFCB          01C2 V  2425  2434  2454  3067  3124  4095#
LODFIFTY        01EC V  2456  2881  4102#
LODFIXREC       01EF V  2517  2753  2763  2795  4105#
LODFIXREC1      01F1 V  2516  2762  4106#
LODGROUP        0EB2 L  2694  2957#  3016
LODINDMA        01F6 V  2444  2964  3015  4096#
LODLBYTE        01EE V  2515  2750  4104#
LODLSTK         018A V  2898  2924  4091#
LODNLDT         01BE V  2476  2486  2596  2605  2636  2658  2670  2691  2806  2831
                        2835  2848  4093#
LODNREC         01F8 V  2606  2619  2665  2676  2685  4097#
LODPD           01C0 V  2416  2417  2419  2478  2483  2504  2533  2623  2626  2888
                        2892  3245  4094#
LODUDA          01B8 V  2896  2919  4090#
LODUSER         01EA V  2451  2462  2913  3132  3195  4100#
LOGFXS          0850 V  4416#
LRET            080D V  4376#
LSCSEG          0058 V  3615#
LSFLAGS         005A V  2945  3616#
```

```
LSLMRE              0036 L  2671# 2686
LSLNXT              0074 L  2671  2686#
LSMORE              0010 L  2655# 2666
LSNEXT              002A L  2659  2662  2665#
LSOFFSET            0056 V  2944  3614#
LSRCSEG             005E V  2947  3615#
LSROFFSET           005C V  2946  3617#
LSSP                0056 V  2937  3613#
LSTKLEN             0060 N  2481  2482  2901  3608# 3611
LIM                 0395 L  2612  2614#
LIMI                03A7 L  2617  2619#
LIMORE              0E73 L  2607# 2620
LINEXT              0BAB L  2607  2608  2620#
LISPRD              0C09 L  2630  2653#
MAPLEN              0006 N   713#
MAPMAU              0000 N   710#  711
MAPSAT              0002 N   711#  712
MAPSTART            0004 N   712#  713
MAKEFLAG            0309 V  4541#
MAKEXFCB            0813 V  4381#
MAL                 0076 V  3847#
MAXALL              08BD V  4527#
MDLEN               000A N   645#
MDUL                0058 V  3825#
MEM                 0003 N    88#  216   217   218   219   220   631  3985  3986  3987
                           3988  3989  3990  4027  4028  4029
MEMCNT              0640 V  4247#
MEMMOD              0010 N  1041  3768#
MEMMODBIT           0004 N    98#
MEMSPB              0541 V  3890  4250#
MFCODE              0004 N   684# 2542  2544  2578  2712  3240  3241
MFL                 005A V  3826#
MFLOAD              0001 N   682# 2434  2535  2625  2712  3240  3241
MFPBLEN             0004 N   698#
MFPBPD              0002 N   697#  698
MFPBSTART           0000 N   696#  697
MFSHARE             0002 N   683# 2544  2712  3240  3241
MFUDAONLY           0040 N   688#
MLENGTH             0004 N   642#  643   653
MLINK               0000 N   640#  641   651
MMP                 004C V  3814#
MODENTRY            0000 N   243#  244
MODINIT             0004 N   244#  245  1039
MODLEN              0008 N   245#
MODULEMAP           0046 V  1042  1046  1050  1253  1267  3801#
MODULETABLE         0000 N  1272  3759#
MPBFLAGS            0008 N   675#  677
MPBLEN              000A N   677#
MPBMAX              0004 N   673#  674
MPBMIN              0002 N   672#  673
MPBPDADR            0006 N   674#  675
MPBSTART            0000 N   671#  672
MPLIST              0006 N   643#  644   654
```

```
MPM             0000 N    65#   66    275   1065  1429  1829  2539  2704  3105  3853
MSFLAGS         0006 N    654#  3239
MSGSPB          03A0 V   4275  4784#
MSLENGTH        0004 N    653#
MSLINK          0000 N    651#  3234  3237
MSMAU           0008 N    655#
MSSTART         0002 N    652#  2134  2727  3243  3244
MSTART          0002 N    641#   642   652
MULTCNT         0894 V   4481#
MULTNUM         0618 V   4385#
MULTSEG         082C V   4391#
MUNUSED         0008 N    644#   645   655
MX1             09AC L   2316  2320#
MXDTSKQD        0874 V   4074  4763# 4776
MXDISKQPB       0890 V   4774#
MXLOADQD        0194 V   3846  4074# 4083
MXLOADQPB       01B0 V   2413  2953  4082#
N80M            0E83 L   2940  2942#
N80D            008D L   1042  1046#
NCCB            0049 V   3811#
NC10            0099 L   1046  1050#
NC10DEV         0085 V   1067  1069  3875#
NCMDCHAR        06A4 L   2003# 2014
NCNS            0047 V   3809#
NCONDEV         0083 V   1067  1071  1830  3873#
NDELAMS         000E N   3383  3401#
NDP8087         0091 V   3884#
NET             0007 N     92# 4000  4001  4002  4003  4004  4005
NETCHK          0178 L   1171  1214  1221#
NETMOD          0030 N   1256  3785#
NETMODBTI       0040 N    102# 1254
NFLAGS          004A V   3812#
NIMP            0262 L   1275  1380  1406#
NLOW            0686 L   2008  2009  2011#
NLST            0048 V   3810#
NLSTDEV         0084 V   1069  3874#
NODEL           1115 L   3385  3389#
NODF            0387 L   1653  1657  1662#
NOES            0E70 L   2935  2937#
NOLOADF         03BC L   2623  2626#
NOPREFIL        0a4E L   1963  1968#
NOQM            0498 L   1777  1780#
NOSWI           0120 L   1144  1158#
NOTAUXIN        02F5 L   1539  1542#
NOTAUXOUT       02EB L   1535  1538#
NOTCMOD         0602 L   1913  1938  1940#
NOTCONIN        02D5 L   1526  1529#
NOTCONOUT       02AF L   1511  1516#
NOTCONSTS       02A3 L   1485  1510#
NOTF1           05E4 L   1920  1924#
NOTF2           05EE L   1924  1928#
NOTF3           05F8 L   1932  1938#
NOTFS           01FD L   1316  1326#
```

```
NOTLISTOUT          02E1 L  1530  1534#
NOTLOAD             0821 L  2534  2536  2541#  2574#
NOTTOOMUCH          0C65 L  2681  2683#
NPRIOR              0769 L  2088  2092#
NRSP                00C1 L  1092#  1094
NSLAVES             004E V  3815#
NSTK                0161 L  1180  1186#
NTAIL               0898 L  2209  2222#
NTLOG               0804 V  4362#
NX10                00A7 L  1050  1055#
NXIOSFUNCS          000C N   272#  302#
OFFSETV             08C5 V  4531#
OKFUNC              0195 L  1237  1240  1244#
OLAPTYPE            0819 V  4386#
OPENCTL             08FC V  4569#
OPENEND             080A V  4369#
OPENMAX             0088 V  3880#
OPENROOT            0905 V  4579#
OPENVEC             0088 V  3878#
OSAF                0206 L  1329  1331#  1441  1501  1514  1519  1528  1533  1656  1757
                            1772  1974  2091  2096  2138  2175  2189  2237  2313  2321
                            2414  2497  2562  2585  2628  2644  2730  2954  3070  3088
                            3125  3176  3205  3249  3482  3490
OSINT               00E0 N    54#    55  1078  1079  1093  1098  1362  3642
OSSEG               0040 V  3797#
OSVERNUM            007C V  1426  3855#  3860#
OVERENT             0277 L  1391  1424#
OWNERBOOT           008C V  3881#
PABORT              0021 N   390#   391
PACKEDDCNT          08A1 V  4496#
PARAMETERSEGMENT    092B V  4628#
PARLO               080C V  4374#
PARSEGNT            0FFF L  1387  3270#
PATCH               11EF L  3508  3515#
PBCBCNT             0879 V  4441#
PCBFCBPTR           0002 N   788#   789  3303
PCBFLNMPTR          0000 N   787#   788  3302
PCBLEN              0004 N   789#
PCM11               0001 N   436#
PCMCTLC             0008 N   439#
PCMCTLO             0080 N   441#
PCMCTLS             0002 N   437#
PCMOD               0018 N   385#   386  1919  1921  1925  1935  1939
PCMROUT             0004 N   438#
PCMRSX              0300 N   442#
PCNS                0020 N   389#   390  1449  1668  1827
PCONMODE            0022 N   391#   392
PDADDR              08A4 V  4499#
PDCNT               089D V  4486#
PDLEN               0030 N    58#   776  1805
PDSP                0012 N   380#   381  1664  1820  2154  2281  2452
PERMSG              0960 V  4641  4647#
PF3067              8000 N   431#
```

```
PFACTIVE        0100 N   424#
PFCMLDABORT     0800 N   427#
PFCTLC          0080 N   423# 1359  1908  2101  2173  2186  2190  3095
PFCTLD          0400 N   426# 2104  2105  2109
PFDSKLD         2000 N   429# 2093
PFHELP          0002 N   416# 1359  2173
PFKEXNAL        0004 N   417# 4161
PFLAG           0006 N   377#  378  1359  1632  1636  1813  1908  2093  2101  2104
                         2106  2109  2173  2182  2185  2186  2190  3095
PFN             0982 L  1709  2040  2064  2323#
PFNENDP         1002 L  3308  3358  3360#
PFNERR          1068 L  3312  3313  3318  3319  3320  3329# 3337  3353  3391
PFNOCTLS        1000 N   428# 1635  2191  2194
PFNOKBD         4000 N   430#
PFNOUT          1005 L  3356  3359  3361#
PFPOKE          0008 N   418#
PFRAW           0040 N   421#
PFRESOURCE      0020 N   420#
PFSYS           0001 N   415# 1359  2173  4161
PFTABLE         0010 N   419# 1813
PFTEMPKEEP      0200 N   425# 1359  1635  2173  2181  2184
PHYMSK          08C8 V  4533#
PHYOFF          0870 V  4433#
PHYSHF          08C7 V  4532#
PLASTBCBA       0877 V  4440#
PLDSK           0014 N   382#  383  2914  3131  3196
PLINK           0000 N   373#  374  1802  2147  2738  3168  3221
PLR             006E V  3843#
PLST            0024 N   392#  393  1072  1456  1828  1832
PLUSER          0015 N   383#  384  2913  3133  3195
PMEM            0016 N   384#  385  1091  2130  2133  2502  2502  2504  2727  3234
PNAME           0008 N   378#  379  1814  3187  3187
PNAMSIZ         0008 N    50#   51    52   379   611  1817
PPARENT         001E N   388#  389  2169  2170
PPREF           002C N   393#  394
PPRTOR          0005 N   376#  377  2909
PRCHAR          08A5 L  2229  2233# 2242  2245  2248  2252  2254  2260  2263  2282
                        2283
PRDSK           0957 L  2274  2287  2289#
PRDUSR          0948 L  1779  2285#
PRDISK          0929 L  2257  2276#
PRDSK1          0930 L  2278  2282#
PRET            0923 L  2267  2271#
PRFCB           0A5D V  4700#
PRFCB1          0A65 V  4702#
PRFILNAM        08F0 L  1965  2256#
PRFX            0A48 V  4696#
PRFX1           0A5A V  4699#
PRNT            0976 L  2297  2299#
PRNAM           0974 L  2259  2280  2298#
PRNUM           080C L  2240  2243  2246  2250#
PRPDDSK         0936 L  2277  2279#
PRR             08B8 L  2236  2238#
```

```
PRR1            096F L   2291  2295# 2302
PRS1            1021 L   3307  3309#
PRS2            1053 L   3316  3322#
PRS3            1054 L   3317  3321  3324#
PRS4            1070 L   3327  3328  3331#
PRS5            1080 L   3331  3335  3336  3340#
PRS51           1081 L   3349  3351#
PRS8            108E L   3341  3351  3352  3355#
PRTIME          0889 L   1660  2240#
PRTYP           0970 L   2262  2297#
PRUSER          0924 L   2265  2270  2273#
PRVPOS          08F9 V   4566#
PSCIOWAIT       0009 N    410#
PSCRATCH        002E N    394#  395
PSDELAY         0002 N    403#
PSDQ            0006 N    407#
PSFLAGWAIT      0008 N    409#
PSNQ            0007 N    408#
PSPOLL          0001 N    402#
PSROOT          0000 N    401#  2908  4159
PSSLEEP         0005 N    406#
PSSWAP          0003 N    404#
PSSYNC          000A N    411#
PSTAT           0004 N    375#  376  2908
PSTERM          0004 N    405#
PTHREAD         0002 N    374#  375  2720  3122  3148  3149  3150  3151  3152  3153
                          3154  3155  3157  3212  3213  3214  3215  3217
PTKCNT          0019 N    386#  387
PUDA            0010 N    379#  380  1020  1139  1945  1960  2894  2907
PUL             005C V   1796  1802  2146  2147  2736  2737  3166  3168  3220  3222
                         3827#
PUSER           0013 N    381#  382  1665  1821  1887  1888  2156  2270  2450
PWAIT           001A N    387#  388
PWMODE          08DD V   4545#
PWSCRTCH        002E N    395#
QBUF            001A N    539#  540
QDLEN           001C N    540#
QDQ             0012 N    535#  536
QFDEV           0040 N    551#
QFHIDE          0004 N    546#
QFHELP          0002 N    545#  4076  4765
QFLAGS          0004 N    531#  532  1765  3487
QFMX            00C1 N    544#  4076  4765
QFRPL           0020 N    550#  3487
QFRSP           0008 N    548#  1765
QFTABLE         0010 N    549#
QLINK           0000 N    528#  529
QLR             0074 V   3846#
QMAU            0060 V   3830#
QMSGCNT         0016 N    537#  538
QMSGLEN         000E N    533#  534
QMSGOUT         0018 N    538#  539
QNAME           0006 N    532#  533
```

```
QNAMSIZ          0008 N    518   533   578  1745  1750  3433
QNET             0002 N   5298   530
QNMSGS           0010 N   5348   535
QNQ              0014 N   5368   537
QUKC             0003 N   5308   531
QPBBUFFPTR       0006 N   5768   577
QPUFLGS          0000 N   5728   573
QPULEN           0010 N   5788
QPUNAML          0008 N   5778   578  1744  2285
QPUNET           0001 N   5738   574
QPUNMSGS         0004 N   5758   576
QPUQADDR         0002 N   5748   575  1764  3485
QSPB             0649 V   4250  4259#
QUESTR           09C2 V   2286  2329#
QUL              005E V   3828#
RCOUNT           0913 V   4602#
RDISI            0EFE L   2994  2996#
RDAGN            0F30 L   3009  3013#
RDDIRFLAG        0810 V   4378#
RDFIRSI          0EF5 L   2969  2973  2993# 3040
RDINDMA          0F0B L   2997  3000#
RDK3             0F2F L   3010  3012#
RDK4             0F65 L   3033  3035#
RELOG            0821 V   4392#
REQUIREDTABLE    08DF V   4551#
RESEL            080F V   4377#
RETCLEI          0468 L   1758# 1767  1773
RETURNSEG        092D V   4629#
RLOG             0800 V   4360#
RLR              0068 V   1138  1358  1447  1632  1662  1819  1886  1907  1959  2100
                          2153  2167  2182  2235  2269  2280  2290  2301  2449  2501
                          3094  3840#
RMF              0900 V   4595#
RMPDI            07DB L   2130  2145#
RUDMSG           0969 V   4641  4649#
RUDSK            0806 V   4366#
RUFMSG           0978 V   4641  4652#
ROOTSCBA         0873 V   4435#
RPLENT           1168 L   1386  3436#
RPLERR           11E3 L   3484  3488  3491  3501#
RPLRET           11DA L   3467  3497#
RSPBOTTOM        0140 N    777#
RSPLINK          0000 N    769#   770   771  1086  1088
RSPMD            0008 N    774#
RSPNCOPIES       0004 N    772#   773
RSPO             00EC L   1094  1095#
RSPPD            0010 N    775#   776  1089
RSPRESERVED      0005 N    773#
RSPSDAIVAR       0002 N    771#   772
RSPSEG           0042 V   1084  1087  3798#
RSPSYSDAI        0000 N    770#
RSPTOP           0000 N    768#   769
RSPUDA           0040 N    776#   777
```

```
                        3991  3993  3994  3995  3996  4018  4019  4050  4051  4052
                        4054  4055  4066
SUPERVISOR      0008 V   984#
SUPFUNC         0244 V  1279  1380#
SUPMOD          0000 N  3760#
SUPMODBIT       0001 N    96#
SYSDAT          0006 V   983# 1020  1083  1137  1166  1336  1684  1838  1949  1957
                        2439  2470  2906  3006  3486  3728
SYSENT          00A0 V  1247  3923#
SYSFUNC         0170 L  1200# 1318  1338  1398
TAB             0009 N   833# 3398  3409
TBLSRCH         0CFF L  2774  2780  2798#
TEMPDISK        0050 V  3817#
THRDRT          0072 V  3845#
THRDSPB         0659 V  4267  4275#
TICK            0C0C V  4838#
TICKSPERSEC     0051 V  3818#
TLOG            0802 V  4361#
TOD             007E V  1562  3865# 4590#
TODDAY          007E V  3866#
TODENT          0305 L  1389  1557#
TODHR           0080 V  2240  3867#
TODLEN          0005 N    59# 1562
TODMIN          0081 V  2243  3868#
TODSEC          0082 V  2246  3869#
TRACK           087B V  4447#
TRUE            FFFF N    45# 1257  1647  1658  1680  1728  1730  1777  1913  1942
                        1963  1971  1973  2088  2114  2165  2171  2419  2487  4301
TRYCODE         0B2D L  2574  2577#
U8087LEN        015E N    57#
UA0             0827 V  4397  4400#
UA1             082D V  4400  4402#
UA2             0833 V  4402  4404#
UA3             0839 V  4404  4406#
UA4             083F V  4406  4408#
UA5             0845 V  4408  4410#
UALROOT         0825 V  4397#
UAX             0020 V  3690#
UBP             002C V  3696#
UBX             0022 V  3691#
UCONCCB         0062 V  2234  2289  2300  3719#
UCX             0024 V  3692#
UDAOVLLEN       0019 N  3683#
UDASAVE         0929 V  4627#
UDBLK           000E V  3676#
UDCNI           000C V  3677#
UDEBUGCS        005E V  3716#
UDEBUGIP        005C V  3715#
UDELIM          0066 V  3721#
UDFPASSWORD     0012 V  3681#
UDI             0028 V  3694#
UDINSYS         0060 N  3663# 4188
UDMAUFST        0002 V  1672  1841  2161  2920  3051  3671# 3683
```

```
UDMASEG          0004 V  1674  1844  2163  2923  3052  3091  3672#
UDPARAM          0000 V  3668#
UDSSAV           0032 V  3699#
UDX              0026 V  3693#
UECONT           0127 L  1157  1162#
UEOUT            0165 L  1185  1189#
UERR1            016E L  1192  1194#
UERR2            01FB L  1321  1323#
UERR3            022C L  1344  1346#
UERRORMODE       0010 V  1670  1853  1976  2158  3679#
UESSAV           004C V  3706#
UFLAGSAV         004E V  3707#
UFUNC            0006 V  1169  3673#
UINTNT           001B V  3687#
UINTICS          0050 V  2933  3709#
UINTIDS          0052 V  2926  3710#
UINTIES          0054 V  2927  2936  3711#
UINTISS          0056 V  2928  2943  3712#
UINSYS           0060 V  1144  1164  1180  3717#
UIVECTORS        0038 V  3703#
UIVECTORS2       0044 V  3705#
ULEN             0100 N    56#   57   777  1156  1953  2481  2482  2898  2901  4186
ULSTCOD          0064 V  3720#
UMULTCNT         0011 V  3063  3065  3072  3680#
UNKNOWN          0000 N    47#  3851  4164
UOSCS            005A V  3714#
UOSIP            0058 V  3713#
UPDCNT           001A V  3682#
URETSEG          0030 V  1140  1177  1552  3495  3499  3698#
USEARCHA         0008 V  3675#
USEARCHABASE     000A V  3676#
USEARCHL         0007 V  3674#
USEDDSK          0A59 L  2457  2460#
USEDUSR          0A64 L  2461  2463#
USER             0000 N    85#   107   110   114   115   120   121   126   131   132
                           138   139   142   148   149   155   156   158   159   160
                           162   163   164   165   167   168   169   170   172   173
                           174   175   176   178   180   183   189   190   193
USERENTRY        00F2 L  1033  1059  1109#
USERREIF         022E L  1352#  2946
USERSTK          09CC V  2273  2331#
USERZEROPASS     0923 V  4618#
USI              002A V  3695#
USP              001C V  3688#
USRCODE          0880 V  4453#
USS              001E V  3689#
USTACKSP         0034 V  1154  1184  2937  3700#
USTACKSS         0036 V  1152  1183  3702#
USTATSAV         0061 V  3718#
UUNUSED          0040 V  1328  3704#
UWRKSEG          002E V  1024  1151  1159  1160  1179  1187  1333  1335  1339  1341
                         1436  1480  1560  1570  1695  2427  3299  3464  3494  3697#
VERSION          0078 V  3851#
```

```
VRECORD          0915 V  4604#
WFLAG            090E V  4596#
XCBCX            0001 N   312#  313
XCBDX            0003 N   313#  314
XCBFUNC          0000 N   311#  312
XCBLEN           0005 N   314#
XCS              02A2 L  1504  1507#
XDLNI            0B15 V  4385#
XE10             0A10 V  4643  4684#
XE11             0A29 V  4643  4689#
XE3              0995 V  4643  4659#
XE5              0983 V  4643  4666#
XE6              09C7 V  4643  4670#
XE7              09DC V  4643  4674#
XE8              09EB V  4643  4677#
XE9              09FF V  4643  4681#
XERRLIST         0946 V  4641#
XFCBREADONLY     089F V  4491#
XFD              0F3E L  2999  3020#
XFER             0ED2 L  2978  2980#
XFERN            0F66 L  3032  3036#
XIOS             0006 N    91#
XIOS.F           023E L  1066  1068  1368# 1460  1537  1541  1544
XIOSMOD          0028 N  1053  1370  3781#
XIOSMODBIT       0020 N   101# 1050
ZEROLENGTH       0015 N  4390#
```

SER. # _____