```
                        include copyright.def
                ;********************************************************
                ;*                                                      *
                ;*              Concurrent CP/M-86 V2.1                 *
                ;*              =======================                 *
                ;*                                                      *
                ;*              Copyright (c) 1982                      *
                ;*                                                      *
                ;*                 Digital Research                     *
                ;*                   P.O.Box 579                        *
                ;*          Pacific Grove, California 93950             *
                ;*                                                      *
                ;*                 (408) 649-3896                       *
                ;*                 TWX 9103605001                       *
                ;*                                                      *
                ;* All Information contained in this source listing is  *
                ;*                                                      *
                ;*                    PROPRIETORY                       *
                ;*                    ===========                       *
                ;*                                                      *
                ;* All rights reserved.  No part of this document       *
                ;* may be reproduced, transmitted, stored in a          *
                ;* retrieval system, or translated into any language    *
                ;* or computer language, in any form or by any means    *
                ;* without the prior written permission of Digital      *
                ;* Research, P.O Box 579, Pacific Grove, California.    *
                ;*                                                      *
                ;********************************************************


                ;********************************************************
                ;*                                                      *
                ;*      MP/M-86 and Concurrent CP/M-86                  *
                ;*           Character I/O Module                       *
                ;*                                                      *
                ;********************************************************
```

```
58
59  =                           eject 1 include system.def
60  =                       ;*****************************************************
61  =                       ;*
62  =                       ;*      SYSTEM DEFINITIONS
63  =                       ;*
64  =                       ;*****************************************************
65  =
66  =  FFFF                 true        equ  0ffffh     ; value of TRUE
67  =  0000                 false       equ  0          ; value of FALSE
68  =  0000                 unknown     equ  0          ; value to be filled in
69  =  0080                 dskrecl     equ  128        ; log. disk record len
70  =  0020                 fcblen      equ  32         ; size of file control block
71  =  0008                 pnamsiz     equ  8          ; size of process name
72  =  0008                 qnamsiz     equ  pnamsiz    ; size of queue name
73  =  0008                 fnamsiz     equ  pnamsiz    ; size of file name
74  =  0003                 ftypsiz     equ  3          ; size of file type
75  =  00E0                 osint       equ  224        ; int vec for O.S. entry
76  =  00E1                 debugint    equ  osint+1    ; int vec for debuggers
77  =  0100                 ulen        equ  0100h      ; size of uda
78  =  015E                 u8087len    equ  ulen + 94  ; size of uda when process uses 8087
79  =  0030                 pdlen       equ  030h       ; size of Process Descriptor
80  =  0005                 todlen      equ  5          ; size of Time of Day struct
81  =  0001                 flag_tick   equ  1          ; flag 0 = tick flag
82  =  0002                 flag_sec    equ  2          ; flag 1 = second flag
83  =  0003                 flag_min    equ  3          ; flag 2 = minute flag
84  =  00AA                 ldtabsiz    equ  0aah       ; ldtablen=11, 10 entries
85  =
86  =  0000                 mpm     equ     false       ; MP/M-86 or
87  =  FFFF                 ccpm    equ     not mpm     ; CCP/M-86
88  =
89  =  0000                 3CPM    equ     false       ; CP/M-86 BDOS
90  =  FFFF                 3MPM    equ     not 3CPM    ; multi-process BDOS
91
```

```
                        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
                ;*             DX = 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
```

```
                        ;f_fdelete      equ     (user * 0100h) + 19
0014                    f_freadseq      equ     (user * 0100h) + 20
                        ;f_fwriteseq    equ     (user * 0100h) + 21
                        ;f_fmake        equ     (user * 0100h) + 22
                        ;f_frename      equ     (user * 0100h) + 23
                        ;f_loginvector  equ     (user * 0100h) + 24
0019                    f_getdefdisk    equ     (user * 0100h) + 25
001A                    f_setdma        equ     (user * 0100h) + 26
                        ;f_getallocvec  equ     (user * 0100h) + 27
                        ;f_writeprotect equ     (user * 0100h) + 28
                        ;f_getrovector  equ     (user * 0100h) + 29
                        ;f_setfileattr  equ     (user * 0100h) + 30
                        ;f_getdpb       equ     (user * 0100h) + 31
0020                    f_usercode      equ     (user * 0100h) + 32
0021                    f_freadrdm      equ     (user * 0100h) + 33
                        ;f_fwriterdm    equ     (user * 0100h) + 34
                        ;f_filesize     equ     (user * 0100h) + 35
0024                    f_setrndrec     equ     (user * 0100h) + 36
                        ;f_resetdrive   equ     (user * 0100h) + 37
                        ;f_accessdrive  equ     (user * 0100h) + 38
                        ;f_freedrive    equ     (user * 0100h) + 39
                        ;f_writerndzero equ     (user * 0100h) + 40
                        ;f_chain        equ     (user * 0100h) + 47
0032                    f_callbios      equ     (user * 0100h) + 50
0033                    f_setdmab       equ     (user * 0100h) + 51
                        ;f_getdma       equ     (user * 0100h) + 52
                        ;f_getmaxmem    equ     (user * 0100h) + 53
                        ;f_getabsmaxmem equ     (user * 0100h) + 54
                        ;f_allocmem     equ     (user * 0100h) + 55
                        ;f_allocabsmem  equ     (user * 0100h) + 56
0039                    f_freemem       equ     (user * 0100h) + 57
003A                    f_freeall       equ     (user * 0100h) + 58
                        ;f_userload     equ     (user * 0100h) + 59
006E                    f_delim         equ     (user * 0100h) + 110
0080                    f_malloc        equ     (user * 0100h) + 128
0082                    f_memfree       equ     (user * 0100h) + 130
                        ;f_polldev      equ     (user * 0100h) + 131
0084                    f_flagwait      equ     (user * 0100h) + 132
0085                    f_flagset       equ     (user * 0100h) + 133
0086                    f_qmake         equ     (user * 0100h) + 134
0087                    f_qopen         equ     (user * 0100h) + 135
                        ;f_qdelete      equ     (user * 0100h) + 136
0089                    f_qread         equ     (user * 0100h) + 137
008A                    f_cqread        equ     (user * 0100h) + 138
008B                    f_qwrite        equ     (user * 0100h) + 139
008C                    f_cqwrite       equ     (user * 0100h) + 140
                        ;f_delay        equ     (user * 0100h) + 141
008E                    f_dispatch      equ     (user * 0100h) + 142
008F                    f_terminate     equ     (user * 0100h) + 143
0090                    f_createproc    equ     (user * 0100h) + 144
0091                    f_setprior      equ     (user * 0100h) + 145
0092                    f_conattach     equ     (user * 0100h) + 146
```

```
178
179  =  0093                f_condetach      equ     (user * 0100h) + 147
180  =                      ;f_setdefcon     equ     (user * 0100h) + 148
181  =  0095                f_conassign      equ     (user * 0100h) + 149
182  =                      ;f_clicmd        equ     (user * 0100h) + 150
183  =                      ;f_callrsp       equ     (user * 0100h) + 151
184  =  0098                f_parsefilename  equ     (user * 0100h) + 152
185  =                      ;f_getdefcon     equ     (user * 0100h) + 153
186  =                      ;f_sdataddr      equ     (user * 0100h) + 154
187  =                      ;f_timeofday     equ     (user * 0100h) + 155
188  =                      ;f_pdaddress     equ     (user * 0100h) + 156
189  =                      ;f_abortprocess  equ     (user * 0100h) + 157
190  =  009E                f_lstattach      equ     (user * 0100h) + 158
191  =  009F                f_lstdetach      equ     (user * 0100h) + 159
192  =                      ;f_setdeflst     equ     (user * 0100h) + 160
193  =                      ;f_clstattch     equ     (user * 0100h) + 161
194  =  00A2                f_cconattch      equ     (user * 0100h) + 162
195  =                      ;f_osvernum      equ     (user * 0100h) + 163
196  =                      ;f_getdeflst     equ     (user * 0100h) + 164
197  =
198  =
199  =                              ; Internal SUP functions
200  =  010A                f_load           equ     (sup * 0100h) + 10
201  =  010E                f_cload          equ     (sup * 0100h) + 14
202  =
203  =
204  =                              ; Internal RTM functions
205  =  0203                f_inflagset      equ     (rtm * 0100h) + 3
206  =  0212                f_sleep          equ     (rtm * 0100h) + 18
207  =  0213                f_wakeup         equ     (rtm * 0100h) + 19
208  =  0214                f_findpdname     equ     (rtm * 0100h) + 20
209  =  0215                f_sync           equ     (rtm * 0100h) + 21
210  =  0216                f_unsync         equ     (rtm * 0100h) + 22
211  =  0217                f_no_abort       equ     (rtm * 0100h) + 23
212  =  0218                f_ok_abort       equ     (rtm * 0100h) + 24
213  =  0219                f_no_abort_spec  equ     (rtm * 0100h) + 25
214  =
215  =
216  =                              ; Internal MEM functions
217  =  0308                f_share          equ     (mem * 0100h) + 8
218  =  0309                f_maualloc       equ     (mem * 0100h) + 9
219  =  030A                f_maufree        equ     (mem * 0100h) + 10
220  =  030B                f_mlalloc        equ     (mem * 0100h) + 11
221  =  030C                f_mlfree         equ     (mem * 0100h) + 12
222  =
223  =
224  =                              ; Internal CIO functions
225  =  040E                f_conprint       equ     (cio * 0100h) + 14
226  =  0417                f_cioterm        equ     (cio * 0100h) + 23
227  =  0418                f_ciostat        equ     (cio * 0100h) + 24
228  =  0402                f_rconin         equ     (cio * 0100h) + 2
229  =  0403                f_rconout        equ     (cio * 0100h) + 3
230  =
```

```
231
232  =
233  =                              ; internal BDOS functions
234  =  0520                f_bdosterm      equ     (bdos * 0100h) + 45
235
```

```
                        eject ! include xiosch.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_listst       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_listst      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
```

```
                                io_flush        equ     24
                                nxiosfuncs      equ     io_flush
                                endif

                                ;*******************************************************
                                ;*
                                ;*      XIOS Parameter Block for CALL XIOS functions
                                ;*
                                ;*******************************************************

0000                            xcb_func        equ     0
0001                            xcb_cx          equ     word ptr xcb_func + byte
0003                            xcb_dx          equ     word ptr xcb_cx + word
0005                            xcblen          equ     xcb_dx + word
```

```
305
306  =                      eject | include pd.def
307  =                ;*****************************************************
308  =                ;*
309  =                ;*      Process Descriptor - with the UDA associated
310  =                ;*              with the PD, describes the current
311  =                ;*              state of a Process under MP/M-CCP/M
312  =                ;*
313  =                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
314  =                ;* 00|   link    |  thread   |stat |prior|   flag    |
315  =                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
316  =                ;* 08|                     Name                      |
317  =                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
318  =                ;* 10|    uda    | dsk | user| ldsk|luser|    mem    |
319  =                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
320  =                ;* 18| cmod|tkcnt|   wait    | reserved  |  parent   |
321  =                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
322  =                ;* 20| cns |abort|  conmode  | lst |     reserved    |
323  =                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
324  =                ;* 28|        reserved       |   pret    |  scratch  |
325  =                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
326  =                ;*
327  =                ;*      link    - Used for placement into System Lists
328  =                ;*      thread  - link field for Thread List
329  =                ;*      stat    - Current Process activity
330  =                ;*      prior   - priority
331  =                ;*      flag    - process state flags
332  =                ;*      name    - name of process
333  =                ;*      uda     - Segment Adress of User Data Area
334  =                ;*      dsk     - Current default disk
335  =                ;*      user    - Current default user number
336  =                ;*      ldsk    - Disk program loaded from
337  =                ;*      luser   - User number loaded from
338  =                ;*      mem     - pointer to MD list of memory owned
339  =                ;*                by this process
340  =                ;*      cmod    - compatibility mode bits.
341  =                ;*                80 -> F1 bit
342  =                ;*                40 -> F2 bit
343  =                ;*                20 -> F3 bit
344  =                ;*                10 -> F4 bit
345  =                ;*      tkcnt   - temp keep count, # times tempkeep has been
346  =                ;*                turned on
347  =                ;*      wait    - parameter field while on System Lists
348  =                ;*      org     - Network node that originated this process
349  =                ;*      net     - Network node running this process
350  =                ;*      parent  - process that created this process
351  =                ;*      cns     - default console #
352  =                ;*      abort   - abort code
353  =                ;*      lst     - default list #
354  =                ;*      conmode - console mode for function 109
355  =                ;*      reserved- not currently used
356  =                ;*      pret    - return code at termination
357  =                ;*      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_ciowait      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
```

```
411
412  =  0080                pf_ctlc         equ     00080h  ; abort pending
413  =  0100                pf_active       equ     00100h  ; active tty
414  =  0200                pf_tempkeep     equ     00200h  ; don't terminate yet...
415  =  0400                pf_ctld         equ     00400h  ; explicit detach occured
416  =  0800                pf_childabort   equ     00800h  ; child terminated abnormally
417  =  1000                pf_noctls       equ     01000h  ; control S not allowed
418  =  2000                pf_dskld        equ     02000h  ; process was loaded from disk
419  =  4000                pf_nokbd        equ     04000h  ; ignore keyboard
420  =  8000                pf_8087         equ     08000h  ; process uses 8087
421  =                      ;
422  =                      ;       Process descriptor pcm_flag bit values
423  =                      ;               (console mode set by function 109)
424  =                      ;
425  =  0001                pcm_ll          equ     00001h  ; control C status for function 11
426  =  0002                pcm_ctls        equ     00002h  ; disable control S-Q
427  =  0004                pcm_rout        equ     00004h  ; raw output, no tabs or control P
428  =  0008                pcm_ctlc        equ     00008h  ; disable control C
429  =                      ;pcm_rsrv       equ     00040h  ; used by CP/M 3.0
430  =  0080                pcm_ctlo        equ     00080h  ; disable control O
431  =  0300                pcm_rsx         equ     00300h  ; 2 bits used by RSX for status
432
```

```
                        eject ! 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_nocliq            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_noclip            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 q1.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     byte ptr q_net + byte
0004                q_flags         equ     word ptr q_org + byte
0006                q_name          equ     byte 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
```

```
                        
0008                    qf_rsp          equ     008h    ; rsp queue
0010                    qf_table        equ     010h    ; from qd table
0020                    qf_rpl          equ     020h    ; rpl queue
0040                    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)
                        ;*
                        ;*******************************************************
                        
0000                    qpb_flgs        equ     byte ptr 0
0001                    qpb_net         equ     byte ptr qpb_flgs + byte
0002                    qpb_qaddr       equ     word ptr qpb_net + byte
0004                    qpb_nmsgs       equ     word ptr qpb_qaddr + word
0006                    qpb_buffptr     equ     word ptr qpb_nmsgs + word
0008                    qpb_name        equ     byte ptr qpb_buffptr + word
0010                    qpblen          equ     qpb_name + qnamsiz
```

```
                        eject 1 include acb.def
                ;***************************************************
                ;*
                ;*      Assign Console Control Block
                ;*
                ;***************************************************
                
acb_cns         equ     byte ptr 0
acb_match       equ     byte ptr acb_cns + byte
acb_pd          equ     word ptr acb_match + byte
acb_name        equ     byte ptr acb_pd + word
acblen          equ     acb_name + pnamsiz

                if mpm
                eject 1 include ccb.def
                endif
                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  |      qponmsgs     |     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 QPB 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.

                        ;CCB state flags

0001                    csm_buffered    equ     00001h
```

```
                                
0002 =                  csm_background  equ     00002h
0004 =                  csm_purging     equ     00004h
0008 =                  csm_noswitch    equ     00008h
0010 =                  csm_suspend     equ     00010h
0020 =                  csm_abort       equ     00020h
0040 =                  csm_filefull    equ     00040h
0080 =                  csm_ctrlS       equ     00080h
0100 =                  csm_ctrlQ       equ     00100h
0200 =                  csm_ctrlP       equ     00200h
                        
                        ;LCB - list control block is first ten bytes of WCCB
000A =                  lcblen          equ     10
                        
                                endif
```

```
                        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
```

```
                                eject | include cmode.def
                        ;
                        ;               bit masks for the console mode word
                        ;
0001                    cm_ctlc         equ     0001H
0002                    cm_ctls         equ     0002H
0004                    cm_rawo         equ     0004H
0100                    cm_rsx1         equ     0100H
0200                    cm_rsx2         equ     0200H
```

CP/M 8086 1.1 SOURCE: CIO.A86

```
                                eject 1 include cb.def
                        ;
                        ;       Character Block (CP/M 3.0 Character Control Block)
                        ;
0000                    cb_off          equ     word ptr 0
0002                    cb_seg          equ     word ptr cb_off + word
0004                    cb_len          equ     word ptr cb_seg + word
```

```
                        eject | include cioif.cio
                ;*****************************************************
                ;*
                ;*      Character I/O interface
                ;*
                ;*****************************************************

                        CSEG
                        org     0

0000 E93F00     0042    jmp init
0003 E97700     007D    jmp entry

                                                ;3 variables set by GENSYS
0006 0000               sysdat          dw      0
0008                    supervisor      equ     (offset $)
                                        rw      2

                        org     0ch
000C 06                 dev_ver         db      6       ;development system data version
                                                        ;set in sysdat.fmt

000D 434F50595249                       db      'COPYRIGHT (C) 1982,'
        474854202843
        292031393832
        2C
0020 204449474954                       db      ' DIGITAL RESEARCH '
        414C20524553
        454152434820
0032 585858582D30                       db      'XXXX-0000-'
        30303020
003C 363534333231       serial          db      '654321'

                ;====
                init:
                ;====
0042 CB                 retf

                ;*****************************************************
                ;*
                ;*      CIO function table
                ;*
                ;*****************************************************

0043 2D03               functab dw      conin_entry     ; 0 - Console Input
0045 4003                       dw      conout_entry    ; 1 - Console Output
0047 5003                       dw      rconin_entry    ; 2 - raw console input
0049 5603                       dw      rconout_entry   ; 3 - raw console output
004B 5C03                       dw      listout_entry   ; 4 - list output
004D BE03                       dw      dirio_entry     ; 5 - direct console I/O
004F D803                       dw      conwrite_entry  ; 6 - print string
0051 A904                       dw      conread_entry   ; 7 - read buffer
```

```
                                dw      constat_entry   ; 8 - console status
                                dw      conattach_entry ; 9 - attach console
                                dw      condetach_entry ; 10- detach console
                                dw      setdefcon_entry ; 11- set default console
                                dw      conassign_entry ; 12- assign console
                                dw      getdefcon_entry ; 13- get default console
                                dw      conprint_entry  ; 14- print string (internal)
                                dw      lstattach_entry ; 15- attach list
                                dw      lstdetach_entry ; 16- detach list
                                dw      setdeflst_entry ; 17- set default list
                                dw      clstattch_entry ; 18- cond list attach
                                dw      cconattch_entry ; 19- cond list detach
                                dw      getdeflst_entry ; 20- get default list
                                dw      notimp          ; 21
                                dw      notimp          ; 22
                                dw      cioterm         ; 23- cleanup for term.
                                dw      ciostat         ; 24- check status w/out
                                                        ;       changing raw/cooked flag
                                dw      cmode_entry     ; 25- get/set console mode word -
                                                        ;       user func 109
                                dw      delim_entry     ; 26- get/set delimiter - user func 110
                                dw      pblock_entry    ; 27- print block - user func 111
                                dw      lblock_entry    ; 28- list block - user func 112

                        ;=====          ================
                        entry:          ; CIO entry point
                        ;=====          ================
                                shl cx,1 ! mov si,cx
                                call cs:functab[si] ! retf

                        ;====           ==============
                        osif:           ; OS interface
                        ;====           ==============
                                callf cs:dword ptr .supervisor ! ret

                        ;======         ================
                        xiosif:         ; XIOS interface
                        ;======         ================
                                callf dword ptr .xiosmod ! ret

                        ;======
                        notimp:
                        ;======
                                mov bx,0ffffh
                                ret
```

```
                                eject 1 include console.cio
                        ;*****************************************************
                        ;*
                        ;*      Character I/O ownership routines
                        ;*
                        ;*****************************************************

                        ;===============
                        lstattach_entry:
                        ;===============
0096 33C9                       xor cx,cx                       ;unconditional attach code
0098 EB03       009D            jmps lcb_verify

                        ;===============
                        clstattch_entry:
                        ;===============
009A B9FFFF                     mov cx,0ffffh                   ;conditional attach code

                        lcb_verify:
                        ;----------
009D 51                         push cx
009E 2688366400                 mov si,u_lstccb                 ;U_LSTCCB=0 if not yet initialized
00A3 85F67508   00AF            test si,si 1 jnz lcb_ok
00A7 E80A02     0284              call getlcbadr
00AA 2689366400                   mov u_lstccb,si
                        lcb_ok:
00AF 59                         pop cx                          ;restore attach code
00B0 EB1A       00CC            jmps cioattach

                        ;===============
                        conattach_entry:
                        ;===============
00B2 33C9                       xor cx,cx                       ;unconditional attach code
00B4 EB03       00B9            jmps ccb_verify

                        ;===============
                        cconattch_entry:
                        ;===============
00B6 B9FFFF                     mov cx,0ffffh                   ;conditional attach code
                        ccb_verify:
00B9 51                         push cx
00BA 2688366200                 mov si,u_conccb
00BF 85F67508   00CB            test si,si 1 jnz ccb_ok
00C3 E8D001     0296              call getccbadr
00C6 2689366200                   mov u_conccb,si
                        ccb_ok:
00CB 59                         pop cx                  ;conditional attach if 0ffffh
                                ;jmps cioattach

                        ;=========
                        cioattach:              ;NOTE: PIN does attaches also for ^P
                        ;=========
```

```
                        ;       Attach calling process to default ciodev.
                        ;       if ciodev is being used, sleep on c_queue
                        ;       until another process detaches from device
                        ;       and calling process is next in queue.
                        ;       input:  SI = CCB address of device
                        ;               DI = offset of dev number in PD
                        ;               CX = 0ffffh if conditional
                        
=00CC A168008BD8                mov ax,rlr ! mov bx,ax
                                        ; BX = PD addr
                                        ; SI = CCB addr
                                        ; CX = Condition Flag
=00D1 9CFA                      pushf ! cli
=00D3 381C7425      00FC        cmp bx,c_attach[si] ! je ca_ret          ;we already own it
=00D7 833C00741E    00FA            cmp c_attach[si],0 ! je ca_atch      ;nobody owns it, we'll take it
=00DC E308          00E6            jcxz ca_sleep                        ;somebody else owns it
=00DE 9D8BFFFF                          popf ! mov bx,0ffffh             ;or it is a mimic list (0ffffh)
=00E2 892400C3                          mov cx,e_no_attach ! ret         ;- no attach in either case
                        ca_sleep:
=00E6 52                                push dx                          ;save DH=cons #
=00E7 8D5402                            lea dx,c_queue[si]               ;unconditional if CX=0
=00EA B309                              mov bl,ps_ciowait
=00EC 56                                push si
=00ED B91202E894FF  0087                mov cx,f_sleep ! call osif
=00F3 5E5A9D33C9                        pop si ! pop dx ! popf ! xor cx,cx
=00F8 EBD2          00CC                jmps cioattach
=00FA 891C              ca_atch:    mov c_attach[si],bx
=00FC 9D33DBC3          ca_ret: popf ! xor bx,bx ! ret
                        
                        ;===============
                        lstdetach_entry:
                        ;===============
=0100 E89AFF        009D        call lcb_verify         ;ensure U_LSTCCB is non zero
=0103 EB03          0108        jmps ciodetach
                        
                        ;===============
                        condetach_entry:
                        ;===============
=0105 E881FF        0089        call ccb_verify         ;ensure U_CONCCB is non zero
                                ;jmps ciodetach
                        
                        ;=========
                        ciodetach:              ;NOTE PIN also does detaching for ^P
                        ;=========
                        ;       Detach Calling process from default ciodev.
                        ;       If current owner of ciodev then zero c_attach
                        ;       and wakeup the c_queue list to give another
                        ;       process the ciodev.
                        ;       input:  SI = CCB addr of device
                        
=0108 881E6800                  mov bx,rlr
                                        ; SI = CCB address
```

```
 962
 963  =                                      ; BX = PD address
 964  =010C 9CFA                      pushf ! cli
 965  =010L 833C00741A    012D        cmp c_attach[si],0 ! je cd_ret
 966  =0113 391C7408      011F            cmp c_attach[si],bx ! je cd_detach
 967  =0117 9D8BFFFF                      popf ! mov bx,0ffffh
 968  =011B B92000C3                      mov cx,e_not_owner ! ret
 969  =                           cd_detach:
 970  =011F 8B4402                        mov ax,c_queue[si]
 971  =0122 8904                          mov c_attach[si],ax
 972  =0124 B91302                        mov cx,f_wakeup
 973  =0127 8D5402                        lea dx,c_queue[si]
 974  =012A E854FF        0087            call osif
 975  =                           cd_ret:
 976  =012D 9D                    popf
 977  =012E 33DBC3                xor bx,bx ! ret
 978  =
 979  =
 980  =                           ;===============
 981  =                           conassign_entry:
 982  =                           ;===============
 983  =                           ;       Attach specified console to specified process
 984  =                           ;
 985  =                           ;       entry:  U_WRKSEG:DX -> ACB
 986  =                           ;
 987  =                           ;       +-----+-----+-----+-----+
 988  =                           ;       | CNS |Match|  PD Addr  |
 989  =                           ;       +-----+-----+-----+-----+-----+-----+-----+-----+
 990  =                           ;       |                     Name                      |
 991  =                           ;       +-----+-----+-----+-----+-----+-----+-----+-----+
 992  =                           ;
 993  =                           ;               CNS     - console to assign
 994  =                           ;               Match   - if not 0,
 995  =                           ;                         PD's default console must match acb_cns
 996  =                           ;               PD Addr - PD address to assign to
 997  =                           ;               Name    - use PD name to assign, if PD addr=0
 998  =                           ;               Name    - use PD name to assign, if PD addr=1,
 999  =                           ;                         but also ensure DSKLD flag is off
1000  =                           ;                         (DiSK LoaD is set when PD was created
1001  =                           ;                         from disk transient, RSPs have DSKLD=0)
1002  =                           ;                         Should check for DSKLD=1 if PD addr = 0
1003  =                           ;                         but we may have compatibility issues
1004  =                           ;
1005  =                           ;       exit:   BX = 0 if success, 0ffffh if failure
1006  =                           ;               CX = Error Code
1007  =
1008  =0131 52                          push dx                         ;save ACB
1009  =0132 BB5906                        mov bx,offset thrd_spb          ;get ownership of thread
1010  =0135 B91502                        mov cx,f_sync
1011  =0138 E84CFF        0087            call osif
1012  =013B 5F                           pop di                          ;U_WRKSEG:DI->ACB
1013  =013C C6052206FF                   mov indisp,true                 ;stop dispatching
1014  =
```

```
1015
1016  =0141 1E                          push ds                         ;save SYSDAT
1017  =0142 268E1E2E00                  mov ds,u_wrkseg
1018  =0147 837D0201                    cmp acb_pd[di],1                ;name or address match ?
1019  =014B 761B          0168          jbe as_name
1020  =
1021  =014D 8B5D02                      mov bx,acb_pd[di]               ;find by PD address
1022  =0150 1F                          pop ds                          ;DS=SYSDAT
1023  =0151 BE7000                      mov si,(offset thrdrt)-p_thread
1024  =                           as_next1:                             ;verify PD is on thread
1025  =0154 8B7402                      mov si,p_thread[si]
1026  =0157 85F67428      0183          test si,si ! jz as_nom          ;no match if end of thread
1027  =015B 3B5E7402      0161            cmp bx,si ! je as_found_id
1028  =015F EBF3          0154            jmps as_next1
1029  =                           as_found_id:
1030  =0161 E83300        0217          call as_chk                     ;check console match
1031  =0164 E329          018F          jcxz as_mok                     ;match ok
1032  =0166 EB1B          0183            jmps as_nom                   ;no match
1033  =
1034  =                           as_name:                              ;look for PD by name on thread
1035  =0168 1F                        pop ds                          ;DS=SYSDAT
1036  =0169 BB7000                    mov bx,(offset thrdrt)-p_thread
1037  =                           as_next2:                             ;look for PD by name on thread
1038  =016C 57                        push di                         ;save ACB
1039  =016D 8D5504                    lea dx,acb_name[di]             ;U_WRKSEG:DI->ACB
1040  =0170 268E1E2E00                mov ds,u_wrkseg                 ;DS:DX->ASCII name
1041  =0175 B91402                    mov cx,f_findpdname
1042  =0178 E80CFF        0087        call osif
1043  =017B 2E8E1E0600                mov ds,sysdat
1044  =0180 5F                        pop di                          ;ACB
1045  =0181 E305          0188        jcxz as_okname                  ;CX<>0 if no match
1046  =
1047  =                           as_nom:
1048  =0183 B91400                      mov cx,e_no_pdname            ;couldn't find PD specified
1049  =0186 EB35          018D          jmps as_err                   ;by ACB on thread list
1050  =                           as_okname:
1051  =                                                                ;matched PD by name check by ACB
1052  =0188 E88C00        0217          call as_chk                   ;BX->PD matched by name
1053  =018B E302          018F          jcxz as_mok
1054  =                                                                ;console or DSKLD didn't match
1055  =018D E8DD          016C            jmps as_next2               ;search for another PD
1056  =
1057  =                           as_mok:                               ;match OK, found the PD
1058  =018F 5357                      push bx ! push di               ;BX=matched PD, DI=ACB
1059  =0191 BB5906                    mov bx,offset thrd_spb
1060  =0194 B91602                    mov cx,f_unsync
1061  =0197 E8EDFE        0087        call osif
1062  =019A 5F58                      pop di ! pop ax                 ;DI=ACB, AX=matched PD
1063  =019C 268E1E2E00                mov ds,u_wrkseg
1064  =01A1 50                        push ax                         ;save matched PD
1065  =01A2 8A05                      mov al,acb_cns[di]
1066  =01A4 2E8E1E0600                mov ds,sysdat
1067  =01A9 E8F100        029D        call getccbadr_spec             ;SI=CCB on return
```

```
1068
1069  =01AC 881E6800                mov bx,rlr                ;BX=running process
1070  =01B0 58                      pop ax                    ;AX=matched PD
1071  =
1072  =01B1 833C00                  cmp c_attach[si],0        ;and change CCB
1073  =01B4 7419            01CF    je as_ok                  ;CCB must be unused (0) or
1074  =01B6 391C                      cmp c_attach[si],bx     ;running process must own it
1075  =01B8 7415            01CF      je as_ok
1076  =01BA b92000                      mov cx,e_not_owner
1077  =                         as_err:
1078  =01BD 51                          push cx               ;save error code
1079  =01BE E84000           0201       call as_ok_disp
1080  =01C1 BB5906                      mov bx,offset thrd_spb
1081  =01C4 B91602                      mov cx,f_unsync
1082  =01C7 E83DFE           0087       call osif
1083  =01CA 59                          pop cx
1084  =01CB 88FFFF                      mov bx,0ffffh
1085  =01CE C3                          ret
1086  =                         as_ok:
1087  =01CF 8904                      mov c_attach[si],ax     ;new owner
1088  =01D1 5056                      push ax ! push si       ;save owner and CCB
1089  =01D3 E82800           0201     call as_ok_disp         ;allow dispatches
1090  =01D6 5E58                      pop si ! pop ax
1091  =01D8 8BD8                      mov bx,ax               ;AX=BX=new owner
1092  =01DA 83C602                    add si,c_queue-p_link   ;go down CCB queue for PD
1093  =01DD 8BFE                      mov di,si               ;DI=SI=address of CCB.QUEUE
1094  =                         as_nqpd:
1095  =01DF 833C00                    cmp p_link[si],0        ;look on C_QUEUE for PD being
1096  =01E2 741A             01FE     jz as_gexit             ;assigned the console
1097  =01E4 391C                        cmp p_link[si],bx
1098  =01E6 7404             01EC       je as_qfix
1099  =01E8 8B34                          mov si,p_link[si]
1100  =01EA E3F3             01DF       jmps as_nqpd          ;next PD
1101  =                         as_qfix:
1102  =01EC 8B07                          mov ax,p_link[bx]   ;take PD out of C_QUEUE
1103  =01EE 8904                          mov p_link[si],ax
1104  =01F0 8B35                          mov si,p_link[di]   ;SI = first PD on C_QUEUE
1105  =01F2 8919                          mov p_link[di],bx   ;put PD in front of C_QUEUE
1106  =01F4 8937                          mov p_link[bx],si   ;attach rest of list
1107  =01F6 8BD7                          mov dx,di           ;pass C_QUEUE address to wakeup
1108  =01F8 B91302                        mov cx,f_wakeup
1109  =01FB E889FE           0087         call osif
1110  =                         as_gexit:
1111  =01FE 33DB                      xor bx,bx
1112  =0200 C3                        ret
1113  =
1114  =                         as_ok_disp:
1115  =                         ;----------
1116  =                         ;       entry:  none
1117  =                         ;       exit:   none
1118  =                         ;       Allow dispatching
1119  =
1120  =0201 C606220600                mov indisp,false
```

```
1121
1122  =0206 9CFA                        pushf ! cli
1123  =0208 833E6C000074 0215           cmp drl,0 ! je as_nwake
1124        06           0215
1125  =020F B98E00                        mov cx,f_dispatch
1126  =0212 E872FE       0087             call osif
1127  =                           as_nwake:
1128  =0215 9DC3                        popf ! ret
1129  =
1130  =                           as_chk:
1131  =                           ;------
1132  =                           ;       check PD against Assign Control Block parameters
1133  =                           ;       entry:  BX = PD to check
1134  =                           ;               U_WRKSEG:DI = ACB
1135  =                           ;       exit:   CX = 0 if assign match
1136  =                           ;               CX <> 0 if no match
1137  =                           ;               CX,AX used
1138  =
1139  =0217 33C9                        xor cx,cx
1140  =0219 06                          push es                         ;save UDA
1141  =021A 268E062E00                  mov es,u_wrkseg
1142  =021F 26884502                    mov ax,es:acb_pd[di]
1143  =0223 3D0100                      cmp ax,1                        ;if >1 then matching on PD address
1144  =0226 770A         0232           ja asc_cns
1145  =0228 48                            dec ax
1146  =0229 7507         0232             jnz asc_cns
1147  =022B F747060020                      test p_flag[bx],pf_dskld    ;RSP, DSKLD must be off
1148  =0230 750F         0241                 jnz asc_no
1149  =                           asc_cns:
1150  =0232 2680780100                  cmp es:acb_match[di],0          ;console match requested ?
1151  =0237 7409         0242           je asc_ok
1152  =0239 268A05                        mov al,es:acb_cns[di]         ;must match on console
1153  =023C 3A4720                        cmp al,p_cns[bx]
1154  =023F 7401         0242             je asc_ok
1155  =                           asc_no:
1156  =0241 41                          inc cx
1157  =                           asc_ok:
1158  =0242 07                          pop es                          ;ES=UDA
1159  =0243 C3                          ret
1160  =
1161  =                           ;===============       ====================
1162  =                           getdefcon_entry:      ; Get Default Console
1163  =                           ;===============       ====================
1164  =
1165  =0244 8B366800                    mov si,rlr
1166  =0248 32FF8A5C20                  xor bh,bh ! mov bl,p_cns[si]
1167  =024D C3                          ret
1168  =
1169  =                           ;===============       ==================
1170  =                           getdeflst_entry:      ; Get Default List
1171  =                           ;===============       ==================
1172  =
1173  =024E 8B366800                    mov si,rlr
```

```
1174
1175  =0252 32FF8A5C24              xor bh,bh ! mov bl,p_lst[si]
1176  =0257 C3                      ret
1177  =
1178  =                         ;===============      ====================
1179  =                         setdefcon_entry:      ; Set Default Console
1180  =                         ;===============      ====================
1181  =
1182  =0258 3A168300                cmp dl,ncondev
1183  =025C 7207          0265      jb sd_good
1184  =025E B91300                    mov cx,e_ill_cns
1185  =0261 BBFFFFC3                  mov bx,0ffffh ! ret
1186  =                         sd_good:
1187  =0265 8B366800                mov si,rlr
1188  =0269 885420                  mov p_cns[si],dl
1189  =026C E82700        0296      call getccbadr
1190  =026F 2689366200              mov u_conccb,si
1191  =0274 33DB                    xor bx,bx
1192  =0276 C3                      ret
1193  =
1194  =                         ;===============      ==================
1195  =                         setdeflst_entry:      ; Set Default List
1196  =                         ;===============      ==================
1197  =
1198  =0277 3A168400                cmp dl,nlstdev
1199  =027B 7207          0284      jb sdl_good
1200  =027D B92500                    mov cx,e_ill_lst
1201  =0280 BBFFFFC3                  mov bx,0ffffh ! ret
1202  =                         sdl_good:
1203  =0284 8B366800                mov si,rlr
1204  =0288 885424                  mov p_lst[si],dl
1205  =028B E82600        02B4      call getlcbadr
1206  =028E 2689366400              mov u_lstccb,si
1207  =0293 33DB                    xor bx,bx
1208  =0295 C3                      ret
1209  =
1210  =
1211  =                         getccbadr:
1212  =                         ;---------
1213  =                         ; get CCB for calling process' console device
1214  =                         ;       input:  None
1215  =                         ;       output: SI = address of CCB
1216  =                         ;               BX = running PD address
1217  =
1218  =0296 8B1E6800                mov bx,rlr
1219  =029A 8A4720                  mov al,p_cns[bx]
1220  =
1221  =                         getccbadr_spec:                 ;AL is console number
1222  =029D 32E4                    xor ah,ah                 ;when called from ASSIGN
1223  =029F 8B365400                mov si,ccb
1224  =02A3 B102                    mov cl,2
1225  =02A5 D3E0                    shl ax,cl
1226  =02A7 03F0                    add si,ax                 ;+4
```

```
1227
1228  =02A9 03F0                    add si,ax                       ;+4 = 8
1229  =02AB 03F0                    add si,ax                       ;+4 =12
1230  =02AD B103                    mov cl,3
1231  =02AF D3E0                    shl ax,cl
1232  =02B1 03F0                    add si,ax                       ;+32 =44
1233  =02B3 C3                      ret
1234  =
1235  =                       getlcbadr:
1236  =                       ;---------
1237  =                       ; Get LCB for calling process list device
1238  =                       ;       entry:  None
1239  =                       ;       output: SI = address of LCB
1240  =                       ;               BX = PD address
1241  =
1242  =02B4 8B1E6800                mov bx,rlr
1243  =02B8 32E4                    xor ah,ah
1244  =02BA 8A4724                  mov al,p_lst[bx]
1245  =02BD 8B368600                mov si,lcb
1246  =02C1 50                      push ax                         ;lcb#
1247  =02C2 D1E0                    shl ax,1                        ;2*lcb#
1248  =02C4 03F0                    add si,ax                       ;lcbadr + 2*lcb#
1249  =02C6 58                      pop ax                          ;lcb#
1250  =02C7 B103                    mov cl,3
1251  =02C9 D3E0                    shl ax,cl                       ;8*lcb#
1252  =02CB 03F0                    add si,ax                       ;lcbadr + 2*lcb# + 8*lcb#
1253  =02CD C3                      ret
1254  =
1255  =
1256  =                       ;=======
1257  =                       cioterm:
1258  =                       ;=======
1259  =                       ;       Detach Calling process from all character devices.
1260  =                       ;       Called only from the dispatcher
1261  =
1262  =02CE A08300                  mov al,ncondev
1263  =02D1 8B355400                mov si,ccb
1264  =02D5 3C007426      02FF nxtccb: cmp al,0 | je ct_dlcb
1265  =02D9 263836620075  02F1          cmp si,u_conccb | jne ct_dccb
1266        11            02F1
1267  =02E0 806404C7                      and c_flag[si],not (cf_conout + cf_vout + cf_bufp)
1268  =02E4 891302                        mov cx,f_wakeup ;release proc waiting for XIOS conout
1269  =02E7 8D5422                        lea dx,c_cosleep[si]
1270  =02EA 5056                          push ax | push si
1271  =02EC E898FD        0087            call osif
1272  =02EF 5E58                          pop si | pop ax
1273  =                       ct_dccb:
1274  =02F1 5056                          push ax | push si
1275  =02F3 E812FE        0108            call ciodetach
1276  =02F6 5E58                          pop si | pop ax
1277  =02F8 FEC8                          dec al
1278  =02FA 83C62C                        add si,ccblen
1279  =02FD EBD6          02D5            jmps nxtccb
```

```
1280
1281  =
1282  =                         ct_dlcb:
1283  =02FF A03400                      mov al,nlstdev
1284  =0302 BB368600                    mov si,lcb
1285  =                         ct_nlcb:
1286  =0306 3C00740E      0318          cmp al,0 ! je ct_ret
1287  =030A 5056                        push ax ! push si
1288  =030C E8F9FD        0108          call cioudetach
1289  =030F 5E58                        pop si ! pop ax
1290  =0311 FEC8                        dec al
1291  =0313 83C60A                      add si,lcblen
1292  =0316 EBEE          0306          jmps ct_nlcb
1293  =
1294  =0318 33DBC3              ct_ret: xor bx,bx ! ret
1295
```

```
1296
1297  =                         eject 1 include chario.cio
1298  =                    ;*****************************************************
1299  =                    ;*
1300  =                    ;*      Character I/O Routines
1301  =                    ;*
1302  =                    ;*****************************************************
1303  =
1304  =                    ;============
1305  =                    cmode_entry:    ;Get/Set Console Mode
1306  =                    ;============
1307  =                    ;
1308  =                    ;       entry:  DX = 0FFFFH if returning current value
1309  =                    ;                       else DX = new console mode word
1310  =                    ;       return: BX = console mode if DX = 0FFFFH on entry
1311  =                    ;
1312  =031B 881E6800               mov bx,rlr
1313  =031F 427407         0329    inc dx 1 jz cm_get
1314  =0322 4A                         dec dx
1315  =0323 895722                     mov p_conmode[bx],dx
1316  =0326 33DB                       xor bx,bx
1317  =0328 C3                         ret
1318  =                    cm_get:
1319  =0329 8B5F22                 mov bx,p_conmode[bx]
1320  =032C C3                     ret
1321  =
1322  =                    ;============
1323  =                    conin_entry:    ;Function 1:console input
1324  =                    ;============
1325  =032D E86F02         05EF    call coninit
1326  =0330 E8E803         071B    call coninf                     ;char in AL and BL
1327  =0333 8AD0                   mov dl,al                       ;PIN handles control chars
1328  =                    co_notp:
1329  =0335 E8D9027203     0611    call echoc 1 jb ciexit          ;check nonprint char
1330  =033A E88303         06C0      call tabout                   ;echo character - if printable
1331  =                    ciexit:
1332  =033D 8ADAC3                 mov bl,dl 1 ret
1333  =
1334  =                    ;============
1335  =                    conout_entry:   ;Function 2:console output - DL = char
1336  =                    ;============
1337  =0340 E8AC02         05EF    call coninit
1338  =0343 F747220400             test p_conmode[bx],pcm_rout     ;BX=RLR PD
1339  =0348 7403           034D    jz co_tab
1340  =034A E90604         0753      jmp conoutf
1341  =                    co_tab:
1342  =034D E97003         06C0    jmp tabout
1343  =
1344  =                    ;============                   ;internal only under CCP/M
1345  =                    rconin_entry:   ; raw console input
1346  =                    ;============
1347  =0350 E89102         05E4    call rawinit
1348  =0353 E9C503         0718    jmp coninf              ;returns char in AL and BL
```

```
1349
1350  =
1351  =                         ;=============                    ;internal only under CCP/M
1352  =                         rconout_entry:  ; raw console output - DL = char
1353  =                         ;=============
1354  =0356 E88302        05E4          call rawinit
1355  =                         rconout:
1356  =0359 E9F703        0753          jmp conoutf
1357  =
1358  =                         ;=============
1359  =                         listout_entry:  ; write to list device - DL = char
1360  =                         ;=============
1361  =035C 881E68008A77                mov bx,rlr ! mov dh,p_lst[bx]
1362        24
1363  =0363 2688366400                  mov si,u_lstccb
1364  =0368 52E82BFD5A    0096          push dx ! call lstattach_entry ! pop dx
1365  =036D E9AB04        081B          jmp listf
1366  =
1367  =                         ;=============
1368  =                         constat_entry:          ;check console status
1369  =                         ;=============
1370  =0370 E87C02        05EF          call coninit                    ;console mode is made "cooked"
1371  =0373 E83C00        03B2          call cse_cl
1372  =0376 84DB7418      0392          test bl,bl ! jz co_r            ;BL=1 if char ready
1373  =037A 883E6800                        mov di,rlr
1374  =037E F745220100                      test p_conmode[di],pcm_ll
1375  =0383 7400          0392              jz co_r
1376  =0385 E89303        071B                  call coninf                     ;read the char
1377  =0388 3C037507      0393                  cmp al,ctlc ! jne co_nc         ;want only control C
1378  =038C C6440703                            mov c_nchar[si],ctlc
1379  =0390 B301                                mov bl,1
1380  =                         co_r:
1381  =0392 C3                          ret                             ;return BL=0 or 1
1382  =                         co_nc:
1383  =0393 32DBC3                      xor bl,bl ! ret         ;char not a ^C
1384  =
1385  =                         ;=======
1386  =                         ciostat:                ; check status w/out changing pf_raw
1387  =                         ;=======
1388  =0396 881E6800                    mov bx,rlr
1389  =039A 8A7720                      mov dh,p_cns[bx]
1390  =039D 2688366200                  mov si,u_conccb
1391  =03A2 85F6                        test si,si              ;U_CONCCB=0 if no attach yet
1392  =03A4 7404          03AA          jz cse_no
1393  =03A6 3B1C                          cmp bx,c_attach[si]
1394  =03A8 7408          03B2            je cse_cl
1395  =                         cse_no:
1396  =03AA B81EE8D8FC    0087              mov cl,f_dispatch ! call osif
1397  =03AF B300C3                          mov bl,0 ! ret
1398  =                         cse_cl:
1399  =03B2 E84703        06FC          call constf
1400  =03B5 84DB7402      03BB          test bl,bl ! jz cse_c2
1401  =03B9 B301                            mov bl,1
```

```
1402
1403  =                       cse_c2:
1404  =03BB 8AC3C3                  mov al,bl ! ret
1405  =
1406  =                       ;============
1407  =                       dirio_entry:                        ;function 6
1408  =                       ;============
1409  =                       ; direct console i/o - read if 0ffh
1410  =                       ;       entry:  DL = 0ffh if readif (ret=0 on not ready, else char)
1411  =                       ;                  = 0feh if status (ret=0 not read or 0ffh if ready)
1412  =                       ;                  = 0fdh if input  (wait there is a character)
1413  =                       ;                  = char to output
1414  =                       ;       exit:   AL = 0, 0ffh or character if status or input
1415  =
1416  =03BE E82302      05E4          call rawinit
1417  =03C1 80FAFD                    cmp dl,0fdh
1418  =03C4 7293        0359           jb rconout                  ;output DL
1419  =03C6 7703        03CB             ja diosts
1420  =03C8 E95003      0718               jmp coninf              ;DL=0FDh, get char
1421  =                       diosts:                             ;some type of status
1422  =03CB E82E03      06FC            call constf
1423  =03CE 80FAFE7501  03D4            cmp dl,0feh ! jne diocin   ;DL=0FEh, return status
1424  =                       diol:
1425  =03D3 C3                          ret                       ;AL=0FFh or 0h
1426  =                       diocin:
1427  =03D4 84D874F8    03D3            test bl,bl ! jz diol      ;DL=0FFh, if char,conin
1428  =03D8 E94303      071D              jmp coninf
1429  =
1430  =                       ;==============
1431  =                       conwrite_entry: ;write line until delimiter in UDA encountered
1432  =                       ;============== ;user function 9, delimiter set by function 110
1433  =                       ;
1434  =                       ;       input:  DX = buffer addr in u_wrkseg
1435  =
1436  =03DB 52                        push dx
1437  =03DC E81002      05EF          call coninit
1438  =03DF 5B                        pop bx
1439  =
1440  =03E0 1E268E1E2E00      cwr1:   push ds ! mov ds,u_wrkseg
1441  =03E6 8A171F                    mov dl,[bx] ! pop ds
1442  =03E9 263A16660074 0408         cmp dl, u_delim ! je cwr_e
1443        18           0408
1444  =03F0 53                            push bx
1445  =03F1 881E6800                      mov bx,rlr
1446  =03F5 F74722040074 0401             test p_conmode[bx],pcm_rout ! jz cw_tab
1447        05           0401
1448  =03FC E85403E803   0753                 call conoutf ! jmps cw_notab
1449  =                       cw_tab:
1450  =0401 E8BC02      06C0              call tabout
1451  =                       cw_notab:
1452  =0404 5B                          pop bx
1453  =0405 43E8D8      03E0            inc bx ! jmps cwr1
1454  =0408 C3                cwr_e:  ret
```

```
1455
1456  =
1457  =                         ;===========
1458  =                         delim_entry:    ;get/set output delimiter, user function 110
1459  =                         ;===========
1460  =                         ;
1461  =                         ;       entry:  DX = 0FFFFH if returning current delimiter
1462  =                         ;                    else DL = new delimiter
1463  =                         ;       return: BL = delimiter if DX = 0FFFFH on entry
1464  =
1465  =0409 83FAFF7406    0414         cmp dx,0FFFFH ! jz d_get
1466  =040E 2688166600                  mov u_delim,dl
1467  =0413 C3                          ret
1468  =                         d_get:
1469  =0414 268A1E6600                  mov bl,u_delim
1470  =0419 C3                          ret
1471  =
1472  =                         ;=============
1473  =                         p_block_entry:
1474  =                         ;=============
1475  =                         ;       entry:  DX = CB addr where 1st word is offset,
1476  =                         ;                    2nd word is segment, 3rd word
1477  =                         ;                    is length
1478  =                         ;       return: None
1479  =
1480  =041A 52                          push dx
1481  =041B E8D101        05EF          call coninit            ;BX=PD address
1482  =041E 5F                          pop di
1483  =041F 33C0                        xor ax,ax
1484  =0421 8B6F22                      mov bp,p_conmode[bx]
1485  =0424 EB0F          0435          jmps block_out
1486  =
1487  =                         ;=============
1488  =                         l_block_entry:
1489  =                         ;=============
1490  =                         ;       return: None
1491  =                         ;       entry:  DX = CB addr where 1st word is offset,
1492  =                         ;                    2nd word is segment, 3rd word
1493  =                         ;                    is length
1494  =
1495  =0426 52                          push dx
1496  =0427 E85CFC        0096          call lstattach_entry
1497  =042A 5F                          pop di
1498  =042B 8B1E68008A77                mov bx,rlr ! mov dh,p_lst[bx]
1499        24
1500  =0432 B8FFFF                      mov ax,0ffffh
1501  =                                 ;jmps block_out
1502  =                                                 ;SI -> CCB or LCB (in XIOS)
1503  =                         block_out:              ;u_wrkseg:DI -> user's parameter block
1504  =                                                 ;DH=list or console device number
1505  =                                                 ;AX = 0 if console, 0FFFFH if list
1506  =0435 1E268E1E2E00                push ds ! mov ds,u_wrkseg
1507  =043B 8B4D04                      mov cx,cb_len[di]           ;CX = string length
```

```
1508
1509  =043E 8B5702                    mov bx,cb_seg[di]
1510  =0441 8B3D                      mov di,cb_off[di]               ;BX:DI = string start
1511  =0443 1F                        pop ds
1512  =0444 85C974zA      0472        test cx,cx ! jz b_ret           ;test for zero length
1513  =
1514  =                         b_loop:                           ;BX:DI -> string to print or list
1515  =                                                           ;AX = 0 if to console = 0FFFFH if to list
1516  =                                                           ;BP = p_conmode if going to console
1517  =0446 1E8EDB                    push ds ! mov ds,bx
1518  =0448 8A15                      mov dl,[di]
1519  =044D 1F                        pop ds
1520  =044E 5557515350                push bp ! push di ! push cx ! push bx ! push ax
1521  =0453 85C07405      045C        test ax,ax ! jz b_con
1522  =0457 E8C103E80E    0818            call listf ! jmps nb_nxt
1523  =                         b_con:
1524  =045C F7C504007505  0467            test bp,pcm_rout ! jnz nb_nt        ;raw output mode ?
1525  =0462 E85802E803    06C0            call tabout ! jmps nb_nxt           ;test for ctrlP ?
1526  =                         nb_nt:
1527  =0467 E8E902        0753        call conoutf
1528  =                         nb_nxt:
1529  =046A 5858595F5D                pop ax ! pop bx ! pop cx ! pop di ! pop bp
1530  =046F 47                        inc di
1531  =0470 E2D6          0448        loop b_loop
1532  =                         b_ret:
1533  =0472 C3                        ret
1534  =
1535  =
1536  =                         ;===============
1537  =                         conprint_entry: ;write line until delimiter or max
1538  =                         ;=============== ;mask parity, do not check for ctrl chars
1539  =                         ;
1540  =                         ;       input:  DX = address of buffer in u_wrkseg
1541  =                         ;               BH = max characters (0=no max)
1542  =                         ;               BL = delimiter character
1543  =
1544  =0473 5352                      push bx ! push dx
1545  =0475 E87701        05EF        call coninit
1546  =0478 5B58                      pop bx ! pop ax
1547  =047A 33C9                      xor cx,cx
1548  =047C 247F                      and al,07fh
1549  =                                       ; AH=max, AL=delimiter, BX->buffer
1550  =                                       ; CX = count = 0
1551  =047E 80FC007404    0487 cpr1:  cmp ah,0 ! je cpr2
1552  =0483 3AE17421      04A8        cmp ah,cl ! je cpr_e
1553  =0487 1E268E1E2E00      cpr2:       push ds ! mov ds,u_wrkseg
1554  =048D 8A171F                        mov dl,[bx] ! pop ds
1555  =0490 80E27F                        and dl,07fh
1556  =0493 3AD07411      04A8 cpr3:      cmp dl,al ! je cpr_e
1557  =0497 505153                        push ax ! push cx ! push bx
1558  =049A E8B602        0753            call conoutf
1559  =049D E8E701        0687            call chk_ctrlP
1560  =04A0 5B5958                        pop bx ! pop cx ! pop ax
```

```
1561
1562  =04A3 FEC143                          inc cl ! inc bx
1563  =04A6 EBD6          047E              jmps cpr1
1564  =04A8 C3            cpr_e:  ret
1565  =
1566  =                   ;==============
1567  =                   conread_entry:  ;read a buffered console line
1568  =                   ;==============
1569  =                   ; buffer=(max length, current length, buffer)
1570  =
1571  =04A9 52                  push dx
1572  =04AA E84201        05EF  call coninit
1573  =04AD 5B                  pop bx
1574  =04AE 53            read:   push bx                     ;[SP] = buffer offset
1575  =04AF 8A4406                mov al,c_column[si]
1576  =04B2 884405                mov c_strtcol[si],al        ;start column = current column
1577  =04B5 33DB                  xor bx,bx                   ;BX = character count
1578  =
1579  =                               ;read next character, CX, BX active
1580  =04B7 53            readnx: push bx
1581  =04B8 E85002        071B readn0: call coninf     ;AL, BL = char just read
1582  =04BB 8AD0                  mov dl,al
1583  =04BD 5B                    pop bx
1584  =
1585  =                               ;       Carriage Return
1586  =04BE 80FA0D7503    04C6        cmp dl,cr ! jnz notcr
1587  =04C3 E90901        05CF eofre:     jmp readen
1588  =                   notcr:
1589  =                               ;       Line Feed
1590  =04C6 80FA0A74F8    04C3        cmp dl,lf ! jz eofre
1591  =
1592  =                   tryh:       ;       Backspace
1593  =04CB 80FA087513    04E3        cmp dl,ctlh ! jnz noth
1594  =04D0 0AD374E3 r    04B7            or bl,bl ! jz readnx                ;if not 1st char
1595  =04D4 FEC88A4406                    dec bl ! mov al,c_column[si]        ;decrement char count
1596  =04D9 88440C                        mov c_btmp[si],al                   ;temporary hold
1597  =04DC 804C0402E96D  0550            or c_flag[si],cf_compc ! jmp linelen
1598        00            0550
1599  =                   noth:
1600  =                               ;       Rubout
1601  =
1602  =04E3 80FA7F7517    04FF        cmp dl,rubout ! jnz notrub
1603  =04E8 0AD87410      04FC            or bl,al ! jz readnxx               ;only if not 1st char
1604  =04EC 5F57                          pop di ! push di
1605  =04EE 1E258E1E2E00                  push ds ! mov ds,u_wrkseg
1606  =04F4 8A51011F                      mov dl,1[di+bx] ! pop ds            ;make char=last char
1607  =04F8 4B                            dec bx                              ;dec char count
1608  =04F9 E9BE00        05BA            jmp rdech1                          ;echo, read next
1609  =04FC E9B8FF        04B7 readnxx:   jmp readnx
1610  =                   notrub:
1611  =                               ;       Control E
1612  =
1613  =04FF 80FA057516    051A        cmp dl,ctle ! jnz note
```

```
1614
1615  =0504 53                              push bx
1616  =0505 8A20E84902    0753              mov dl,cr ! call conoutf
1617  =050A 8A20E84402    0753              mov dl,lf ! call conoutf
1618  =050F C6440600                        mov c_column[si],0
1619  =0513 C6440500                        mov c_strtcol[si],0
1620  =0517 E99EFF        04B8              jmp readn0
1621  =                         note:
1622  =                                     ;       Control X
1623  =051A 80FA187520    053F          cmp dl,ctlx ! jnz notx
1624  =051F 8A4405              backx:      mov al,c_strtcol[si]
1625  =0522 3A44067204    052B              cmp al,c_column[si] ! jb backxxx
1626  =0527 33DBEBD1      04FC                xor bx,bx ! jmps readnxx
1627  =052B 8A20E82302    0753 backxxx:     mov dl,ctlh ! call conoutf
1628  =0530 8A20E81E02    0753              mov dl,' ' ! call conoutf
1629  =0535 8A20E81902    0753              mov dl,ctlh ! call conoutf
1630  =053A FE4C06                          dec c_column[si]
1631  =053D EBE0          051F              jmps backx
1632  =                         notx:
1633  =                                     ;       Control U
1634  =053F 80FA157507    054B          cmp dl,ctlu ! jnz notu
1635  =0544 E89D01        06E4              call crlfp
1636  =0547 58E963FF      04AE              pop bx ! jmp read
1637  =                         notu:
1638  =                                     ;       CONTROL R - redraw line
1639  =054B 80FA12755D    05AD          cmp dl,ctlr ! jnz notr
1640  =0550 53E89001      06E4 linelen:     push bx ! call crlfp
1641  =                                       ;print buffer on screen
1642  =0554 5933DB                          pop cx ! xor bx,bx
1643  =0557 0AC9741C      0577 rep0:        or cl,cl ! jz rep1
1644  =055B 5F57                              pop di ! push di
1645  =055D 1E268F1E2E00                      push ds ! mov ds,u_wrkseg
1646  =0563 43FEC9                            inc bx ! dec cl
1647  =0566 8A51011F                          mov dl,1[di+bx] ! pop ds        ;DL = nxt char
1648  =056A 8AE851                            mov ch,bl ! push cx
1649  =056D E84101        06B1                call ctlout
1650  =0570 5987008ADD                        pop cx ! mov bh,0 ! mov bl,ch
1651  =0575 EBE0          0557                jmps rep0
1652  =0577 F6440402            rep1:       test c_flag[si],cf_compc
1653  =057B 53742C        05AA                push bx ! jz l_22
1654  =057E 806404FD                          and c_flag[si],not cf_compc
1655  =0582 8A640C8AC4                        mov ah,c_btmp[si] ! mov al,ah
1656  =0587 3A4406              backsp:       cmp al,c_column[si]
1657  =058A 721E          05AA                jb l_22
1658  =058C 2A4406                            sub al,c_column[si]
1659  =058F 0AC07417      05AA                or al,al ! jz l_22
1660  =0593 50                                    push ax
1661  =0594 8A20E88A01    0753                    mov dl,ctlh ! call conoutf
1662  =0599 8A20E88501    0753                    mov dl,' ' ! call conoutf
1663  =059E 8A20E88001    0753                    mov dl,ctlh ! call conoutf
1664  =05A3 58FECC8AC4                            pop ax ! dec ah ! mov al,ah
1665  =05A8 EBDD          0587                    jmps backsp
1666  =05AA E903FF        04B8 l_22:        jmp readn0
```

```
1667
1668  =                         notr:
1669  =                                 ;       All other characters
1670  =05AD 43                  rdecho:     inc bx
1671  =05AE 5F57                            pop di ! push di
1672  =05B0 1E268E1E2E00                    push ds ! mov ds,u_wrkseg
1673  =05B6 8851011F                        mov 1[di+bx],dl ! pop ds
1674  =05BA 53E8F3005B    06B1  rdech1:     push bx ! call ctlout ! pop bx
1675  =05BF 5F57                            pop di ! push di
1676  =05C1 1E268E1E2E00                    push ds ! mov ds,u_wrkseg
1677  =05C7 3A1D1F7303    05CF              cmp bl,[di] ! pop ds ! jae readen
1678  =05CC 53EBDB        05AA                  push bx ! jmps l_22
1679  =                                     ;       End of Console String
1680  =05CF 5F                  readen: pop di
1681  =05D0 1E268E1E2E00                push ds ! mov ds,u_wrkseg
1682  =05D6 885D011F                    mov 1[di],bl ! pop ds
1683  =05DA B20DE87401    0753          mov dl,cr ! call conoutf
1684  =05DF C6440600                    mov c_column[si],0
1685  =05E3 C3                          ret
1686  =
1687  =                         rawinit:
1688  =                         ;-------
1689  =05E4 8B1E6800                    mov bx,rlr              ;turn on raw flag
1690  =05E8 814F064000                  or p_flag[bx],pf_raw
1691  =05ED EB09          05F8          jmps ninit
1692  =
1693  =                         coninit:
1694  =                         ;-------
1695  =                         ;       entry:  DL = char if called by output routine
1696  =                         ;       exit:   DH = console device number
1697  =                         ;               BX = RLR PD - raw flag turned off
1698  =                         ;               SI = CCB owned by calling PD
1699  =
1700  =05EF 8B1E6800                    mov bx,rlr              ;turn off raw flag
1701  =05F3 8167068FFF                  and p_flag[bx],not pf_raw
1702  =                         ninit:
1703  =05F8 8A7720                      mov dh,p_cns[bx]
1704  =05FB 268B366200                  mov si,u_conccb
1705  =0600 85F67405      0609          test si,si ! jz attach
1706  =0604 391C                          cmp c_attach[si],bx
1707  =0606 7501          0609            jne attach
1708  =0608 C3                              ret
1709  =
1710  =                         attach:
1711  =                         ;       exit:   SI=CCB
1712  =                         ;               BX,DX preserved
1713  =
1714  =0609 5352                            push bx ! push dx
1715  =060B E8A4FA        0082              call conattach_entry
1716  =060E 5A5B                            pop dx ! pop bx
1717  =0610 C3                              ret
1718  =
1719  =                         echoc:
```

```
1720
1721  =                         ;-----
1722  =                         ; check if character is graphic
1723  =                         ;       input:  DL = character
1724  =                         ;               DH = console #
1725  =                         ;               SI = CCB address
1726  =                         ;       output: carry set if not graphic
1727  =                         ;               DL,DH,SI pass through
1728  =
1729  =0611 80FA0D7412    0628         cmp dl,cr ! je ret0
1730  =0616 80FA0A740D    0628         cmp dl,lf ! je ret0
1731  =0618 80FA097408    0628         cmp dl,tab ! je ret0
1732  =0620 80FA087403    0628         cmp dl,ctln ! je ret0
1733  =0625 80FA20                     cmp dl,' '
1734  =0628 C3                 ret0:   ret
1735  =
1736  =                         prcsmsg:
1737  =                         ;-------
1738  =                         ; print string in Code Segment
1739  =                         ;       input:  DX->null terminated string in CS
1740  =
1741  =0629 33DB8CC8                   xor bx,bx ! mov ax,cs
1742  =0620 26FF362E00         prstr:  push u_wrkseg
1743  =0632 26A32E00                   mov u_wrkseg,ax
1744  =0636 E83AFE        0473         call conprint_entry
1745  =0639 268F062E00C3               pop u_wrkseg ! ret
1746  =
1747  =                         prname:
1748  =                         ;------
1749  =                         ; print name
1750  =                         ;       input:  DX->8 char name in DS
1751  =
1752  =063F B7088320                   mov bh,8 ! mov bl,' '
1753  =0643 8CD8EBE6      062D         mov ax,ds ! jmps prstr
1754  =
1755  =
1756  =                         conout:
1757  =                         ;------
1758  =                         ; compute character position/write console
1759  =                         ;       input:  DL = character
1760  =                         ;               DH = console #
1761  =                         ;               SI = ccb address
1762  =                         ;       output: DL = character
1763  =                         ;               DH = console #
1764  =                         ;               SI = ccb address
1765  =
1766  =                                 ;check if only calculating column position
1767  =0647 F64404027511 065E          test c_flag[si],cf_compc ! jnz compout
1768  =                                 ;write the character, then compute column.
1769  =064D E80301        0753             call conoutf
1770  =0650 8B1E6800                       mov bx,rlr
1771  =0654 F747220400                     test p_conmode[bx],pcm_rout
1772  =0659 7503          065E             jnz co_np
```

```
1773
1774  =0658 E82900          0687              call chk_ctrlP          ;printer echo handling
1775  =                           co_np:
1776  =
1777  =                                       ;compute column position
1778  =                           compout:
1779  =065B 8A4406                        mov al,c_column[si]
1780  =065E 80FA7F7410      0683          cmp dl,rubout ! je ccret
1781  =0663 FEC0                          inc al
1782  =0665 80FA207316      0683          cmp dl,' ' ! jae ccret
1783  =066A FEC8                          dec al
1784  =066C 0AC07410        0683          or al,al ! jz ccret
1785  =0670 80FA087504      067C              cmp dl,ctlh ! jnz notbs
1786  =0675 FEC8E807        0683                  dec al ! jmps ccret
1787  =067A 80FA0D7502      0683 notbs:           cmp dl,cr ! jne ccret
1788  =067E B000                                  mov al,0
1789  =0680 884406                        ccret:  mov c_column[si],al
1790  =0683 C3                            ret
1791  =
1792  =
1793  =                           chk_ctrlP:
1794  =                           ;---------
1795  =                           ;       Entry:  SI=CCB address
1796  =                           ;               DH=console device number
1797  =                           ;               DL=character just sent to console
1798  =                           ;       Exit:   SI,DX preserved
1799  =
1800  =                           ;       Echo the character to the printer if the ctrlP flag is set
1801  =                           ;       in the CCB, and the ctrl0 flag is off, and the cf_bufp
1802  =                           ;       flag is off.  Cf_bufp is set when we send a character
1803  =                           ;       to a VOUT process when we are a background buffered
1804  =                           ;       console.  The VOUT process decides whether to send the
1805  =                           ;       character to the screen when it purges the buffered output.
1806  =
1807  =0683 8B440E                        mov ax,c_state[si]
1808  =0686 A90002                        test ax,csm_ctrlP
1809  =0689 7421            068C          jz cp_ret
1810  =068B A90001                          test ax,csm_ctrl0
1811  =068E 751C            068C            jnz cp_ret
1812  =0690 8A4404                            mov al,c_flag[si]
1813  =0693 A820                              test al,cf_bufp
1814  =0695 7406            069D              jz cp_listit
1815  =0697 806404DF                            and c_flag[si],not cf_bufp      ;turn off the flag
1816  =069B EB0F            068C                jmps cp_ret
1817  =                           cp_listit:
1818  =069D 5255                               push dx ! push si
1819  =069F 8A7408                             mov dh,c_mimic[si]
1820  =06A2 80FEFF                             cmp dh,0ffh                     ;mimic = 0ffh if
1821  =06A5 7403            06AA               je cp_no                        ;PIN turned off ^P
1822  =06A7 E86D01          0817                 call listf
1823  =                           cp_no:
1824  =06AA 5E5A                               pop si ! pop dx
1825  =                           cp_ret:
```

```
1826
1827  =06P0 C3                          ret
1828  =
1829  =                           ctlout:
1830  =                           ;------
1831  =                           ; send CTRL character with possible preceding up-arrow
1832  =                           ;       input:  DL = character
1833  =                           ;               DH = console #
1834  =                           ;               SI = ccb address
1835  =                           ;       output: DL,DH = character,console
1836  =                           ;               SI = ccb address
1837  =
1838  =06B1 E850FF730A    0611            call echoc ! jae tabout
1839  =06B6 52B25EE88BFF  0647                push dx ! mov dl,ctl ! call conout
1840  =06BC 5A80CA40                          pop dx ! or dl,40h
1841  =                                       ; jmp tabout
1842  =
1843  =                           tabout:
1844  =                           ;------
1845  =                           ; expand tabs to console
1846  =                           ;       input:  DL = character
1847  =                           ;               DH = console
1848  =                           ;               SI = ccb address
1849  =                           ;       output: DL,DH = char,console
1850  =                           ;               SI = ccb address
1851  =
1852  =
1853  =06C0 80FA097403    06C8            cmp dl,tab ! je tab1
1854  =06C5 E97FFF        0647              jmp conout
1855  =                           tab1:
1856  =06C8 B220                              mov dl,' '
1857  =06CA E87AFF        0647    tab0:       call conout
1858  =06CD 8A4406                            mov al,c_column[si]
1859  =06D0 240775F6      06CA                and al,111b ! jnz tab0
1860  =06D4 B209                              mov dl,tab
1861  =06D6 C3                    ret2:       ret
1862  =
1863  =                           crlf:
1864  =                           ;----
1865  =06D7 52B20DE86AFF  0647            push dx ! mov dl,cr ! call conout
1866  =06DD B20AE865FF5A  0647            mov dl,lf ! call conout ! pop dx
1867  =06E3 C3                            ret
1868  =                           crlfp:
1869  =                           ;-----
1870  =                           ; print #, cr, lf for ctlx, ctlu, ctlr functions
1871  =                           ; then print ' ' until we are at strtcol (starting column)
1872  =                           ;       input:  DH = console
1873  =                           ;               SI = ccb address
1874  =                           ;       output: DH = console
1875  =                           ;               SI = ccb address
1876  =
1877  =06E4 B223E85EFF    0647            mov dl,'#' ! call conout
1879  =06E9 E8E8FF        06D7            call crlf
```

```
1879
1880  =06EC B220                    mov dl,' '
1881  =06EE 8A4406              crlfp0: mov al,c_column[si]
1882  =06F1 3A44057305    06F8        cmp al,c_strtcol[si] ! jae crlfp1
1883  =06F6 E84EFFEBF3    0647        call conout ! jmps crlfp0
1884  =06FB C3                  crlfp1: ret
1885  =
1886  =                             ;
1887  =                             ;       XIOS CHARACTER I/O INTERFACE
1888  =                             ;
1889  =
1890  =
1891  =                             ;       XIOS CONSTAT CALL
1892  =                             ;
1893  =                             ;       entry:  DH = console #
1894  =                             ;               SI = ccb address
1895  =                             ;       returns:DH = console #
1896  =                             ;               SI = ccb address
1897  =                             ;               AX = BX = 0 if no char, = 0FFH if char ready
1898  =
1899  =                         constf:
1900  =06FC 3A354700                  cmp dh,ncns                     ;if not VC goto XIOS
1901  =0700 7205          0707        jb cst_vc
1902  =0702 8000                        mov al,io_const
1903  =0704 E91701        081E          jmp goxios
1904  =                         cst_vc:
1905  =0707 33D3                      xor bx,bx
1906  =0709 385C07                    cmp c_nchar[si],bl
1907  =070C 7508          0716        jne cst_true
1908  =070E 8B7C12                      mov di,c_vinq[si]
1909  =0711 395D16                      cmp q_msgcnt[di],bx             ;number of chars in type ahead
1910  =0714 7402          0718        je cst_ret                      ;queue
1911  =                         cst_true:
1912  =0716 B3FF                          mov bl,0ffh                 ;char ready
1913  =                         cst_ret:
1914  =0718 8AC3                      mov al,bl
1915  =071A C3                        ret
1916  =
1917  =
1918  =                             ;       XIOS CONIN CALL
1919  =                             ;
1920  =                             ;       entry:  DH = console #
1921  =                             ;               SI = ccb address
1922  =                             ;       returns:DH = console #
1923  =                             ;               AL = BL = character
1924  =                             ;               SI = ccb address
1925  =                             ;
1926  =                         coninf:
1927  =071B 3F354700                  cmp dh,ncns                     ;if not VC goto XIOS
1928  =071F 7205          0726        jb ci_vc
1929  =0721 8001                        mov al,io_conin
1930  =0723 E9F800        081E          jmp goxios
1931  =                         ci_vc:
```

```
1932
1933  =0726 807C0700              cmp c_nchar[si],0
1934  =072A 7505          0732    jne ci_gotc
1935  =072C 698300                    mov cx,f_qread
1936  =072F E80400        073C        call csi_readq
1937  =                       ci_gotc:
1938  =0732 8A5C07                mov bl,c_nchar[si]
1939  =0735 8AC3                  mov al,bl
1940  =0737 C6440700              mov c_nchar[si],0
1941  =073B C3                    ret
1942  =
1943  =                       csi_readq:
1944  =073C 5256                  push dx ! push si
1945  =073E 8B4412                mov ax,c_vinq[si]
1946  =0741 89441A                mov c_qpbqaddr[si],ax
1947  =0744 8D4407                lea ax,c_nchar[si]
1948  =0747 89441E                mov c_qpbbuffptr[si],ax
1949  =074A 8D5418                lea dx,c_qpbflags[si]
1950  =074D E837F9        0087    call osif
1951  =0750 5E5A                  pop si ! pop dx
1952  =0752 C3                    ret
1953  =
1954  =
1955  =                       ;       XIOS CONOUT CALL
1956  =                       ;
1957  =                       ;       entry:  DH = console #
1958  =                       ;               DL = character
1959  =                       ;               SI = ccb address
1960  =                       ;       exit:   DH = console #
1961  =                       ;               SI = ccb address
1962  =
1963  =                       conoutf:
1964  =0753 3A364700              cmp dh,ncns                     ;if not VC goto XIOS
1965  =0757 7205          075E    jb co_vc
1966  =0759 B002                      mov al,io_conout            ;console number
1967  =075B E9C000        081E        jmp goxios                  ;control C could hang up this
1968  =                       co_vc:                              ;hardware, should handle similar
1969  =075E 9CFA                  pushf ! cli                     ;to virtual screens
1970  =0760 8B440E                mov ax,c_state[si]
1971  =0763 8BD8                  mov bx,ax
1972  =0765 25F7FC                and ax,not(csm_ctrlP + csm_ctrlO + csm_noswitch)
1973  =0768 3D0200                cmp ax,2                        ;0=foreground&dynamic, 1=foreground&
1974  =076B 763F          07AC    jbe co_getconout                ;buffered, 2=background&dynamic
1975  =
1976  =076D 3D0300                 cmp ax,csm_buffered + csm_background
1977  =0770 7412          0784      je co_buffer
1978  =                                                           ;sleep on all other states
1979  =0772 5256                    push dx ! push si             ;and wait for the PIN or
1980  =0774 8D5424                  lea dx, c_usleep[si]          ;VOUT to wake us up
1981  =0777 B309                    mov bl, ps_ciowait
1982  =0779 B91202                  mov cx, f_sleep
1983  =077C E808F9        0087      call osif
1984  =077F 5E5A                    pop si ! pop dx
```

```
1985
1986  =0781 9D                          popf
1987  =0782 EBDA          075E          jmps co_vc                  ;start over, state could
1988  =                                                             ;have changed
1989  =
1990  =                           co_buffer:              ;Virtual Console is in some type of buffered state
1991  =0784 9D                          popf                        ;allow interrupts
1992  =0785 804C0430                    or c_flag[si],cf_vout+cf_bufp   ;VOUT we are writing a msg.
1993  =                                                             ;If we switch screens, and VOUT
1994  =                                                             ;just finishes purging, VOUT
1995  =                                                             ;will wait for this message.
1996  =                                                             ;Turn on cf_bufp so we don't
1997  =                                                             ;echo to the printer if ctrlP is o
1998  =0789 8B4414                      mov ax,c_voutq[si]
1999  =078C 89441A                      mov c_qpbqaddr[si],ax
2000  =078F 8D4420                      lea ax,c_qbuff[si]          ;points to 2-byte message
2001  =0792 89441E                      mov c_qpbbuffptr[si],ax
2002  =0795 5256                      push dx ! push si
2003  =0797 32F6                      xor dh,dh                   ;DL = data, DH = data msg type (0)
2004  =0799 895420                    mov c_qbuff[si],dx
2005  =079C 8D5418                    lea dx,c_qpbflags[si]
2006  =079F B98800                    mov cx, f_qwrite
2007  =07A2 E8E2F8        0087        call osif
2008  =07A5 5E5A                      pop si ! pop dx
2009  =07A7 806404EF                  and  c_flag[si],not cf_vout ;tell VOUT we're done writing
2010  =07AB C3                        ret                         ;message
2011  =
2012  =                           co_getconout:
2013  =07AC F7C30001                  test bx,csm_ctrl0           ;BX = ccb.state
2014  =07B0 7402          07B4        jz co_cbitsget              ;famous racehorse
2015  =07B2 9DC3                        popf ! ret                ;ctrl 0 byte bucket
2016  =                           co_cbitsget:                    ;output is going to XIOS,
2017  =                                                           ;ensure no other process
2018  =07B4 F6440408                  test c_flag[si],cf_conout   ;is in XIOS console output code
2019  =07B8 7413          07CD        jz co_gotconout             ;for this screen
2020  =
2021  =                           co_conoutsleep:
2022  =07BA 5256                        push dx ! push si
2023  =07BC 8D5422                      lea dx, c_cosleep[si]
2024  =07BF B309                        mov bl, ps_ciowait
2025  =07C1 B91202                      mov cx, f_sleep
2026  =07C4 E8C0F8        0087          call osif
2027  =07C7 5E5A                        pop si ! pop dx
2028  =07C9 9D                          popf
2029  =07CA E991FF        075E          jmp co_vc                   ;start over, state
2030  =                                                           ;could have changed
2031  =                           co_gotconout:
2032  =07CD 804C0408                    or c_flag[si], cf_conout    ;set XIOS conout "ownership" flag
2033  =07D1 9D                          popf                        ;allow interrupts
2034  =07D2 8B1E6800                    mov bx,rlr
2035  =07D6 FF7706                      push p_flag[bx]             ;save flags
2036  =07D9 814F060002                  or p_flag[bx],pf_tempkeep   ;don't allow abort
2037  =07DE 53                          push bx
```

```
2038
2039  =07DF B80200              mov ax,io_conout              ;XIOS function number
2040  =07E2 E83900       081E   call goxios
2041  =07E5 806404F7            and c_flag[si], not cf_conout ;wake any process wanting XIOS
2042  =07E9 837C2200            cmp c_cosleep[si],0           ;test here for faster output
2043  =07ED 740F         07FE   je co_nowake
2044  =07EF 5256                    push dx ! push si
2045  =07F1 891302                  mov cx,f_wakeup
2046  =07F4 8A09                    mov bl,ps_ciowait
2047  =07F6 8D5422                  lea dx,c_cosleep[si]
2048  =07F9 E88BF8       0087       call osif
2049  =07FC 5E5A                    pop si ! pop dx
2050  =                       co_nowake:
2051  =07FE 5B                      pop bx                    ;process address
2052  =07FF F747068000              test p_flag[bx],pf_ctlc   ;was ^C typed ?
2053  =0804 8F4706                  pop p_flag[bx]            ;restore PD flags
2054  =0807 7411         081A       jz  co_xret
2055  =0809 814F068000              or p_flag[bx],pf_ctlc
2056  =080E 5256                    push dx ! push si
2057  =0810 33D2                    xor dx,dx
2058  =0812 B98F00                  mov cx,f_terminate
2059  =0815 E86FF8       0087       call osif                 ;returns here if keep flg is on
2060  =0818 5E5A                    pop si ! pop dx
2061  =                       co_xret:
2062  =081A C3                      ret
2063  =
2064  =                       ;     XIOS LIST OUT CALL
2065  =                       ;
2066  =                       ;     entry:  DH = console #
2067  =                       ;             DL = character
2068  =                       ;             SI = ccb address
2069  =                       ;     exit:   DH = console #
2070  =                       ;             SI = ccb address
2071  =
2072  =                       listf:
2073  =081B B80400                  mov ax,io_list
2074  =                             ;jmps goxios
2075  =
2076  =                       goxios:
2077  =081E 5256                    push dx ! push si
2078  =0820 8AC88AD6                mov cl,dl ! mov dl,dh     ;CL = char, DL = device
2079  =0824 E865F8       008D       call xiosif
2080  =0827 5E5A                    pop si ! pop dx
2081  =0829 C3                      ret
2082
```

```
2083
2084  =                         eject 1 include patch.cod
2085  =                       ;*******************************************************
2086  =                       ;*
2087  =                       ;*      PATCH AREA  --  128 bytes long
2088  =                       ;*
2089  =                       ;*******************************************************
2090  =
2091  =                       patch:
2092  =082A 909090909090              nop ! nop ! nop ! nop ! nop ! nop  ;00-0f
2093  =0830 909090909090              nop ! nop ! nop ! nop ! nop ! nop
2094  =0836 90909090                  nop ! nop ! nop ! nop
2095  =
2096  =083A 909090909090              nop ! nop ! nop ! nop ! nop ! nop  ;10-1f
2097  =0840 909090909090              nop ! nop ! nop ! nop ! nop ! nop
2098  =0846 90909090                  nop ! nop ! nop ! nop
2099  =
2100  =084A 909090909090              nop ! nop ! nop ! nop ! nop ! nop  ;20-2f
2101  =0850 909090909090              nop ! nop ! nop ! nop ! nop ! nop
2102  =0856 90909090                  nop ! nop ! nop ! nop
2103  =
2104  =085A 909090909090              nop ! nop ! nop ! nop ! nop ! nop  ;30-3f
2105  =0860 909090909090              nop ! nop ! nop ! nop ! nop ! nop
2106  =0866 90909090                  nop ! nop ! nop ! nop
2107  =
2108  =086A 909090909090              nop ! nop ! nop ! nop ! nop ! nop  ;40-4f
2109  =0870 909090909090              nop ! nop ! nop ! nop ! nop ! nop
2110  =0876 90909090                  nop ! nop ! nop ! nop
2111  =
2112  =087A 909090909090              nop ! nop ! nop ! nop ! nop ! nop  ;50-5f
2113  =0880 909090909090              nop ! nop ! nop ! nop ! nop ! nop
2114  =0886 90909090                  nop ! nop ! nop ! nop
2115  =
2116  =088A 909090909090              nop ! nop ! nop ! nop ! nop ! nop  ;60-6f
2117  =0890 909090909090              nop ! nop ! nop ! nop ! nop ! nop
2118  =0896 90909090                  nop ! nop ! nop ! nop
2119  =
2120  =089A 909090909090              nop ! nop ! nop ! nop ! nop ! nop  ;70-7f
2121  =08A0 909090909090              nop ! nop ! nop ! nop ! nop ! nop
2122  =08A6 90909090                  nop ! nop ! nop ! nop
2123  =
2124
```

```
2125
2126  =                       eject 1 include uda.fmt
2127  =                     ;*******************************************************
2128  =                     ;*
2129  =                     ;*      User Data Area - The User Data Area is an
2130  =                     ;*      extension of the process descriptor but it
2131  =                     ;*      travels with the user.  It contains info
2132  =                     ;*      that is needed only while in context.
2133  =                     ;*
2134  =                     ;*      While in the operating system, The Extra
2135  =                     ;*      Segment register points to the beginning
2136  =                     ;*      of the User Data Area.
2137  =                     ;*
2138  =                     ;*******************************************************
2139  =
2140  =  0060              ud_insys        equ     60h
2141  =
2142  =                       eseg
2143  =                       org     0
2144  =
2145  =0000                 u_dparam        rw      1       ; arg to dispatch
2146  =
2147  =                     ;       this area overlays part of BDOS
2148  =0002                 u_dma_ofst      rw      1       ; BDOS dma offset
2149  =0004                 u_dma_seg       rw      1       ; BDOS dma segment
2150  =0006                 u_func          rb      1       ; actual function number
2151  =0007                 u_searchl       rb      1       ; BDOS search length
2152  =0008                 u_searcha       rw      1       ; BDOS search FCB offset
2153  =000A                 u_searchabase   rw      1       ; BDOS search user's segment
2154  =000C                 u_dcnt          rw      1       ; BDOS directory count
2155  =000E                 u_dblk          rw      1       ; BDOS directory block #
2156  =0010                 u_error_mode    rb      1       ; BDOS error mode
2157  =0011                 u_mult_cnt      rb      1       ; BDOS multi-sector count
2158  =0012                 u_df_password   rb      8       ; BDOS default password
2159  =001A                 u_pd_cnt        rb      1       ; BDOS process count
2160  =  0019               uda_ovl_len     equ     (offset $)-(offset u_dma_ofst)
2161  =                     ;       end of overlay area
2162  =
2163  =
2164  =001B                 u_in_int        rb      1
2165  =001C                 u_sp            rw      1       ; save register area
2166  =001E                 u_ss            rw      1
2167  =0020                 u_ax            rw      1
2168  =0022                 u_bx            rw      1
2169  =0024                 u_cx            rw      1
2170  =0026                 u_dx            rw      1
2171  =0028                 u_di            rw      1
2172  =002A                 u_si            rw      1
2173  =002C                 u_bp            rw      1
2174  =002E                 u_wrkseg        rw      1       ; curr seg addr of buf
2175  =0030                 u_retseg        rw      1       ; usr ES return
2176  =0032                 u_ds_sav        rw      1       ;\
2177  =0034                 u_stack_sp      rw      1       ; usr stack segment
```

```
2178
2179  =0036                 u_stack_ss      rw      1       ; usr stack pointer
2180  =0038                 u_ivectors      rw      4       ; save int 0,1
2181  =0040                 u_unused1       rw      2       ;
2182  =0044                 u_ivectors2     rw      4       ; save int 3,4
2183  =004C                 u_es_sav        rw      1       ; > Used during interrupts
2184  =004E                 u_flag_sav      rw      1       ;/
2185  =
2186  =0050                 u_initcs        rw      1
2187  =0052                 u_initds        rw      1
2188  =0054                 u_inites        rw      1
2189  =0056                 u_initss        rw      1
2190  =0058                 u_os_ip         rw      1       ; O.S. vec save
2191  =005A                 u_os_cs         rw      1
2192  =005C                 u_debug_ip      rw      1       ; RTS,Debug Vector Save
2193  =005E                 u_debug_cs      rw      1
2194  =0060                 u_insys         rb      1       ; # times through user_entry
2195  =0061                 u_stat_sav      rb      1       ; used during interrupts
2196  =0062                 u_conccb        rw      1       ; default console's CCB addr
2197  =0064                 u_lstccb        rw      1       ; default list devices CCB addr
2198  =0066                 u_delim         rb      1       ; delimiter for user function 9
2199  =
2200  =                     ;       org     ulen
2201  =                     ;u_8087         rw      47      ; 8087 save area
2202  =                     ;                               ; see dispatcher, loader and terminate
2203
```

```
2204
2205  =                         eject | include sysdat.dat
2206  =                       ;*****************************************************
2207  =                       ;*
2208  =                       ;*      System Data Area
2209  =                       ;*
2210  =                       ;*****************************************************
2211  =
2212  =                               CSEG
2213  =                               org     0ch
2214  =                       ;dev_ver
2215  =000C 06                                db      6       ;development system data version
2216  =                                                       ;SYSDAT.CON has 16 byte code segment
2217  =
2218  =                               DSEG
2219  =                               org     0
2220  =
2221  =                               ;
2222  =                               ;This data is initialized by GENCCPM
2223  =                               ;
2224  =
2225  =                               ;Module Table - contains the FAR CALL addresses
2226  =                               ;       of each module for their initialization
2227  =                               ;       and entry routines.
2228  =                               ;
2229  =                               ;       +---+---+---+---+---+---+---+---+
2230  =                               ;       |      entry      |   initialize  |
2231  =                               ;       +---+---+---+---+---+---+---+---+
2232  =                               ;
2233  =                               ;               entry   init
2234  =                               ;               -----   ----
2235  =
2236  =  0000                 module_table    equ     dword ptr (offset $)
2237  =  0000                 supmod          equ     (offset $)
2238  =0000 030000000000                      dw      3,0,    0,0     ;SUP
2239         0000
2240  =
2241  =  0008                 rtmmod          equ     (offset $)
2242  =0008 030000000000                      dw      3,0,    0,0     ;RTM
2243         0000
2244  =
2245  =  0010                 memmod          equ     (offset $)
2246  =0010 030000000000                      dw      3,0,    0,0     ;MEM
2247         0000
2248  =
2249  =  0018                 ciomod          equ     (offset $)
2250  =0018 030000000000                      dw      3,0,    0,0     ;CIO
2251         0000
2252  =
2253  =  0020                 bdosmod         equ     (offset $)
2254  =0020 030000000000                      dw      3,0,    0,0     ;BDOS
2255         0000
2256  =
```

```
2257
2258  =  0028                  xiosmod         equ     (offset $)
2259  =0028 030C0000000C                       dw      0C03H,0,        0C00H,0 ;XIOS
2260        0000
2261  =
2262  =  0030                  netmod          equ     (offset $)
2263  =0030 030000000000                       dw      3,0,    0,0     ;NET
2264        0000
2265  =
2266  =  0038                  dispatcher      equ     (offset $)
2267  =0038 00000000                           dw      0,0             ;far dispatcher (does IRET)
2268  =
2269  =  003C                  rtm_pdisp       equ     (offset $)
2270  =003C 00000000                           dw      0,0             ;far dispatcher (does RETF)
2271  =
2272  =                                ; location in memory of MP/M-86 or CCP/M-86
2273  =
2274  =0040 0810               osseg           dw      1008h   ;1st parag. of MP/M-86 or CCP/M
2275  =0042 0000               rspseg          dw      0       ;segment of first RSP
2276  =0044 0000               endseg          dw      0       ;1st parag. outside of MP/M or CCP/M
2277  =
2278  =0046 3F                 module_map      db      03fh    ;bit map of modules that exist
2279  =                                                        ; in this system.  low order bit
2280  =                                                        ; corresponds to 1st module in
2281  =                                                        ; module table.  If bit is on,then
2282  =                                                        ; module exists.
2283  =
2284  =                                ; some answers to GENCCPM questions
2285  =
2286  =0047 04                 ncns            db      4       ;# system console devices
2287  =0048 01                 nlst            db      1       ;# system list devices
2288  =0049 05                 nccb            db      5       ;# character control blocks
2289  =004A 20                 nflags          db      20h     ;# flags
2290  =004B 01                 srchdisk        db      1       ;system disk
2291  =004C 0040               mmp             dw      04000h  ;Max Memory per process
2292  =004E 00                 nslaves         db      0       ;Number of network requestors
2293  =004F 00                 dayfile         db      0       ;If Offh,display command info
2294  =0050 01                 tempdisk        db      1       ;Temporary disk
2295  =0051 3C                 tickspersec     db      60      ;Number of ticks per second
2296  =
2297  =                                ; data lists created by GENCCPM
2298  =
2299  =0052 0000               free_root       dw      0       ;locked unused list
2300  =0054 0000               ccb             dw      0       ;addr. Console Ctrl Blk Table
2301  =0056 0000               flags           dw      0       ;addr. Flag Table
2302  =0058 2000               mdul            dw      020h    ;Mem descr. Unused List
2303  =005A 0000               mfl             dw      0       ;Memory Free List
2304  =005C 1400               pul             dw      014h    ;Proc. descr. Unused List
2305  =005E 2000               qul             dw      020h    ;QCB Unused List
2306  =                                                        ;MAU for queue buffer info
2307  =0060 0000               qmau            dw      0       ;link
2308  =0062 0000                               dw      0       ;start segment
2309  =0064 0004                               dw      400h    ;length
```

```
2310
2311  =0066 0000                        dw      0       ;plist
2312  =
2313  =                         ;
2314  =                         ;This data is initialized at Assembly time
2315  =                         ;
2316  =
2317  =0068 7704        rlr             dw      initpd  ;Ready List Root
2318  =006A 0000        dlr             dw      0       ;Delay List Root
2319  =006C 0000        drl             dw      0       ;Dispatcher Ready List
2320  =006E 0000        plr             dw      0       ;Poll List Root
2321  =0070 0000        scl             dw      0       ;Shared Code List
2322  =0072 7704        thrdrt          dw      initpd  ;Process Thread Root
2323  =0074 9401        qlr             dw      mxloadqd;Queue List Root
2324  =0076 0000        mal             dw      0       ;Memory Alloc List
2325  =
2326  =                         ; Version information
2327  =
2328  =0078 0000        version         dw      unknown ;addr. version str in SUP code segment
2329  =                                         ;set by GENCCPM if CCP/M
2330  =                 if mpm
2331  =                 bvernum         dw      01130h  ;MPM-86 w/BDOS v3.0
2332  =                 osvernum        dw      01121h  ;MPM-86 V2.1
2333  =                 endif
2334  =
2335  =                 if ccpm
2336  =007A 3114        bvernum         dw      01431h  ;CCP/M w/BDOS 3.1
2337  =007C 2014        osvernum        dw      01420h  ;CCP/M V2.0
2338  =                 endif
2339  =
2340  =                         ; Time of Day Structure
2341  =
2342  =007E             tod             rw      0
2343  =007E 7E06        tod_day         dw      067EH   ;day since 1/1/78 (09 Jul 82)
2344  =0080 12          tod_hr          db      12h     ;hour of day
2345  =0081 00          tod_min         db      00h     ;minute of hour
2346  =0082 00          tod_sec         db      00h     ;second of minute
2347  =
2348  =                         ; info from XIOS
2349  =
2350  =0083 00          ncondev         db      0               ;# console devs in XIOS
2351  =0084 00          nlstdev         db      0               ;# character devs in XIOS
2352  =0085 00          nciodev         db      0               ;# character i/o devices
2353  =                                                         ; supported by XIOS.
2354  =0086 0000        lcb             dw      0               ; list control block address
2355  =0088 0000        openvec         dw      0               ; open file vector
2356  =008A 20          lock_max        db      20h             ; Max Locked Records/process
2357  =008B 20          open_max        db      20h             ; Max Open Files/process
2358  =008C 0000        owner8087       dw      0               ; no one owns it initially
2359  =008E             rw              1                       ; RESERVED
2360  =0090 FF          cmod            db      0ffh            ; BDOS Compatibility
2361  =0091 00          ndp8087         db      false           ; true 8087 exits
2362  =                                                         ; (Numeric Data Processor)
```

```
2363
2364  =0092 00000000          err_intercept   dw      0,0             ; BDOS does a callf here
2365  =                                                               ; to print error msgs,
2366  =                                                               ; if second word is <> 0
2367  =0096 4105              slr             dw      offset mem_spb  ; Sync List Root
2368  =0098 000000000000              dw      0,0,0,0         ; RESERVED
2369        0000
2370  =
2371  =
2372  =                               ; SYSENT Table - MP/M-86, CCP/M-86 system function information
2373  =                               ;       The supervisor calls the appropriate module
2374  =                               ;       through this table.
2375  =                               ;
2376  =                               ;        Low Byte  High byte
2377  =                               ;       +----+----+--------+
2378  =                               ;       |function |flgs|mod|
2379  =                               ;       +----+----+--------+
2380  =                               ;
2381  =                               ;       flgs    - 001h - network intercept
2382  =                               ;                 if on, the network module is called
2383  =                               ;                 first, on return, either the function
2384  =                               ;                 is called or it is considered complete
2385  =                               ;                 depending on the return.
2386  =                               ;       mod     - module number (0-15)
2387  =                               ;       function- function to call within module
2388  =
2389  =                               ;       note: sup function 0 returns not the
2390  =                               ;                 implemented error code to the caller,
2391  =                               ;                 and sup function 1 returns the illegal
2392  =                               ;                 function error code.
2393  =
2394  =                               ; standard CPM-2 functions
2395  =
2396  =                               ;       func,   module
2397  =
2398  =                               org     ((offset $) + 1) and 0fffeh
2399  =
2400  =00A0 0002              sysent  db      0,      rtm     ;  0-system reset
2401  =00A2 0004                      db      0,      cio     ;  1-conin
2402  =00A4 0104                      db      1,      cio     ;  2-conout
2403  =00A6 0001                      db      0,      sup     ;  3-raw conin/aux in
2404  =00A8 0001                      db      0,      sup     ;  4-raw conout/aux out
2405  =00AA 0404                      db      4,      cio     ;  5-list out
2406  =00AC 0504                      db      5,      cio     ;  6-raw conio
2407  =00AE 0001                      db      0,      sup     ;  7-getiobyte
2408  =00B0 0001                      db      0,      sup     ;  8-setiobyte
2409  =00B2 0604                      db      6,      cio     ;  9-conwrite
2410  =00B4 0704                      db      7,      cio     ; 10-conread
2411  =00B6 0804                      db      8,      cio     ; 11-constat
2412  =00B8 0201                      db      2,      sup     ; 12-get version
2413  =00BA 0005                      db      0,      bdos    ; 13-diskreset
2414  =00BC 0105                      db      1,      bdos    ; 14-diskselect
2415  =00BE 0205                      db      2,      bdos    ; 15-file open
```

```
2416
2417  =00C0 0305                db      3,      bdos    ; 16-file close
2418  =00C2 0405                db      4,      bdos    ; 17-search first
2419  =00C4 0505                db      5,      bdos    ; 18-search next
2420  =00C6 0605                db      6,      bdos    ; 19-file delete
2421  =00C8 0705                db      7,      bdos    ; 20-file read seq
2422  =00CA 0805                db      8,      bdos    ; 21-file write seq
2423  =00CC 0905                db      9,      bdos    ; 22-file make
2424  =00CE 0A05                db      10,     bdos    ; 23-file rename
2425  =00D0 0B05                db      11,     bdos    ; 24-login vector
2426  =00D2 0C05                db      12,     bdos    ; 25-get def disk
2427  =00D4 0D05                db      13,     bdos    ; 26-set dma
2428  =00D6 0E05                db      14,     bdos    ; 27-get alloc vector
2429  =00D8 0F05                db      15,     bdos    ; 28-write protect
2430  =00DA 1005                db      16,     bdos    ; 29-get r/O vector
2431  =00DC 1105                db      17,     bdos    ; 30-set file attr.
2432  =00DE 1205                db      18,     bdos    ; 31-get disk parm block
2433  =00E0 1305                db      19,     bdos    ; 32-user code
2434  =00E2 1405                db      20,     bdos    ; 33-file read random
2435  =00E4 1505                db      21,     bdos    ; 34-file write random
2436  =00E6 1605                db      22,     bdos    ; 35-file size
2437  =00E8 1705                db      23,     bdos    ; 36-set random record
2438  =00EA 1805                db      24,     bdos    ; 37-reset drive
2439  =00EC 1905                db      25,     bdos    ; 38-access drive
2440  =00EE 1A05                db      26,     bdos    ; 39-free drive
2441  =00F0 1B05                db      27,     bdos    ; 40-file write random w/zero fill
2442  =
2443  =                                         ;CPM-3 extensions
2444  =
2445  =00F2 0001                db      0,      sup     ; 41-Test and Write (NOT IMPLEMENTED)
2446  =                                         ; Would be BDOS func # 28
2447  =00F4 1C05                db      28,     bdos    ; 42-Lock Record
2448  =00F6 1D05                db      29,     bdos    ; 43-Unlock Record
2449  =00F8 1E05                db      30,     bdos    ; 44-Set Multi-sector
2450  =00FA 1F05                db      31,     bdos    ; 45-Set Bdos Error Mode
2451  =00FC 2005                db      32,     bdos    ; 46-Get Disk Free Space
2452  =00FE 0C01                db      12,     sup     ; 47-Chain to Program
2453  =                                         ; In CP/M-86 BDOS func # 34
2454  =0100 2105                db      33,     bdos    ; 48-Flush Buffers
2455  =0102 0101                db      1,      sup     ; 49-
2456  =
2457  =                                         ;CPM-86 extensions
2458  =
2459  =0104 0301                db      3,      sup     ; 50-call xios
2460  =0106 2205                db      34,     bdos    ; 51-set dma base
2461  =0108 2305                db      35,     bdos    ; 52-get dma
2462  =010A 0003                db      0,      mem     ; 53-get max mem
2463  =010C 0103                db      1,      mem     ; 54-get abs max mem
2464  =010E 0203                db      2,      mem     ; 55-alloc mem
2465  =0110 0303                db      3,      mem     ; 56-alloc abs mem
2466  =0112 0403                db      4,      mem     ; 57-free mem
2467  =0114 0503                db      5,      mem     ; 58-free all mem
2468  =0116 0401                db      4,      sup     ; 59-load
```

```
2469
2470  =0118 0101                db      1,      sup     ; 60-
2471  =011A 0101                db      1,      sup     ; 61-
2472  =011C 0101                db      1,      sup     ; 62-
2473  =011E 0101                db      1,      sup     ; 63-
2474  =
2475  =                                         ;CP/NET functions
2476  =
2477  =0120 4007                db      64,     net     ; 64-network login
2478  =0122 4107                db      65,     net     ; 65-network logoff
2479  =0124 4207                db      66,     net     ; 66-network send msg
2480  =0126 4307                db      67,     net     ; 67-network rcv msg
2481  =0128 4407                db      68,     net     ; 68-network status
2482  =012A 4507                db      69,     net     ; 69-get network config addr
2483  =
2484  =                                         ;CP/M-3 extensions
2485  =
2486  =012C 2405                db      36,     bdos    ; 98-Reset Alloc Vector
2487  =012E 2505                db      37,     bdos    ; 99-Truncate File
2488  =0130 2605                db      38,     bdos    ;100-Set Dir Label
2489  =0132 2705                db      39,     bdos    ;101-Return Dir Label
2490  =0134 2805                db      40,     bdos    ;102-Read File XFCB
2491  =0136 2905                db      41,     bdos    ;103-Write File XFCB
2492  =0138 2A05                db      42,     bdos    ;104-Set Date and Time
2493  =013A 2B05                db      43,     bdos    ;105-Get Date and Time
2494  =013C 2C05                db      44,     bdos    ;106-Set Default Password
2495  =013E 0D01                db      13,     sup     ;107-Return Serial Number
2496  =0140 0001                db      0,      sup     ;108-(not implemented)
2497  =0142 1904                db      25,     cio     ;109-Get/Set Console Mode
2498  =0144 1A04                db      26,     cio     ;110-Get/Set Output Delimiter
2499  =0146 1B04                db      27,     cio     ;111-Print Block
2500  =0148 1C04                db      28,     cio     ;112-List Block
2501  =
2502  =                                         ; MP/M functions
2503  =
2504  =014A 0603                db      6,      mem     ;128-mem req
2505  =014C 0603                db      6,      mem     ;129-(same function as 128)
2506  =014E 0703                db      7,      mem     ;130-mem free
2507  =0150 0102                db      1,      rtm     ;131-poll device
2508  =0152 0202                db      2,      rtm     ;132-flag wait
2509  =0154 0302                db      3,      rtm     ;133-flag set
2510  =0156 0402                db      4,      rtm     ;134-queue make
2511  =0158 0502                db      5,      rtm     ;135-queue open
2512  =015A 0602                db      6,      rtm     ;136-queue delete
2513  =015C 0702                db      7,      rtm     ;137-queue read
2514  =015E 0802                db      8,      rtm     ;138-cond. queue read
2515  =0160 0902                db      9,      rtm     ;139-queue write
2516  =0162 0A02                db      10,     rtm     ;140-cond. queue write
2517  =0164 0B02                db      11,     rtm     ;141-delay
2518  =0166 0C02                db      12,     rtm     ;142-dispatch
2519  =0168 0D02                db      13,     rtm     ;143-terminate
2520  =016A 0E02                db      14,     rtm     ;144-create process
2521  =016C 0F02                db      15,     rtm     ;145-set priority
```

```
2522
2523  =016E 0904                    db      9,      cio     ;146-console attach
2524  =0170 0A04                    db      10,     cio     ;147-console detach
2525  =0172 0B04                    db      11,     cio     ;148-set def console
2526  =0174 0C04                    db      12,     cio     ;149-console assign
2527  =0176 0501                    db      5,      sup     ;150-CLI
2528  =0178 0601                    db      6,      sup     ;151-call RPL
2529  =017A 0701                    db      7,      sup     ;152-parse filename
2530  =017C 0D04                    db      13,     cio     ;153-get def console
2531  =017E 0801                    db      8,      sup     ;154-sysdat addr
2532  =0180 0901                    db      9,      sup     ;155-time of day
2533  =0182 1002                    db      16,     rtm     ;156-get PD addr
2534  =0184 1102                    db      17,     rtm     ;157-abort process
2535  =
2536  =                             ; MPM II extensions
2537  =
2538  =0186 0F04                    db      15,     cio     ;158-attach list
2539  =0188 1004                    db      16,     cio     ;159-detach list
2540  =018A 1104                    db      17,     cio     ;160-set list dev
2541  =018C 1204                    db      18,     cio     ;161-Cond. Attach list
2542  =018E 1304                    db      19,     cio     ;162-Cond. Attach Console
2543  =0190 0B01                    db      11,     sup     ;163-MP/M Version Number
2544  =0192 1404                    db      20,     cio     ;164-get list dev
2545  =
2546  =
2547  =                             ; Initialized Queues
2548  =
2549  =                             org     ((offset $) + 1) and 0fffeh
2550  =
2551  =0194 7408            mxloadqd        dw      mxdiskqd
2552  =0196 0000                            db      0,0
2553  =0198 0300                            dw      qf_keep+qf_mx
2554  =019A 4D584C6F6164                    db      'MXLoad  '
2555        2020
2556  =01A2 000001000000                    dw      0,1,0,0,1,0,0
2557        000001000000
2558        0000
2559  =01B0 0000            mxloadqpb       db      0,0
2560  =01B2 940101000000                    dw      mxloadqd,1,0
2561  =                     ;               db      'MXLoad  '
2562  =
2563  =                             ; Data Used by Load Program
2564  =
2565  =                             org     ((offset $) + 1) and 0fffeh
2566  =
2567  =01B8 0000            lod_uda         dw      0
2568  =01BA 0000            lod_lstk        dw      0
2569  =01BC 0000            lod_basep       dw      0
2570  =01BE 0000            lod_nldt        dw      0
2571  =01C0 0000            lod_pd          dw      0
2572  =01C2                 lod_fcb         rs      36
2573  =01E6 0000            lod_indma       dw      0
2574  =01E8 00              lod_nrels       db      0
```

```
2575
2576  =01E9 00              lod_chain       db      0
2577  =01EA 00              lod_user        db      0
2578  =01EB 00              lod_disk        db      0
2579  =01EC 00              lod_fifty       db      0
2580  =01ED 00              lod_8080        db      0
2581  =01EE 00              lod_lbyte       db      0
2582  =01EF 0000            lod_fixrec      dw      0
2583  =01F1 0000            lod_fixrecl     dw      0
2584  =01F3                 lod_dma         rb      dskrecl
2585  =0273                 ldtab           rb      ldtabsiz
2586  =
2587  =031D                 cli_dma_ofst    rw      1
2588  =031F                 cli_dma_seg     rw      1
2589  =0321                 cli_pflag       rw      1
2590  =0323                 cli_chain       rb      1
2591  =0324                 cli_term        rb      1
2592  =0325                 cli_dma         rb      dskrecl         ;dma buffer
2593  =
2594  =                                     ;copy of user's clicb
2595  =03A5                 cli_net         rb      1               ;net
2596  =03A6                 cli_ppd         rw      1               ;parent PD
2597  =03A8                 cli_cmdtail     rb      129             ;command sent
2598  =0429                                 rb      1
2599  =
2600  =042A                 cli_fcb         rb      fcblen+1        ;internal FCB
2601  =
2602  =0448 0000            cli_cuspqpb     db      0,0             ;QPB of command
2603  =044D 00000000                        dw      0,0
2604  =0451 A603                            dw      offset cli_ppd
2605  =0453 242424242424                    db      '$$$$$$$$'
2606        2424
2607  =
2608  =0458 0000            cli_acb         db      0,0             ;cns,match
2609  =045D 0000                            dw      0               ;pd
2610  =045F 242424242424                    db      '$$$$$$$$'      ;name
2611        2424
2612  =
2613  =0467 A803            cli_pcb         dw      offset cli_cmdtail      ;parse
2614  =0469 2A04                            dw      offset cli_fcb          ;ctl bk
2615  =
2616  =046B 0000            cli_pd          dw      0       ;pd of load prog
2617  =046D 0000            cli_err         dw      0       ;error return
2618  =046F 0000            cli_bpage       dw      0       ;base page
2619  =0471 01              cli_lddsk       db      1       ;load disk
2620  =
2621  =                             ;parent information
2622  =
2623  =0472 00              cli_cns         db      0       ;pd.p_cns save
2624  =0473 00              cli_user        db      0       ;pd.p_dsk save
2625  =0474 00              cli_dsk         db      0       ;pd.p_user save
2626  =0475 00              cli_err_mode    db      0       ;u_error_mode save
2627  =0476 00              cli_dfil        db      0       ;dayfile flag
```

```
2628
2629  =
2630  =                                ;
2631  =                                ;System Initialization Variables
2632  =                                ;
2633  =
2634  =0477 0000                initpd          dw      0               ;link
2635  =0479 0000                                dw      0               ;thread
2636  =047B 00                                  db      ps_run          ;stat
2637  =047C 01                                  db      1               ;prior
2638  =047D 0500                                dw      pf_sys+pf_kernal;flag
2639  =047F 496E69742020                        db      'Init  '        ;name
2640        2020
2641  =0487 0000                                dw      unknown         ;uda segment
2642  =0489 00                                  db      0               ;disk
2643  =048A 00                                  db      0               ;user
2644  =048B 0000                                db      0,0             ;ldsk,luser
2645  =048D 0000                                dw      0               ;mem
2646  =048F 0000                                dw      0               ;dvract
2647  =0491 0000                                dw      0               ;wait
2648  =0493 0000                                db      0,0             ;org,net
2649  =0495 0000                                dw      0               ;parent
2650  =0497 00                                  db      0               ;cns
2651  =0498 00                                  db      0               ;abort
2652  =0499 0000                                db      0,0             ;cin,cout
2653  =049B 00                                  db      0               ;lst
2654  =049C 000000                              db      0,0,0           ;sf3,4,5
2655  =049F                                     rb      4               ;reserved
2656  =04A3 00000000                            dw      0,0             ;pret,scratch
2657  =
2658  =
2659  =                                ;User Data Area of Init process
2660  =                                ;paragraph aligned
2661  =
2662  =                                org     ((offset $)+0fh) AND 0fff0h
2663  =04B0                     inituda         rb      ulen
2664  =05B0                     init_tos        rw      0
2665  =                                org     offset inituda + ud_insys
2666  =0510 01                                  db      1               ;keep the SUP from doing stack
2667  =                                org     offset init_tos         ;switches
2668  =
2669  =
2670  =                                ; RTM data
2671  =                                ; is word aligned from init uda
2672  =
2673  =05B0 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2674  =05B6 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2675  =05BC CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2676  =
2677  =05C2 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2678  =05C8 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2679  =05CE CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2680  =05D4 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
```

```
2681
2682  =
2683  =050A CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2684  =05E0 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2685  =05E6 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2686  =05EC CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2687  =
2688  =05F2 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2689  =05F8 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2690  =05FE CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2691  =0604 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2692  =
2693  =060A CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2694  =0610 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2695  =0616 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2696  =061C CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2697  =
2698  =0622                 dsptchtos       rw      0
2699  =
2700  =0622 00              indisp          db      false   ;?currently in dispatch?
2701  =0623 00              intflag         db      0       ;if 0, interrupts not enabled -
2702  =                                             ;not implemented
2703  =0624 0000            es_sav          dw      0       ;(staying word aligned)
2704  =0626 0000            bx_sav          dw      0
2705  =0628 0000            ax_sav          dw      0
2706  =
2707  =                             ; MEM Data
2708  =
2709  =062A 0000            beststart       dw      0
2710  =062C 0000            bestlen         dw      0
2711  =062E 0000            bestsi          dw      0
2712  =0630 0000            bestmau         dw      0
2713  =0632 0000            currmau         dw      0
2714  =0634 0000            currsi          dw      0
2715  =0636 000000000000    currmpb         dw      0,0,0,0,0
2716        00000000
2717  =
2718  =
2719  =                     ;       SYNC Parameter Blocks
2720  =
2721  =                     ;       The MEM ENTRY: point uses the following for
2722  =                     ;       mutual exclusion and recursion.
2723  =
2724  =0640 00              mem_cnt         db      0       ;how many times a process has recursivly
2725  =                                             ;called the memory manager
2726  =
2727  =0641 4906            mem_spb         dw      q_spb   ;link   Mem Sync Parameter Block
2728  =0643 0000                            dw      0       ;owner
2729  =0645 0000                            dw      0       ;wait
2730  =0647 0000                            dw      0       ;next
2731  =
2732  =                     ;       The queue sub-system in the RTM uses the following
2733  =                     ;       structure for mutual exclusion
```

```
2734
2735  =
2736  =0649 5106            q_spb           dw      cli_spb ;link   Queue Sync Parameter Block
2737  =064B 0000                            dw      0       ;owner
2738  =064D 0000                            dw      0       ;wait
2739  =064F 0000                            dw      0       ;next
2740  =
2741  =
2742  =                     ;       The CLI uses the CLI_SPB for mutual exclusion
2743  =
2744  =0651 5906            cli_spb         dw      thrd_spb;link   CLI Sync Parameter Block
2745  =0653 0000                            dw      0       ;owner
2746  =0655 0000                            dw      0       ;wait
2747  =0657 0000                            dw      0       ;next
2748  =
2749  =                     ;       When the thread is accessed, the THRD_SPB must be owned
2750  =                     ;       first.
2751  =
2752  =0659 A008            thrd_spb        dw      msg_spb ;link   Thread Sync Parameter Block
2753  =065B 0000                            dw      0       ;owner
2754  =065D 0000                            dw      0       ;wait
2755  =065F 0000                            dw      0       ;next
2756  =
2757  =                     ; Currently the order in which the SYNCs must be obtained if
2758  =                     ; more than one is needed is:
2759  =
2760  =                     ;       CLI
2761  =                     ;       QUEUE                   ;called by CLI for RSPs
2762  =                     ;       MEM                     ;called by make queue
2763  =                     ;       THREAD                  ;used from the MEM module
2764  =
2765  =                     ; The SYNCs must be released in reverse order
2766  =                     ; MSG_SPB is used by the BDOS to protect the BDOS error message
2767  =                     ; buffer.  MSG_SPB is in DATA.BDO
2768
```

```
2769
2770  =                         eject 1 include data.bdo
2771  =
2772  =                       ;*****************************************************
2773  =                       ;*
2774  =                       ;*      bdos data area
2775  =                       ;*
2776  =                       ;*****************************************************
2777  =
2778  =  FFFF                 CCPMOFF equ     true
2779  =
2780  =                       if BCPM
2781  =
2782  =                       ;
2783  =                       ;       8086 variables that must reside in code segment
2784  =                       ;
2785  =                       cseg    $
2786  =                       ;
2787  =                       axsave          dw      0       ; register saves
2788  =                       ss_save         dw      0
2789  =                       sp_save         dw      0
2790  =                       stack_begin     dw      endstack
2791  =                       ;
2792  =                       ;       variables in data segment:
2793  =                       ;
2794  =                       dseg    cpmsegment
2795  =                               org     bdosoffset+bdoscodesize
2796  =
2797  =                       header  rs      128
2798  =                               rs      72
2799  =
2800  =                       pag0    dw      0               ;address of user's page zero
2801  =                       ip0     db      0               ;initial page value for ip register
2802  =                       ;
2803  =                       ;     memory control block
2804  =                       ;
2805  =                       umembase        dw      0       ;user'sbase for memory request
2806  =                       umemlg  dw      0               ;length of memory req
2807  =                       contf   db      0               ;flag indicates added memory is avail
2808  =                       ;
2809  =                       ;
2810  =                       hold_info       dw      0       ;save info
2811  =                       hold_spsave     dw      0       ;save user sp during program load
2812  =                       hold_sssave     dw      0       ;save user ss during program load
2813  =
2814  =                       mod8080         db      0
2815  =                       ;
2816  =                       ;       byte i/o variables:
2817  =                       ;
2818  =                       compcol db      0               ;true if computing column position
2819  =                       strtcol db      0               ;starting column position after read
2820  =                       column  db      0               ;column position
2821  =                       listcp  db      0               ;listing toggle
```

```
2822
2823  =                     kbchar  db      0               ;initial key char = 00
2824  =
2825  =                     endif
2826  =
2827  =                             dseg
2828  =                     if BMPM
2829  =                             org     0c00h   ;org for MP/M system
2830  =                     endif
2831  =
2832  =                     if BMPM and CCPMOFF
2833  =                             org     0800h   ;org for CCP/M system
2834  =                     endif
2835  =
2836  =                     if PMPM
2837  =0800 0000            rlog            dw      0       ;removeable logged-in disks
2838  =0802 0000            tlog            dw      0       ;removeable disk ,test login vector
2839  =0804 0000            ntlog           dw      0       ;new tlog vector
2840  =
2841  =                     endif
2842  =
2843  =0806 0000            rodsk           dw      0       ;read only disk vector
2844  =0808 0000            dlog            dw      0       ;logged-in disks
2845  =
2846  =080A 00              open_fnd        db      0               ;open file found in srch_olist
2847  =
2848  =                     ;the following variables are set to zero upon entry to file system
2849  =
2850  =080B 00              fcbdsk          db      0               ;disk named in fcb
2851  =080C 00              parlg           db      0               ;length of parameter block copied
2852  =080D 0000            aret            dw      0               ;adr value to return
2853  =  080D               lret            equ     byte ptr aret   ;low(aret)
2854  =080F 00              resel           db      0               ;reselection flag
2855  =0810 00              rd_dir_flag     db      0               ;must read/locate directory record
2856  =0811 00              comp_fcb_cks    db      0               ;compute fcb checksum flag
2857  =0812 00              search_user0    db      0               ;search user 0 for file (open)
2858  =0813 00              make_xfcb       db      0               ;make & search xfcb flag
2859  =0814 00              find_xfcb       db      0               ;search find xfcb flag
2860  =0815 0000            xdcnt           dw      0               ;empty directory dcnt
2861  =0817 00              DIR_CNT         DB      0               ;DIRECT I/O COUNT
2862  =0818 00              MULT_NUM        DB      0               ;MULTI-SECTOR COUNT used in bdos
2863  =0819 00              olap_type       db      0               ;record lock overlap type
2864  =081A 00              ff_flag         db      0               ;0ffh xios error return flag
2865  =081B 00              err_type        db      0               ;type of error to print if not zero
2866  =081C                 zerolength      rb      4               ;reserved bytes
2867  =  0015               zerolength      equ     (offset $)-(offset fcbdsk)
2868  =0820 01              mult_sec        db      1               ;multi sector count passed to xios
2869  =0821 00              RELOG           DB      0               ;AUTOMATIC RELOG SWITCH
2870  =
2871  =0822 00              BLK_OFF         DB      0               ;RECORD OFFSET WITHIN BLOCK
2872  =0823 0000            blk_num         dw      0               ;block number
2873  =                     ;
2874  =0825 2708            ua_lroot        dw      offset ua0      ;unallocated block list root
```

```
2875
2876  =                       ;
2877  =0827 20080000          ua0     dw      offset ua1,0
2878  =082B FF00                      db      0ffh,0
2879  =082D 33080000          ua1     dw      offset ua2,0
2880  =0831 FF00                      db      0ffh,0
2881  =0833 39080000          ua2     dw      offset ua3,0
2882  =0837 FF00                      db      0ffh,0
2883  =0839 3F080000          ua3     dw      offset ua4,0
2884  =083D FF00                      db      0ffh,0
2885  =083F 45080000          ua4     dw      offset ua5,0
2886  =0843 FF00                      db      0ffh,0
2887  =0845 00000000          ua5     dw      0,0
2888  =0849 FF00                      db      0ffh,0
2889  =
2890  =084B                   HASH            RB      4               ;HASH CODE WORK AREA
2891  =084F 00                HASHL           DB      0               ;HASH SEARCH LENGTH
2892  =
2893  =0850 0B                LOG_FXS         db      11              ;number of log_fxs entrys
2894  =                                       ;       fxi15,fxi17,fxi19,fxi22,fxi23
2895  =0851 020406090A                        db      02h  ,04h  ,06h  ,09h  ,0ah
2896  =                                       ;       fxi30,fxi35,fxi99,fxi100,fxi102,fxi103
2897  =0856 111625262829                      db      11h  ,16h  ,25h  ,26h   ,28h   ,29h
2898  =085C 00000000                          db      0,0,0,0         ;reserved
2899  =0860 07                rw_fxs          db      7               ;number of rw_fxs entrys
2900  =                                       ;       fxi20,fxi21,fxi33,fxi34,fxi40,fxi42,fxi43
2901  =0861 070814151B1C                      db      07h  ,08h  ,14h  ,15h  ,1bh  ,1ch  ,1dh
2902          1D
2903  =0868 0000                              db      0,0             ;reserved
2904  =086A 02                sc_fxs          db      2               ;number of sc_fxs entrys
2905  =                                       ;       fxi16,fxi18
2906  =086B 0305                              db      03h  ,05h
2907  =086D 0000                              db      0,0             ;reserved
2908  =
2909  =086F 00                DEBLOCK_FX      DB      0               ;DEBLOCK FUNCTION #
2910  =0870 00                PHY_OFF         DB      0               ;RECORD OFFSET WITHIN PHYSICAL RECORD
2911  =0871 0000              CUR_BCBA        DW      0               ;CURRENT BCB OFFSET
2912  =0873 0000              ROOT_BCBA       DW      0               ;ROOT BCB OFFSET
2913  =0875 0000              EMPTY_BCBA      DW      0               ;EMPTY BCB OFFSET
2914  =
2915  =                       if BMPM
2916  =
2917  =0877 0000              P_LAST_BCBA     DW      0               ;PROCESS'S LAST BCB OFFSET
2918  =0879 00                P_BCB_CNT       DB      0               ;PROCESS BCB COUNT IN BCB LIST
2919  =
2920  =                       endif
2921  =
2922  =087A 00                fx_intrn        db      0               ;internal BDOS function number
2923  =
2924  =087B 0000              TRACK           DW      0       ;BCB RECORD'S TRACK
2925  =087D 0000              SECTOR          DW      0       ;BCB RECORD'S SECTOR
2926  =
2927  =                       ;       seldsk,usrcode are initialized as a pair
```

```
2928
2929  =087F 00              seldsk          db      0       ;selected disk num
2930  =0880 00              usrcode         db      0       ;curr user num
2931  =
2932  =0881 0000            info            dw      0       ;info adr
2933  =0883 0000            searcha         dw      0       ;search adr
2934  =
2935  =                     ;the following variable order is critical
2936  =
2937  =                     ;variables copied from uda for mp/m                     x
2938  =
2939  =                     ;variables included in fcb checksum for mp/m and cp/m     x
2940  =
2941  =                     ;variables used to access system lock list for mp/m         x
2942  =
2943  =0885 0000            dma_ofst        dw      0       ;dma offset             1
2944  =0887 0000            dma_seg         dw      0       ;dma segment            2
2945  =0889 00              func            db      0       ;bdos function #        3
2946  =088A 00              searchl         db      0       ;search len             4
2947  =
2948  =                     if BMPM
2949  =
2950  =088B 0000            searchaofst     dw      0       ;search adr ofst        5
2951  =088D 0000            searchabase     dw      0       ;search adr base        6
2952  =
2953  =                     endif
2954  =
2955  =088F 0000            dcnt            dw      0       ;directory counter      7
2956  =0891 0000            dblk            dw      0       ;directory block        8 ?? - not used - ??
2957  =0893 00              error_mode      db      0       ;bdos error mode        9
2958  =0894 00              mult_cnt        db      0       ;bdos multi-sector cnt 10
2959  =0895                 df_password     rb      8       ;process default pw    11
2960  =
2961  =                     if BMPM
2962  =
2963  =089D 00              pd_cnt          db      0       ;bdos process cnt      12 1
2964  =
2965  =                     endif
2966  =
2967  =089E 00              high_ext        db      0       ;fcb high extent bits     2
2968  =089F 00              xfcb_read_only  db      0       ;xfcb read only flag      3
2969  =08A0 FF              curdsk          db      0ffh    ;current disk             4 1
2970  =
2971  =                     if BMPM
2972  =
2973  =08A1 00              packed_dcnt     db      0       ;packed dblk+dcnt           2
2974  =08A2 00                              db      0
2975  =08A3 00                              db      0
2976  =08A4 0000            pdaddr          dw      0       ;process descriptor addr    3
2977  =
2978  =                     endif
2979  =
2980  =                             org ((offset $) + 1) and 0fffeh
```

```
2981
2982  =
2983  =                         ;       curtrka - alloca are set upon disk select
2984  =                         ;       (data must be adjacent)
2985  =
2986  =08A6 0000               cdrmaxa         dw      0       ;ptr to cur dir max val
2987  =08A8 0000               drvlbla         dw      0       ;drive label data byte addr
2988  =08AA 0000               buffa           dw      0       ;ptr to dir dma addr
2989  =08AC 0000               dpbaddr         dw      0       ;curr disk param block addr
2990  =08AE 0000               checka          dw      0       ;curr checksum vector addr
2991  =08B0 0000               alloca          dw      0       ;curr alloc vector addr
2992  =08B2 0000               DIR_BCBA        DW      0       ;DIRECTORY BUFFER CONTROL BLOCK ADDR
2993  =08B4 0000               DAT_BCBA        dw      0       ;DATA BUFFER CONTROL BLOCK ADDR
2994  =08B6 0000               HASH_SEG        dw      0       ;HASH TABLE SEGMENT .
2995  =  000C                  addlist         equ     12      ;"$-buffa" = addr list size
2996  =
2997  =                         ;       sectpt - offset obtained from disk parm block at dpbaddr
2998  =                         ;       (data must be adjacent)
2999  =
3000  =08B8 0000               sectpt          dw      0       ;sectors per track
3001  =08BA 00                 blkshf          db      0       ;block shift factor
3002  =08BB 00                 blkmsk          db      0       ;block mask
3003  =08BC 00                 extmsk          db      0       ;extent mask
3004  =08BD 0000               maxall          dw      0       ;max alloc num
3005  =08BF 0000               dirmax          dw      0       ;max dir num
3006  =08C1 0000               dirblk          dw      0       ;reserved alloc bits for dir
3007  =08C3 0000               chksiz          dw      0       ;size of checksum vector
3008  =08C5 0000               offsetv         dw      0       ;offset tracks at beginning
3009  =08C7 00                 PHYSHF          DB      0       ;PHYSICAL RECORD SHIFT FACTOR
3010  =08C8 00                 PHYMSK          DB      0       ;PHYSICAL RECORD MASK
3011  =08C9                    endlist         rs      0       ;end of list
3012  =  0011                  dpblist         equ     (offset endlist)-(offset sectpt)
3013  =                                                       ;size
3014  =
3015  =                         ;       local variables
3016  =
3017  =08C9                    common_dma      rb      16      ;copy of user's dma 1st 16 bytes
3018  =08D9 00                 make_flag       db      0       ;make function flag
3019  =08DA 00                 actual_rc       db      0       ;directory ext record count
3020  =08DB 00                 save_xfcb       db      0       ;search xfcb save flag
3021  =08DC 00                 save_mod        db      0       ;open_reel module save field
3022  =08DD 00                 pw_mode         db      0       ;password mode
3023  =08DE 00                 attributes      db      0       ;fcb interface attributes hold byte
3024  =
3025  =                         if BMPM
3026  =
3027  =                         ;       number of lock list items required for lock operation
3028  =08DF 010002020101       required_table  db      1,0,2,2,1,1,2,2,1,1,2,2,1,1,2,2
3029          020201010202
3030          01010202
3031  =
3032  =08EF 00                 chk_olist_flag  db      0       ;check | test olist flag
3033  =08F0 0000               lock_sp         dw      0       ;lock stack ptr
```

```
3034
3035  =08F2 00              lock_shell      db      0       ;lock shell flag
3036  =08F3 00              check_fcb_ret   db      0       ;check_fcb return switch
3037  =08F4 00              lock_unlock     db      0       ;lock | unlock function flag
3038  =08F5 00              incr_pdcnt      db      0       ;increment process_cnt flag ??
3039  =08F6 00              free_mode       db      0       ;free lock list entries flag ??
3040  =                                                     ;1=free entries for curdsk
3041  =                                                     ;0=free all entries
3042  =08F7 0000            cur_pos         dw      0       ;current position in lock list
3043  =08F9 0000            prv_pos         dw      0       ;previous position in lock list
3044  =
3045  =08FB 00              dont_close      db      0       ;inhibit actual close flag
3046  =08FC 00              open_cnt        db      0       ;process open file count
3047  =08FD 00              lock_cnt        db      0       ;process locked record count
3048  =08FE 0000            file_id         dw      0       ;address of file' lock list entry
3049  =0900 00              set_ro_flag     db      0       ;set drive r/o flag
3050  =0901 00              check_disk      db      0       ;disk reset open file check flag
3051  =0902 00              flushed         db      0       ;lock list open file flush flag
3052  =
3053  =                     ;free_root, lock_max, open_max initialized by sysgen
3054  =
3055  =0903 5200                            dw      offset free_root
3056  =0905 0000            open_root       dw      0       ;lock list open file list root
3057  =0907 0000            lock_root       dw      0       ;lock list locked record list root
3058  =
3059  =                     endif
3060  =
3061  =0909 0000            sdcnt           dw      0       ;saved dcnt of file's 1st fcb
3062  =090B 0000            sdcnt0          dw      0       ;saved dcnt (user 0 pass)
3063  =
3064  =                     if BCPM
3065  =
3066  =                     chain_flag      db      0       ;chain flag ??
3067  =                     tod             db      0ffh,0ffh,0ffh,0ffh ??
3068  =
3069  =                     endif
3070  =
3071  =
3072  =090D 00              rmf             db      0       ;read mode flag for open$reel
3073  =090E 00              wflag           db      0       ;xios/bios write flag
3074  =090F 00              dirloc          db      0       ;directory flag in rename, etc.
3075  =0910 00              linfo           db      0       ;low(info)
3076  =0911 00              dminx           db      0       ;local for diskwrite
3077  =0912 00              single          db      0       ;set true if single byte
3078  =                                                     ;alloc map
3079  =0913 00              rcount          db      0       ;record count in curr fcb
3080  =0914 00              extval          db      0       ;extent num and extmsk
3081  =0915 00              vrecord         db      0       ;curr virtual record
3082  =0916 00              ADRTVE          DB      0       ;CURRENT DISK - must preceed arecord
3083  =0917 0000            arecord         dw      0       ;curr actual record
3084  =0919 00                              db      0       ;curr actual record high byte
3085  =091A 0000            ARECORD1        dw      0       ;curr actual block# * blkmsk
3086  =091C 0000            drec            dw      0       ;curr actual directory record
```

```
3087
3088  =091E 0000            CUR_dma_seg     DW      0
3089  =0920 0000            CUR_DMA         DW      0
3090  =
3091  =                     ;       local variables for directory access
3092  =
3093  =0922 00              dptr            db      0       ;directory pointer 0,1,2,3
3094  =  088F               ldcnt           equ     byte ptr dcnt   ;low(dcnt)
3095  =0923 00              user_zero_pass  db      0       ;search user zero flag
3096  =
3097  =                     ;       shell variables
3098  =
3099  =0924 0000            shell_si        dw      0       ;bdos command offset
3100  =0926 000000          shell_rr        db      0,0,0   ;r0,r1,r2 save area
3101  =
3102  =                     ;       special 8086 variables:
3103  =
3104  =0929 0000            uda_save        dw      0       ;user data area saved value
3105  =092B 0000            parametersegment dw     0       ;user parameter segment
3106  =092D 0000            returnseg       dw      0       ;user return segment
3107  =
3108  =                     ;       error messages
3109  =
3110  =092F 00              err_drv         db      0
3111  =0930 0000            err_pd_addr     dw      0
3112  =
3113  =0932 0D0A43502F4D    dskmsg          db      13,10,"CP/M Error On "
3114        204572726F72
3115        204F6F20
3116  =0942 203A2000        dskerr          db      " : ",0
3117  =
3118  =0946 000060096909    xerr_list       dw      0,permsg,rodmsg,rofmsg,selmsg
3119        78098709
3120  =0950 B309C709DC09              dw      xe5,xe6,xe7,xe8,xe9,xe10,xe11,xe3
3121        EB09FF09100A
3122        290A9509
3123  =
3124  =0960 446973682049    permsg          db      "Disk I/O",0
3125        2F4F00
3126  =0969 526561642F4F    rodmsg          db      "Read/Only Disk",0
3127        6E6C79204469
3128        736B00
3129  =0978 526561642F4F    rofmsg          db      "Read/Only File",0
3130        6E6C79204669
3131        6C6500
3132  =0987 496E76616C69    selmsg          db      "Invalid Drive",0
3133        642044726976
3134        6500
3135  =
3136  =0995 46696C65204F    xe3             db      "File Opened in Read/Only Mode",0
3137        70656E656420
3138        696E20526561
3139        642F4F6E6C79
```

```
3140
3141            20406F646500
3142    =
3143    =09B3   46696C652043    xe5             db      'File Currently Open',0
3144            757272656E74
3145            6C79204F7065
3146            6E00
3147    =09C7   436C6F736520    xe6             db      'Close Checksum Error',0
3148            43686563756D
3149            73756D204572
3150            726F7200
3151    =09DC   506173737776F   xe7             db      'Password Error',0
3152            726420457272
3153            6F7200
3154    =09EB   46696C652041    xe8             db      'File Already Exists',0
3155            6C7265616479
3156            204578697374
3157            7300
3158    =09FF   496C6C656761    xe9             db      'Illegal ? in FCB',0
3159            6C203F20696E
3160            20466434200
3161    =0A10   4F70656E2046    xe10            db      'Open File Limit Exceeded',0
3162            696C65204C69
3163            6D6974204578
3164            63656564656D
3165            00
3166    =0A29   4E6F20526F6F    xe11            db      'No Room in System Lock List',0
3167            6D20696E2053
3168            797374656D20
3169            4C6F636B204C
3170            69737400
3171    =
3172    =0A45   0D0A00          crlf_str        db      13,10,0
3173    =0A48   0D0A42646F73    pr_fx           db      13,10,'Bdos Function = '
3174            2046756E6374
3175            696F6E203D20
3176    =0A5A   202020          pr_fx1          db      '   '
3177    =0A5D   2046696C6520    pr_fcb          db      ' File = '
3178            3D20
3179    =0A65                   pr_fcb1         rs      12
3180    =0A71   00                              db      0
3181    =
3182    =0A72   0D0A44697368    deniedmsg       db      13,10,'Disk reset denied, Drive '
3183            207265736574
3184            2064656E6965
3185            642C20447269
3186            766520
3187    =0A8D   003A            deniedrv        db      0,':'
3188    =0A8F   20436F6E736F                    db      ' Console '
3189            6C6520
3190    =0A98   00              deniedcns       db      0
3191    =0A99   2050726F6772                    db      ' Program '
3192            616D20
```

```
3193
3194  =0AA2 313233343536      deniedprc       db      '12345678',0
3195        373800
3196  =
3197  =                       if BNPN
3198  =
3199  =                       ;       bdos stack switch variables and stack
3200  =                       ;       used for all bdos disk functions
3201  =
3202  =                               org     ((offset $) + 1) and 0fffeh
3203  =
3204  =                               ; 69 word bdos stack
3205  =
3206  =0AAC CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3207  =0AB2 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3208  =0AB8 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3209  =0ABE CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3210  =0AC4 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3211  =0ACA CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3212  =0AD0 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3213  =0AD6 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3214  =0ADC CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3215  =0AE2 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3216  =0AE8 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3217  =0AEE CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3218  =0AF4 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3219  =0AFA CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3220  =0B00 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3221  =0B06 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3222  =0B0C CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3223  =0B12 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3224  =0B18 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3225  =0B1E CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3226  =0B24 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3227  =0B2A CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3228  =0B30 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3229  =0B36                   bdosstack       rw      0
3230  =
3231  =0B36                   save_sp         rw      1
3232  =0B38                   ss_save         rw      1
3233  =0B3A                   sp_save         rw      1
3234  =
3235  =                       ;       local buffer area:
3236  =
3237  =0B3C                   info_fcb        rb      40      ;local user FCB
3238  =0B64                   save_fcb        rb      16      ;fcb save area for xfcb search
3239  =
3240  =0B74 0000              mxdiskqd        dw      0       ;link
3241  =0B76 0000                              db      0,0     ;net,org
3242  =0B78 0300                              dw      qf_keep+qf_mx   ;flags (MX queue)
3243  =0B7A 4D5864697368              db      'MXdisk  '
3244        2020
3245  =0B82 00000100                  dw      0,1     ;msglen,nmsgs
```

```
3246
3247  =0886 0000u000                    dw      0,0     ;nq,dq
3248  =088A 01000000                    dw      1,0     ;msgcnt,out
3249  =088E 0000                        dw      0       ;buffer ptr
3250  =
3251  =0890 00              mxdiskqpb   db      0       ;flgs
3252  =0891 00                          db      0       ;net
3253  =0892 7408                        dw      mxdiskqd        ;qaddr
3254  =0894 0100                        dw      1       ;nmsgs
3255  =0896 0000                        dw      0       ;buffer
3256  =0898 4D5864697368                db      'MXdisk  '
3257        2020
3258  =
3259  =                       ;       Error Message sync param block for mutual exclusion
3260  =
3261  =08A0 0000              msg_spb     dw      0       ;link   Error Message sync parameter block
3262  =08A2 0000                          dw      0       ;owner
3263  =08A4 0000                          dw      0       ;wait
3264  =08A6 0000                          dw      0       ;next
3265  =                       endif
3266  =
3267  =                       if BCPM
3268  =
3269  =                       ;
3270  =                       ;       special 8086 variables:
3271  =                       ;
3272  =                       ioloc           db      0       ;iobyte
3273  =                       user_parm_seg   dw      0       ;holds user parameter seg during load
3274  =                       nallocmem       db      0       ;no. of allocated memory segments
3275  =                       ncrmem          db      0       ;no. of available memory segments
3276  =                       crmem           dw      0,0     ;memory table (16 elements)
3277  =                                       dw      0,0,0,0,0,0
3278  =                                       dw      0,0,0,0,0,0,0,0
3279  =                                       dw      0,0,0,0,0,0,0,0
3280  =                                       dw      0,0,0,0,0,0,0,0
3281  =                       ;
3282  =                       mem_stack_length        equ     40
3283  =                       memstack        rs      mem_stack_length
3284  =                                               ;8 possible allocations
3285  =                       stbase  equ     word ptr 0
3286  =                       stlen   equ     word ptr 2
3287  =                       ccpflag equ     byte ptr 4
3288  =                       nccpalloc       db      0       ;number of current ccp allocations
3289  =                       mem_stk_ptr     dw      0       ;current memory stack location
3290  =
3291  =                       stackarea       rw      ssize   ;stack size
3292  =                       endstack        rb      0       ;top of stack
3293  =                       ;
3294  =                       endif
3295  =                                       org     0fffh
3296  =                       if CCPMOFF
3297  =                                       org     0bffh
3298  =                       endif
```

```
3299
3300  =08FF 00                    db      0
3301
```

```
3302
3303                                    eject | end
3304
3305
3306  END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR: 59%
```

```
ACBCNS          0000 N   577#  578  1065  1152
ACBLEN          000C N   581#
ACBNATCH        0001 N   578#  579  1150
ACBNAME         0004 N   580#  581  1039
ACBPD           0002 N   579#  580  1018  1021  1142
ACTUALRC        08DA V  3019#
ADDLAST         000C N  2995#
ADRIVE          0916 V  3082#
ALLOCA          08B0 V  2991#
ARECORD         0917 V  3083#
ARECORD1        091A V  3085#
AREA            080D V  2852# 2853
ASCONS          0232 L  1144  1146  1149#
ASCHK           0217 L  1030  1052  1130#
ASCND           0241 L  1148  1155#
ASCOK           0242 L  1151  1154  1157#
ASERR           01BD L  1049  1077#
ASFOUNDID       0161 L  1027  1029#
ASGEXIT         01FE L  1096  1110#
ASHOK           018F L  1031  1053  1057#
ASNAME          0168 L  1019  1034#
ASNEXT1         0154 L  1024# 1028
ASNEXT2         016C L  1037# 1055
ASNOM           0183 L  1026  1032  1047#
ASNQPD          01DF L  1094# 1100
ASNWAKE         0215 L  1123  1127#
ASOK            01CF L  1073  1075  1086#
ASOKDISP        0201 L  1079  1089  1114#
ASOKNAME        0188 L  1045  1050#
ASQFIX          01EC L  1098  1101#
ATTACH          0609 L  1705  1707  1710#
ATTRIBUTES      08DE V  3023#
AXSAV           0628 V  2705#
BACKSP          0587 L  1656# 1665
BACKX           051F L  1624# 1631
BACKXXX         0526 L  1625  1627#
BCON            045C L  1521  1523#
BDOS            0005 N    69#   70  2780  3064  3267
                        91#  234  2413  2414  2415  2417  2418  2419  2420  2421
                       2422 2423  2424  2425  2426  2427  2428  2429  2430  2431
                       2432 2433  2434  2435  2436  2437  2438  2439  2440  2441
                       2447 2448  2449  2450  2451  2454  2460  2461  2486  2487
                       2488 2489  2490  2491  2492  2493  2494
BDOSMOD         0020 N  2253#
BDOSMODBIT      0008 N   100#
BDOSSTACK       0336 V  3229#
BELL            0007 N   719#
BESTLEN         062C V  2710#
BESTMAU         0630 V  2712#
BESTSI          062E V  2711#
BESTSTART       062A V  2709#
BLKMSK          08B8 V  3002#
BLKNUM          0823 V  2872#
```

```
BLKOFF          0822 V  2671#
BLKSAV          08BA V  3001#
BLOCKOUT        0435 L  1485  1503#
BLOOP           0448 L  1514# 1531
BMPM            FFFF N    70# 2828  2832  2836  2915  2948  2961  2971  3025  3197
BRET            0472 L  1512  1532#
BUFFA           08AA V  2988#
BVERNUM         007A V  2331# 2336#
BXSAV           0626 V  2704#
CAATCH          00FA L   924   936#
CARET           00FC L   923   937#
CASLEEP         00E6 L   925   928#
CATTACH         0000 N   653#  654   923   924   936   965   966   971  1072  1074
                        1087  1393  1706
CBLEN           0004 N   753#  1507
CBUFF           0000 N   751#  752   1510
CBSEG           0002 N   752#  753   1509
CBTMP           000C N   663#  664   1596  1655
CCB             0054 V   584   1223  1263  2300#
CCBLEN          002C N   679#  1278
CCBOK           00CB L   899   902#
CCBVERIFY       00B9 L   890   896#  948
CCBLUMN         0006 N   657#  658   1575  1595  1618  1625  1630  1656  1658  1684
                        1779  1789  1858  1881
CCBNATTCHENTRY  0086 L   821   893#
CCBSLEEP        0022 N   676#  677   1269  2023  2042  2047
CCPM            FFFF N    67#  244   586   2335
CCPMOFF         FFFF N  2778# 2832  3296
CCRET           0683 L  1780  1782  1784  1786  1787  1789#
CDDETACH        011F L   966   969#
CDRET           012D L   965   975#
CDRMAXA         08A6 V  2986#
CFBUFP          0020 N   688#  1267  1813  1815  1992
CFCOMPC         0002 N   684#  1597  1652  1654  1767
CFCONOUT        0008 N   686#  1267  2018  2032  2041
CFLAG           0004 N   655#  656   1267  1597  1652  1654  1767  1812  1815  1992
                        2009  2018  2032  2041
CFLISTCP        0001 N   683#
CFSWITCHS       0004 N   685#
CFVOUT          0010 N   687#  1267  1992  2009
CHECKA          08AE V  2990#
CHECKDISK       0901 V  3050#
CHECKFCBRET     08F3 V  3036#
CHKCIRLP        0687 L  1559  1774  1793#
CHKOLISTFLAG    08EF V  3032#
CHKSIZ          08C3 V  3007#
CIEXIT          0330 L  1329  1331#
CIGOTC          0732 L  1934  1937#
CIO             0004 N    90#  225   226   227   228   229   757   857  1297  2401
                        2402  2405  2406  2409  2410  2411  2497  2498  2499  2500
                        2523  2524  2525  2526  2530  2538  2539  2540  2541  2542
                        2544
CIOATTACH       00CC L   884   907#  935
```

```
CLIODETACH          0108 L   943   952# 1275  1288
CLIOMOD             0018 N  2249#
CIOMODBIT           0010 N   101#
CLIOSTAT            0396 L   826  1386#
CLIOTERM            02CE L   825  1257#
CIVC                0726 L  1928  1931#
CLIACB              0458 V  2608#
CLIBPAGE            046F V  2618#
CLICHAIN            0323 V  2590#
CLICMDTAIL          03A8 V  2597# 2613
CLICNS              0472 V  2623#
CLICUSPOPB          044B V  2602#
CLIDFIL             0476 V  2627#
CLIDMA              0325 V  2592#
CLIDMAOFST          031D V  2587#
CLIDMASEG           031F V  2588#
CLIDSK              0474 V  2625#
CLIERR              046D V  2617#
CLIERRMODE          0475 V  2626#
CLIFCB              042A V  2600# 2614
CLILOGDSK           0471 V  2619#
CLINET              03A5 V  2595#
CLIPCB              0467 V  2613#
CLIPD               046B V  2616#
CLIPFLAG            0321 V  2589#
CLIPPD              03A6 V  2596# 2604
CLISPB              0651 V  2736  2744#
CLITERM             0324 V  2591#
CLIUSER             0473 V  2624#
CLSTATTCHENTRY      009A L   820   871#
CMAXBUFSIZ          0010 N   666#  667
CMCTLC              0001 N   739#
CMCTLS              0002 N   740#
CMGET               0329 L  1313  1318#
CMIMIC              0008 N   659#  660  1819
CMOD                0090 V  2360#
CMODEENTRY          0318 L   828  1305#
CMRAWD              0004 N   741#
CMRSX1              0100 N   742#
CMRSX2              0200 N   743#
CMSOURCE            0009 N   660#  661
CNCHAR              0007 N   658#  659  1378  1906  1933  1938  1940  1947
COBUFFER            0784 L  1977  1990#
COCRTISGET          07B4 L  2014  2016#
COCONOUTSLEEP       07BA L  2021#
COGETCONOUT         07AC L  1974  2012#
COGOTCONOUT         07CD L  2019  2031#
COMMONDMA           08C9 V  3017#
COMPCBCKS           0811 V  2856#
COMPOUT             065E L  1767  1778#
CONASSIGNENTRY      0131 L   814   981#
CONATTACHENTRY      00B2 L   811   887# 1715
CONC                0393 L  1377  1382#
```

```
CONDLTACHENTRY   0105 L   812   946#
CONINENTRY       0320 L   801  1323#
CONINF           0718 L  1326  1348  1376  1420  1428  1581  1926#
CONINIT          05EF L  1325  1337  1370  1437  1481  1545  1572  1693#
CONOIP           0335 L  1328#
CONOUT           0647 L  1756# 1839  1854  1857  1865  1866  1877  1883
CONOUTENTRY      0340 L   802  1335#
CONOUTF          0753 L  1340  1356  1443  1527  1558  1616  1617  1627  1628  1629
                         1661  1662  1663  1683  1769  1963#
CONOWAKE         07FE L  2043  2050#
CONP             065E L  1772  1775#
CONPRINTENTRY    0473 L   816  1537# 1744
CONREADENTRY     04A9 L   808  1567#
CONSTATENTRY     0370 L   810  1368#
CONSTF           06FC L  1399  1422  1899#
CONWRITEENTRY    03DB L   807  1431#
COR              0392 L  1372  1375  1380#
COTAB            0340 L  1339  1341#
COVC             075E L  1965  1968# 1987  2029
COXRET           081A L  2054  2061#
CPC              000A N   661#  662
CPLIST1T         06A1 L  1814  1817#
CPN0             06AE L  1821  1823#
CPR1             047E L  1551# 1563
CPR2             0487 L  1551  1553#
CPR3             0493 L  1556#
CPRE             04A8 L  1552  1556  1564#
CPRET            0680 L  1809  1811  1816  1825#
CQBUFF           0020 N   675#  676  2000  2004
CQPBBUFFPTR      001E N   674#  675  1948  2001
CQPPFILL         0019 N   671#  672
CQPBFLAGS        0018 N   670#  671  1949  2005
CQPBNMSGS        001C N   673#  674
CQPBQADDR        001A N   672#  673  1946  1999
CQUEUE           0002 N   654#  655   930   970   973  1092
CR               000D N   723# 1586  1616  1683  1729  1787  1865
CRLF             06D7 L  1863# 1878
CRLFP            06F4 L  1635  1640  1868#
CRLFP0           06EE L  1881# 1883
CRLFP1           06F8 L  1882  1884#
CRLFSTR          0A45 V  3172#
CRSV0            000D N   664#  665
CS               SREG V   838   843  1741
CSEC2            03BB L  1400  1403#
CSECL            03B2 L  1371  1394  1398#
CSEND            03AA L  1392  1395#
CSIREADQ         073C L  1936  1943#
CSMABORT         0020 N   698#
CSMBACKGROUND    0002 N   694# 1976
CSMBUFFERED      0001 N   692# 1976
CSMCTRLD         0100 N   701# 1810  1972  2013
CSMCTRLP         0200 N   702# 1808  1972
CSMCTRLS         0080 N   700#
```

```
CSMFILEFULL       0040 N   699#
CSMNOSWITCH       0008 N   696# 1972
CSMPURGING        0004 N   695#
CSMSUSPEND        0010 N   697#
CSTATE            000E N   665#  666  1807  1970
CSTRET            0718 L  1910  1913#
CSTRICOL          0005 N   656#  657  1575  1619  1624  1882
CSTTRUE           0716 L  1907  1911#
CSTVC             0707 L  1901  1904#
CTDLCB            02F1 L  1265  1273#
CTDLCB            02FF L  1264  1282#
CTL               005E N   731# 1839
CTLC              0003 N   716# 1377  1378
CTLD              0004 N   717#
CTLE              0005 N   718# 1613
CTLH              0008 N   720# 1593  1627  1629  1661  1663  1732  1785
CTLOUT            0681 L  1649  1674  1829#
CTLP              0010 N   724#
CTLQ              0011 N   725#
CTLR              0012 N   726# 1639
CTLS              0013 N   727#
CTLU              0015 N   728# 1634
CTLX              0018 N   729# 1623
CTLZ              001A N   730#
CTNLCB            0306 L  1285# 1292
CTRET             0318 L  1286  1294#
CURBLBA           0371 V  2911#
CURDMA            0920 V  3089#
CURDMASEG         091E V  3088#
CURDSK            08A0 V  2969#
CURPOS            08F7 V  3042#
CURRMAU           0632 V  2713#
CURRMPB           0636 V  2715#
CURRSI            0634 V  2714#
CUSLEEP           0024 N   677#  678  1980
CVC               000B N   662#  663
CVCMXQ            0016 N   669#  670
CVINQ             0012 N   667#  668  1908  1945
CVOUTQ            0014 N   668#  669  1998
CVSLEEP           0026 N   678#  679
CWNOTAB           0404 L  1448  1451#
CWR1              03F0 L  1440# 1453
CWRE              0408 L  1442  1454#
CWTAB             0401 L  1446  1449#
DATBCNA           0884 V  2993#
DAYFALL           004F V  2293#
DBLK              0891 V  2956#
DCNT              088F V  2955# 3094
DEBLOCKFX         086F V  2909#
DEBUGINT          00E1 N    56#
DELIMENTRY        0409 L   830  1458#
DENIEDCNS         0A98 V  3190#
DENIEDDRV         0A8D V  3187#
```

```
DENIEDMSG       0A72 V   3182#
DENIEDPRC       0AA2 V   3194#
DEVVER          000C V    776#
DFPASSWORD      0B95 V   2959#
DGET            0414 L   1465  1468#
DIO1            03D3 L   1424#  1427
DILCAN          03D4 L   1423  1426#
DIOSIS          03C8 L   1419  1421#
DIRBCBA         08B2 V   2992#
DIRBLK          08C1 V   3006#
DIRCNT          0817 V   2861#
DIRIDENTRY      03BE L    806  1407#
DIRLOC          090F V   3074#
DIRMAX          08BF V   3005#
DISPATCHER      0038 N   2266#
DLOG            0808 V   2844#
DLR             006A V   2318#
DMADFST         0885 V   2943#
DMASEG          0887 V   2944#
DMINX           0911 V   3076#
DUMICLOSE       08FB V   3045#
DPBADDR         08AC V   2989#
DPBLIST         0011 N   3012#
DPIR            0922 V   3093#
DREC            091C V   3086#
DRL             006C V   1123  2319#
DRVLBLA         08A8 V   2987#
DS              SREG V   1016  1017  1022  1035  1040  1043  1063  1066  1440  1440
                         1441  1506  1506  1511  1517  1517  1519  1553  1553  1554
                         1605  1605  1606  1645  1645  1647  1672  1672  1673  1676
                         1676  1677  1681  1681  1682  1753
DSKERR          0942 V   3116#
DSKMSG          0932 V   3113#
DSKRECL         0080 N     49# 2584  2592
DSPTCHIOS       0622 V   2698#
EABORT          0028 N    480#
EACTIVEPD       0022 N    474#
EBADENTRY       0002 N    442#
EBADFNAME       0018 N    464#
EBADFTYPE       0019 N    465#
EBADLOAD        001C N    468#
EBADOPEN        001E N    470#
EBADREAD        001D N    469#
ECHOC           0611 L   1329  1719# 1838
EFIXUPREC       0029 N    481#
EFLAGOVRRUN     0005 N    445#
EFLAGUNDERRUN   0006 N    446#
EILLCNS         0013 N    459# 1184
EILLDISK        0017 N    463#
EILLFLAG        0004 N    444#
EILLLST         0025 N    477# 1200
EILLMD          001B N    467#
EILLPASSWD      0026 N    478#
```

```
EMPTYBCBA           0875 V   2913#
ENCLIP              0016 N    462#
ENCLIQ              0010 N    456#
ENDLIST             08C9 V   3011#  3012
ENDSEG              0044 V   2276#
ENDATTACH           0024 N    476#   927
ENDCHAR             001A N    466#
ENDCNSMATCH         0015 N    461#
ENDCSEG             0021 N    473#
ENDMEMORY           0003 N    443#
ENDMISC             0027 N    479#
ENDPD               000C N    452#
ENDPDNAME           0014 N    460#  1048
ENDQBUF             0008 N    448#
ENDQD               0007 N    447#
ENDQUEUE            0009 N    449#
ENDUNIMPLEMENTED    0001 N    441#
ENDUOWNER           0020 N    472#   968   1076
ENDUMD              0012 N    458#
ENTRY               007D L    768    835#
ENULLCMD            001F N    471#
EOFRE               04C3 L   1587#  1590
EPDNOTERM           0023 N    475#
EQEMPTY             000E N    454#
EQFULL              000F N    455#
EQINUSE             000A N    450#
EQPROTECTED         000D N    453#
ERRDRV              092F V   3110#
ERRINTERCEPT        0092 V   2364#
ERRORMODE           0893 V   2957#
ERRPDADDR           0930 V   3111#
ERRTYPE             081B V   2865#
ES                  SREG V   1140   1141   1142   1150   1152   1158
ESSAV               0624 V   2703#
EXTMSK              08BC V   3003#
EXTVAL              0914 V   3080#
FALSE               0000 N     47#    66     69   1120   2361   2700
FBDOSTERM           052D N    234#
FCALLBIOS           0032 N    149#
FCBDSK              080B V   2850#  2867
FCBLEN              0020 N     50#  2600
FCCONATTCH          00A2 N    194#
FCIOSTAT            0418 N    227#
FCIOTERM            0417 N    226#
FCLOAD              010E N    201#
FCUNASSIGN          0095 N    181#
FCUNATTACH          0092 N    177#
FCUNDETACH          0093 N    179#
FCUNOUT             0002 N    108#
FCUNPRINT           040E N    225#
FCUNREAD            000A N    116#
FCUNWRITE           0009 N    115#
FCQREAD             008A N    169#
```

```
FCQWRITE          008C N   171#
FCREATEPROC       0090 N   175#
FDELIM            006E N   159#
FDISPATCH         008E N   173# 1125  1396
FFCLOSE           0010 N   122#
FFFLAG            081A V  2864#
FFINDPDNAME       0214 N   208# 1041
FFLAGSET          0085 N   164#
FFLAGWAIT         0084 N   163#
FFOPEN            000F N   121#
FFREADRDM         0021 N   140#
FFREADSEQ         0014 N   127#
FFRECALL          003A N   157#
FFREEMEM          0039 N   156#
FGETDEFDISK       0019 N   132#
FILEID            08FE V  3048#
FINDXFCB          0814 V  2859#
FINFLAGSET        0203 N   205#
FLAGMIN           0003 N    63#
FLAGS             0056 V  2301#
FLAGSEC           0002 N    62#
FLAGTICK          0001 N    61#
FLOAD             010A N   200#
FLSTATTACH        009E N   190#
FLSTDETACH        009F N   191#
FLSTOUT           0005 N   111#
FLUSHED           0902 V  3051#
FMALLOC           0080 N   160#
FMAUALLOC         0309 N   216#
FMAUFREE          030A N   219#
FMEMFREE          0082 N   161#
FMLALLOC          030B N   220#
FMLFREE           030C N   221#
FNAMSIZ           0008 N    53#
FNDABORT          0217 N   211#
FNDABORTSPEC      0219 N   213#
FOKABORT          0218 N   212#
FPARSEFILENAME    0098 N   184#
FQMAKE            0086 N   165#
FQOPEN            0087 N   166#
FQREAD            0089 N   168# 1935
FQWRITE           0088 N   170# 2006
FRCONIN           0402 N   228#
FRCONOUT          0403 N   229#
FREEMODE          08F6 V  3039#
FREFROUT          0052 V  2299# 3055
FSETDMA           001A N   133#
FSETDMAB          0033 N   150#
FSETPRIOR         0091 N   176#
FSETRNDREC        0024 N   143#
FSHARE            0308 N   217#
FSLEEP            0212 N   206#  933  1982  2025
FSYNC             0215 N   209# 1010
```

```
FTERMINATE      008F N   174# 2058
FTYPSIZ         0003 N    54#
FUNC            0889 V  2945#
FUNCTAB         0043 V   801#  838
FUNSYNC         0216 N   210# 1060  1081
FUSERCODE       0020 N   139#
FWAKEUP         0213 N   207#  972  1108  1268  2045
FXINTRN         087A V  2922#
GETCCBADR       0296 L   900  1189  1211#
GETCCBADRSPEC   029D L  1067  1221#
GETDEFCONENTRY  0244 L   815  1162#
GETDEFLSTENTRY  024E L   822  1170#
GETLCBADR       02E4 L   880  1205  1235#
GUXIOS          081E L  1903  1930  1967  2040  2076#
HASH            084B V  2890#
HASHL           084F V  2891#
HASHSEG         08B6 V  2994#
HIGHEXT         089E V  2967#
INCRPDCNT       08F5 V  3038#
INDISP          0622 V  1013  1120  2700#
INFO            0881 V  2932#
INFOFCB         08BC V  3237#
INIT            0042 L   767   791#
INITPD          0477 V  2317  2322  2634#
INITIOS         05B0 V  2664# 2667
INITUDA         04F0 V  2663# 2665
INITFLAG        0623 V  2701#
IOAUXIN         0005 N   251#
IOAUXOUT        0006 N   252#
IOCONIN         0001 N   247#  266# 1929
IOCONOUT        0002 N   248#  267# 1966  2039
IOCONST         0000 N   246#  265# 1902
IOFLUSH         000C N   258#  261   290#  291
IOLIST          0004 N   250#  268# 2073
IOLISTST        0003 N   249#
IOPOLLDEV       000D N   259#  282#
IOREAD          000A N   256#  276#
IOSELDSK        0009 N   255#  272#
IOSTATLINE      0008 N   254#
IOSWITCH        0007 N   253#
IOWRITE         000B N   257#  277#
L22             05AA L  1653  1657  1659  1666# 1678
LBLOCKENTRY     0426 L   832  1488#
LCB             0086 V  1245  1284  2354#
LCBLEN          000A N   705# 1291
LCBOK           00AF L   879   882#
LCBVERIFY       009D L   868   875#  942
LDCNT           088F V  3094#
LDIAB           0273 V  2585#
LDIABSIZ        00AA N    64# 2585
LF              000A N   722# 1590  1617  1730  1866
LINELEN         0550 L  1597  1640#
LINFO           0910 V  3075#
```

```
LISTF                0818 L  1365  1522  1822  2072#
LISTOUTENTRY         0356 L   805  1359#
LOCKCNT              03FD V  3047#
LOCKMAX              008A V  2356#
LOCKROUT             0907 V  3057#
LOCKSHELL            08F2 V  3035#
LOCKSP               08F0 V  3033#
LOCKUNLOCK           08F4 V  3037#
LODBOBO              01ED V  2580#
LODBASEP             018C V  2569#
LODCHAIN             01E9 V  2576#
LODDISK              01EB V  2578#
LODDMA               01F3 V  2584#
LODFCB               01C2 V  2572#
LODFIFTY             01EC V  2579#
LODFIXREC            01EF V  2582#
LODFIXREC1           01F1 V  2583#
LODINDMA             01E6 V  2573#
LODLBYTE             01EE V  2581#
LODLSTR              018A V  2568#
LODNLDT              019E V  2570#
LODNRECS             01E8 V  2574#
LODPD                01C0 V  2571#
LODUDA               0168 V  2567#
LODUSER              01EA V  2577#
LOGFXS               0850 V  2893#
LRET                 0A0D V  2853#
LSTATTACHENTRY       0096 L   817   865# 1364  1496
LSTDETACHENTRY       0100 L   818   940#
MAKEFLAG             08D9 V  3018#
MAKEXFCB             0813 V  2858#
MAL                  0076 V  2324#
MAXALL               088D V  3004#
MDUL                 0058 V  2302#
MEM                  0003 N    89#  217   218   219   220   221  2462  2463  2464  2465
                                2466  2467  2504  2505  2506
MEMCNT               0640 V  2724#
MEMMOD               0010 N  2245#
MEMMODBIT            0004 N    99#
MEMSPB               0641 V  2367  2727#
MFL                  005A V  2303#
MMP                  004C V  2291#
MODULEMAP            0046 V  2278#
MODULETABLE          0000 N  2236#
MPM                  0000 N    66#   67   264   583  2330
MSGSPB               0BA0 V  2752  3261#
MULTCNT              0894 V  2958#
MULTNUM              0818 V  2862#
MULTSEC              0820 V  2868#
MXDTSKQD             0B74 V  2551  3240# 3253
MXDISKQPB            0B90 V  3251#
MXLOADQD             0194 V  2323  2551# 2560
MXLOADQPB            01B0 V  2559#
```

```
NBNT              0467 L  1524  1526#
NBNXT             046A L  1522  1525  1528#
NCCP              0049 V  2286#
NCIODEV           0085 V  2352#
NCNS              0047 V  1900  1927  1964  2286#
NCONDEV           0083 V  1182  1262  2350#
NDP8087           0091 V  2361#
NET               0007 N    93# 2477  2478  2479  2480  2481  2482
NETBUD            0030 N  2262#
NETJOBIT          0040 N   103#
NFLAGS            004A V  2289#
NINIT             05F8 L  1691  1702#
NLST              0048 V  2287#
NLSTDEV           0084 V  1198  1283  2351#
NOTBS             067C L  1785  1787#
NOTCR             04C6 L  1586  1588#
NOTE              051A L  1613  1621#
NOTH              04E3 L  1593  1599#
NOTIRP            0092 L   823   824   851#
NOTR              050D L  1639  1668#
NOTRUB            04FF L  1602  1610#
NOTU              054B L  1634  1637#
NOTX              053F L  1623  1632#
NSLAVES           004E V  2292#
NTLOG             0804 V  2839#
NXIOSFUNCS        000C N   261#  291#
NXTCCB            0205 L  1264# 1279
OFFSETV           08C5 V  3006#
OLDPTYPE          0819 V  2863#
OPENCNT           08FC V  3046#
OPENEND           080A V  2846#
OPENMAX           0088 V  2357#
OPENROUT          0905 V  3056#
OPENVEC           0088 V  2355#
OSIF              0087 L   841#  933   974  1011  1042  1061  1082  1109  1126  1271
                         1396  1950  1983  2007  2026  2048  2059
OSINT             00E0 N    55#   56
OSSEG             0040 V  2274#
OSVERNUM          007C V  2332# 2337#
OWNER8087         008C V  2358#
PABORT            0021 N   379#  380
PACKEDDCNT        08A1 V  2973#
PARAMETERSEGMENT  092B V  3105#
PARLG             080C V  2851#
PATCH             082A L  2084  2091#
PBUBCNT           0879 V  2918#
PBLOCKENTRY       041A L   831  1473#
PCALL             0001 N   425# 1374
PCALTLC           0008 N   428#
PCALTLO           0080 N   430#
PCALTLS           0002 N   426#
PCMOD             0018 N   374#  375
PCNROUT           0004 N   427# 1338  1446  1524  1771
```

```
PCMKSX          0300 N   431#
PCLS            0020 N   378#  379  1153  1166  1188  1219  1389  1703
PCONMODE        0022 N   380#  381  1315  1319  1333  1374  1446  1484  1771
PDADDR          08A4 V  2976#
PDCNT           089D V  2963#
PDLEN           0030 N    59#
PDSK            0012 N   369#  370
PERMSG          0960 V  3118  3124#
PF8087          8000 N   420#
PFACTIVE        0100 N   413#
PFCHILDALORT    0800 N   416#
PFCTLC          0080 N   412# 2052  2055
PFCTLD          0400 N   415#
PFDSKLD         2000 N   418# 1147
PFKEEP          0002 N   405#
PFKERNAL        0004 N   406# 2638
PFLAG           0006 N   366#  367  1147  1690  1701  2035  2036  2052  2053  2055
PFNOCTLS        1000 N   417#
PFNOKBD         4000 N   419#
PFPURE          0008 N   407#
PFRAW           0040 N   410# 1690  1701
PFRESOURCE      0020 N   409#
PFSYS           0001 N   404# 2638
PFTABLE         0010 N   408#
PFTEXPKEEP      0200 N   414# 2036
PHYMSK          08C8 V  3010#
PHYOFF          0870 V  2910#
PHYSHF          08C7 V  3009#
PLASTBCBA       0877 V  2917#
PLDSK           0014 N   371#  372
PLINK           0000 N   362#  363  1092  1095  1097  1099  1102  1103  1104  1105
                1106
PLR             006E V  2320#
PLST            0024 N   381#  382  1175  1204  1244  1361  1498
PLUSER          0015 N   372#  373
PMEM            0016 N   373#  374
PNAME           0008 N   367#  368
PNAMSIZ         0008 N    51#   52    53   368   581
PPARENT         001E N   377#  378
PPRET           002C N   382#  383
PPRIOR          0005 N   365#  366
PRCSMSG         0629 L  1736#
PRFCB           0A5D V  3177#
PRFCB1          0A65 V  3179#
PRFX            0A48 V  3173#
PRFX1           0A5A V  3176#
PRNAME          063F L  1747#
PRSTK           062D L  1742# 1753
PRVPOS          08F9 V  3043#
PSCIOWAIT       0009 N   399#  931  1981  2024  2046
PSCRATCH        002E N   383#  384
PSDELAY         0002 N   392#
PSDQ            0006 N   396#
```

```
PSFLAGWAIT      0008 N    398#
PSNQ            0007 N    397#
PSPOLL          0001 N    391#
PSRUN           0000 N    390#  2636
PSSLEEP         0005 N    395#
PSSWAP          0003 N    393#
PSSYNC          000A N    400#
PSTAT           0004 N    364#   365
PSTERM          0004 N    394#
PTHREAD         0002 N    363#   364   1023   1025   1036
PTKCNT          0019 N    375#   376
PUDA            0010 N    368#   369
PUL             005C V   2304#
PUSER           0013 N    370#   371
PWAIT           001A N    376#   377
PWMODE          08DD V   3022#
PWSCRTCH        002E N    384#
QBUF            001A N    528#   529
QDLEN           001C N    529#
QDQ             0012 N    524#   525
QFDEV           0040 N    540#
QFHIDE          0004 N    535#
QFKEEP          0002 N    534#  2553   3242
QFLAGS          0004 N    520#   521
QFMX            0001 N    533#  2553   3242
QFRPL           0020 N    539#
QFRSP           0008 N    537#
QFTABLE         0010 N    538#
QLINK           0000 N    517#   518
QLR             0074 V   2323#
QMAU            0060 V   2307#
QMSGCNT         0016 N    526#   527   1909
QMSGLEN         000E N    522#   523
QMSGOUT         0018 N    527#   528
QNAME           0006 N    521#   522
QNAMSIZ         0008 N     52#   522    567
QNET            0002 N    518#   519
QNMSGS          0010 N    523#   524
QNQ             0014 N    525#   526
QORG            0003 N    519#   520
QPBBUFFPTR      0006 N    565#   566
QPBFLGS         0000 N    561#   562
QPBLEN          0010 N    567#
QPBNAME         0008 N    566#   567
QPBNET          0001 N    562#   563
QPBNMSGS        0004 N    564#   565
QPBQADDR        0002 N    563#   564
QSPB            0649 V   2727   2736#
QUL             005E V   2305#
RAWINIT         05F4 L   1347   1354   1416   1687#
RCUNINENTRY     0350 L    803   1345#
RCUNOUT         0359 L   1355#  1418
RCUNOUTENTRY    0356 L    804   1352#
```

```
RCOUNT          0913 V   3079#
RDDIRFLAG       0810 V   2855#
RDECH1          05BA L   1608  1674#
RDECH0          05AD L   1670#
READ            04AE L   1574#  1636
READEN          05CF L   1587   1677   1680#
READNO          04P8 L   1581#  1620   1666
READNX          04B7 L   1580#  1594   1609
READNXX         04FC L   1603   1609#  1626
RELOG           0821 V   2869#
REP0            0557 L   1643#  1651
REP1            0577 L   1643   1652#
REQUIRELDTABLE  080F V   3028#
RESEL           080F V   2854#
RET0            0628 L   1729   1730   1731   1732   1734#
RET2            06D6 L   1861#
RETURNSEG       092D V   3106#
RLOG            0800 V   2837#
RLR             0068 V    918    960   1069   1165   1173   1187   1203   1218   1242   1312
                       1361   1373   1388   1445   1498   1689   1700   1770   2034   2317#
RMF             090D V   3072#
RODMSG          0969 V   3118   3126#
RODSK           0806 V   2843#
ROFMSG          0978 V   3118   3129#
ROOTDCBA        0873 V   2912#
RSPSEG          0042 V   2275#
RTM             0002 N     88#   205    206    207    208    209    210    211    212    213
                       2400   2507   2508   2509   2510   2511   2512   2513   2514   2515
                       2516   2517   2518   2519   2520   2521   2533   2534
RTMMOD          0008 N   2241#
RTMMODBIT       0002 N     98#
RTNPOTSP        003C N   2269#
RUBOUT          007F N    722#  1602   1780
RWFXS           0860 V   2899#
SAVEFCB         0864 V   3236#
SAVEMOD         08DC V   3021#
SAVESP          0836 V   3231#
SAVEXFCB        08D8 V   3020#
SCFXS           086A V   2904#
SCL             0070 V   2321#
SDCNT           0909 V   3061#
SDCNTO          0908 V   3062#
SDGOOD          0265 L   1183   1186#
SDLGOOD         0284 L   1199   1202#
SEARCHA         0883 V   2933#
SEARCHABASE     088D V   2951#
SEARCHADIST     088B V   2950#
SEARCHL         083A V   2946#
SEARCHUSER0     0812 V   2857#
SECTOR          087D V   2925#
SECTPT          0888 V   3000#  3012
SELDSK          087F V   2929#
SELMSG          0987 V   3118   3132#
```

```
SERIAL          0036 V   788#
SETDEFCONENTRY  0258 L   813  1179#
SETDEFLSTENTRY  0277 L   819  1195#
SETPDFLAG       0900 V  3049#
SHELLRR         0926 V  3100#
SHELLSI         0924 V  3099#
SINGLE          0912 V  3077#
SLR             0096 V  2367#
SPSAVE          083A V  2735#  3233#
SRCHDISK        004B V  2290#
SS              SREG V
SSSAVE          0838 V  2788#  3232#
SUP             0001 N    87#   200   201  2403  2404  2407  2408  2412  2445  2452
                         2455  2459  2460  2470  2471  2472  2473  2495  2496  2527
                         2528  2529  2531  2532  2543
SUPERVISOR      0008 N   772#   843
SUPMOD          0000 N  2237#
SUPMODBIT       0001 N    97#
SYSDAT          0006 V   771#  1043  1065  2205
SYSCNT          00A0 V  2400#
TAB             0009 N   721#  1731  1853  1860
TAB0            06CA L  1857#  1859
TAB1            06C8 L  1853  1855#
TABOUT          06C0 L  1330  1342  1450  1525  1838  1843#
TEMPDISK        0050 V  2294#
THRDRT          0072 V  1023  1036  2322#
THRDSPB         0659 V  1009  1059  1080  2744  2752#
TICKSPERSEC     0051 V  2295#
TLOG            0802 V  2838#
TOD             007E V  2342#  3067#
TODDAY          007E V  2343#
TODHR           0080 V  2344#
TODLEN          0005 N    60#
TODMIN          0081 V  2345#
TODSEC          0082 V  2346#
TRACK           087B V  2924#
TRUE            FFFF N    46#  1013  2778
TRYH            04CB L  1592#
U80871LEN       015E N    58#
UA0             0327 V  2874  2877#
UA1             082D V  2877  2879#
UA2             0833 V  2879  2881#
UA3             0839 V  2881  2883#
UA4             083F V  2883  2885#
UA5             0845 V  2885  2887#
UALROOT         0825 V  2874#
UAX             0020 V  2167#
UBP             002C V  2173#
UBX             0022 V  2168#
UCONCCB         0062 V   898   901  1190  1265  1390  1704  2196#
UCX             0024 V  2169#
UDADVLEN        0019 N  2160#
UDASAVE         0029 V  3104#
```

```
UDBLK              000E V  2155#
UDCNT              000C V  2154#
UDEBUGCS           005E V  2193#
UDEBUGIP           005C V  2192#
UDCLIM             0066 V  1442  1466  1469  2198#
UDPASSWORD         0012 V  2158#
UDI                0028 V  2171#
UDINSYS            0060 N  2140# 2665
UDMADEST           0002 V  2148# 2160
UDMASEG            0004 V  2149#
UDPARAM            0000 V  2145#
UDSSAV             0032 V  2176#
UDX                0026 V  2170#
UERRORMODE         0010 V  2156#
UFSSAV             004C V  2183#
UFLAGSAV           004E V  2184#
UFUNC              0006 V  2150#
UINTNT             001B V  2164#
UINTICS            0050 V  2186#
UINTIDS            0052 V  2187#
UINTIES            0054 V  2188#
UINTISS            0056 V  2189#
UINSYS             0060 V  2194#
UIVECTORS          0038 V  2180#
UIVECTORS2         0044 V  2182#
ULEN               0100 N    57#   58  2663
ULSTCOD            0064 V   878   881  1206  1363  2197#
UMULTCNT           0011 V  2157#
UNKNOWN            0000 N    48# 2328  2641
UOSCS              005A V  2191#
UOSIP              0058 V  2190#
UPDCNT             001A V  2159#
URETSEG            0030 V  2175#
USEARCHA           0008 V  2152#
USEARCHABASE       000A V  2153#
USEARCHL           0007 V  2151#
USER               0000 N    86#  108   111   115   116   121   122   127   132   133
                             139   140   143   149   150   156   157   159   160   161
                             163   164   165   166   168   169   170   171   173   174
                             175   176   177   179   181   184   190   191   194
USERZEROPASS       0923 V  3095#
USI                002A V  2172#
USP                001C V  2165#
USRCODE            0880 V  2930#
USS                001E V  2166#
USTACKSP           0034 V  2177#
USTACKSS           0036 V  2179#
USTATSAV           0061 V  2195#
UUNUSED            0040 V  2181#
UWRKSEG            002E V  1017  1040  1063  1141  1440  1506  1553  1605  1645  1672
                           1676  1681  1742  1743  1745  2174#
VERSION            0078 V  2328#
VRECORD            0915 V  3081#
```

WFLAG	090E	V	3073#	
XCBCX	0001	N	301#	302
XCBDX	0003	N	302#	303
XCBFUNC	0000	N	300#	301
XCBLEN	0005	N	303#	
XDCNT	0815	V	2860#	
XE10	0A10	V	3120	3161#
XE11	0A29	V	3120	3166#
XE3	0995	V	3120	3136#
XE5	09B3	V	3120	3143#
XE6	09C7	V	3120	3147#
XE7	09DC	V	3120	3151#
XE8	09EB	V	3120	3154#
XE9	09FF	V	3120	3158#
XERRLIST	0946	V	3118#	
XFCBREADONLY	089F	V	2968#	
X1OS	0006	N	92#	
X1OS1F	008D	L	846#	2079
X1OSMOD	0028	N	848	2258#
X1OSMODBIT	0020	N	102#	
ZEROLENGTH	0015	N	2867#	