```
1
2  =
3          include cpyright.def
4  =                       ;*****************************************************
5  =                       ;*
6  =                       ;*              Concurrent CP/M-86 V2.1
7  =                       ;*              ========================
8  =                       ;*
9  =                       ;*              Copyright (c) 1982
10 =                       ;*
11 =                       ;*                 Digital Research
12 =                       ;*                   P.O.Box 579
13 =                       ;*           Pacific Grove, California 93950
14 =                       ;*
15 =                       ;*                   (408) 649-3896
16 =                       ;*                   TWX 9103605001
17 =                       ;*
18 =                       ;* All Information contained in this source listing is
19 =                       ;*
20 =                       ;*                      PROPRIETORY
21 =                       ;*                      ===========
22 =                       ;*
23 =                       ;* All rights reserved.  No part of this document
24 =                       ;* may be reproduced, transmitted, stored in a
25 =                       ;* retrieval system, or translated into any language
26 =                       ;* or computer language, in any form or by any means
27 =                       ;* without the prior written permission of Digital
28 =                       ;* Research, P.O Box 579, Pacific Grove, California.
29 =                       ;*
30 =                       ;*****************************************************
31
32
33                         ;*****************************************************
34                         ;*
35                         ;*      MP/M-86 Memory Management Module
36                         ;*
37                         ;*****************************************************
```

```
                        eject ! include system.def
                ;***************************************************
                ;*
                ;*       SYSTEM DEFINITIONS
                ;*
                ;***************************************************

FFFF            true        equ  0ffffh     ; value of TRUE
0000            false       equ  0          ; value of FALSE
0000            unknown     equ  0          ; value to be filled in
0080            dskrecl     equ  128        ; log. disk record len
0020            fcblen      equ  32         ; size of file control block
0008            pnamsiz     equ  8          ; size of process name
0008            qnamsiz     equ  pnamsiz    ; size of queue name
0008            fnamsiz     equ  pnamsiz    ; size of file name
0003            ftypsiz     equ  3          ; size of file type
00E0            osint       equ  224        ; int vec for O.S. entry
00E1            debugint    equ  osint+1    ; int vec for debuggers
0100            ulen        equ  0100h      ; size of uda
015E            u8087len    equ  ulen + 94  ; size of uda when process uses 8087
0030            pdlen       equ  030h       ; size of Process Descriptor
0005            todlen      equ  5          ; size of Time of Day struct
0001            flag_tick   equ  1          ; flag 0 = tick flag
0002            flag_sec    equ  2          ; flag 1 = second flag
0003            flag_min    equ  3          ; flag 2 = minute flag
00AA            ldtabsiz    equ  0aah       ; ldtablen=11, 10 entries

0000            mpm     equ     false       ; MP/M-86 or
FFFF            ccpm    equ     not mpm     ; CCP/M-86

0000            bCPM    equ     false       ; CP/M-86 BDOS
FFFF            bMPM    equ     not bCPM    ; multi-process bDOS
```

```
                        eject 1 include modfunc.def
                ;*****************************************************
                ;*
                ;*      CCP/M-86, MP/M-86 Inter-Module Function Definitions
                ;*
                ;*      Same calling conventions as User programs
                ;*      except CX = function instead of CL
                ;*             BX = 2nd parameter on entry
                ;*      (CH=module, CL=function # in module)
                ;*
                ;*****************************************************

                        ; Module definitions
                user            equ     0
                sup             equ     1
                rtm             equ     2
                mem             equ     3
                cio             equ     4
                bdos            equ     5
                xios            equ     6
                net             equ     7

                        ; Bits that represent present modules
                        ; in module_map
                supmod_bit      equ     001h
                rtmmod_bit      equ     002h
                memmod_bit      equ     004h
                bdosmod_bit     equ     008h
                ciomod_bit      equ     010h
                xiosmod_bit     equ     020h
                netmod_bit      equ     040h

                        ; Supervisor Functions
                ;f_sysreset     equ     (user * 0100h) + 0
                ;f_conin        equ     (user * 0100h) + 1
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_listout       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
```

```
125
126  =                    ;f_fdelete       equ     (user * 0100h) + 19
127  =  0014              f_freadseq       equ     (user * 0100h) + 20
128  =                    ;f_fwriteseq     equ     (user * 0100h) + 21
129  =                    ;f_fmake         equ     (user * 0100h) + 22
130  =                    ;f_frename       equ     (user * 0100h) + 23
131  =                    ;f_loginvector   equ     (user * 0100h) + 24
132  =  0019              f_getdefdisk     equ     (user * 0100h) + 25
133  =  001A              f_setdma         equ     (user * 0100h) + 26
134  =                    ;f_getallocvec   equ     (user * 0100h) + 27
135  =                    ;f_writeprotect  equ     (user * 0100h) + 28
136  =                    ;f_getrovector   equ     (user * 0100h) + 29
137  =                    ;f_setfileattr   equ     (user * 0100h) + 30
138  =                    ;f_getdpb        equ     (user * 0100h) + 31
139  =  0020              f_usercode       equ     (user * 0100h) + 32
140  =  0021              f_freadrdm       equ     (user * 0100h) + 33
141  =                    ;f_fwriterdm     equ     (user * 0100h) + 34
142  =                    ;f_filesize      equ     (user * 0100h) + 35
143  =  0024              f_setrndrec      equ     (user * 0100h) + 36
144  =                    ;f_resetdrive    equ     (user * 0100h) + 37
145  =                    ;f_accessdrive   equ     (user * 0100h) + 38
146  =                    ;f_freedrive     equ     (user * 0100h) + 39
147  =                    ;f_writerndzero  equ     (user * 0100h) + 40
148  =                    ;f_chain         equ     (user * 0100h) + 47
149  =  0032              f_callbios       equ     (user * 0100h) + 50
150  =  0033              f_setdmab        equ     (user * 0100h) + 51
151  =                    ;f_getdmab       equ     (user * 0100h) + 52
152  =                    ;f_getmaxmem     equ     (user * 0100h) + 53
153  =                    ;f_getabsmaxmem  equ     (user * 0100h) + 54
154  =                    ;f_allocmem      equ     (user * 0100h) + 55
155  =                    ;f_allocabsmem   equ     (user * 0100h) + 56
156  =  0039              f_freemem        equ     (user * 0100h) + 57
157  =  003A              f_freeall        equ     (user * 0100h) + 58
158  =                    ;f_userload      equ     (user * 0100h) + 59
159  =  006E              f_delim          equ     (user * 0100h) + 110
160  =  0080              f_malloc         equ     (user * 0100h) + 128
161  =  0082              f_memfree        equ     (user * 0100h) + 130
162  =                    ;f_polldev       equ     (user * 0100h) + 131
163  =  0084              f_flagwait       equ     (user * 0100h) + 132
164  =  0085              f_flagset        equ     (user * 0100h) + 133
165  =  0086              f_qmake          equ     (user * 0100h) + 134
166  =  0087              f_qopen          equ     (user * 0100h) + 135
167  =                    ;f_qdelete       equ     (user * 0100h) + 136
168  =  0089              f_qread          equ     (user * 0100h) + 137
169  =  008A              f_cqread         equ     (user * 0100h) + 138
170  =  008B              f_qwrite         equ     (user * 0100h) + 139
171  =  008C              f_cqwrite        equ     (user * 0100h) + 140
172  =                    ;f_delay         equ     (user * 0100h) + 141
173  =  008E              f_dispatch       equ     (user * 0100h) + 142
174  =  008F              f_terminate      equ     (user * 0100h) + 143
175  =  0090              f_createproc     equ     (user * 0100h) + 144
176  =  0091              f_setprior       equ     (user * 0100h) + 145
177  =  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_sdataaddr     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  =  052D                f_bdosterm      equ     (bdos * 0100h) + 45
235
```

```
                        eject ! include pd.def
                ;***************************************************
                ;*
                ;*      Process Descriptor - with the UDA associated
                ;*              with the PD, describes the current
                ;*              state of a Process under MP/M-CCP/M
                ;*
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 00|   link    |  thread   |stat |prior|   flag    |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 08|                       Name                    |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 10|    uda    | dsk | user| ldsk|luser|    mem    |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 18| cmod|tkcnt|   wait    | reserved  |  parent   |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 20| cns |abort|  conmode  | lst |     reserved    |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;* 28|        reserved       |   pret    |  scratch  |
                ;*   +-----+-----+-----+-----+-----+-----+-----+-----+
                ;*
                ;*      link    - Used for placement into System Lists
                ;*      thread  - link field for Thread List
                ;*      stat    - Current Process activity
                ;*      prior   - priority
                ;*      flag    - process state flags
                ;*      name    - name of process
                ;*      uda     - Segment Adress of User Data Area
                ;*      dsk     - Current default disk
                ;*      user    - Current default user number
                ;*      ldsk    - Disk program loaded from
                ;*      luser   - User number loaded from
                ;*      mem     - pointer to MD list of memory owned
                ;*                by this process
                ;*      cmod    - compatibility mode bits.
                ;*                80 -> F1 bit
                ;*                40 -> F2 bit
                ;*                20 -> F3 bit
                ;*                10 -> F4 bit
                ;*      tkcnt   - temp keep count, # times tempkeep has been
                ;*                turned on
                ;*      wait    - parameter field while on System Lists
                ;*      org     - Network node that originated this process
                ;*      net     - Network node running this process
                ;*      parent  - process that created this process
                ;*      cns     - default console #
                ;*      abort   - abort code
                ;*      lst     - default list #
                ;*      conmode - console mode for function 109
                ;*      reserved- not currently used
                ;*      pret    - return code at termination
                ;*      scratch - scratch word
```

```
                        ;*
                        ;*******************************************************
                        
0000                    p_link          equ     word ptr 0
0002                    p_thread        equu    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
001c                    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
0026                    p_pret          equ     word ptr p_lst + 8
002E                    p_scratch       equ     byte ptr p_pret + word
002c                    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
```

```
342
343  =  0030                pf_ctlc         equ     00030h  ; abort pending
344  =  0100                pf_active       equ     00100h  ; active tty
345  =  0200                pf_tempkeep     equ     00200h  ; don't terminate yet...
346  =  0400                pf_ctld         equ     00400h  ; explicit detach occured
347  =  0800                pf_childabort   equ     00800h  ; child terminated abnormally
348  =  1000                pf_noctls       equ     01000h  ; control S not allowed
349  =  2000                pf_dskld        equ     02000h  ; process was loaded from disk
350  =  4000                pf_nokbd        equ     04000h  ; ignore keyboard
351  =  8000                pf_8087         equ     08000h  ; process uses 8087
352  =                      ;
353  =                      ;       Process descriptor pcm_flag bit values
354  =                      ;               (console mode set by function 109)
355  =                      ;
356  =  0001                pcm_ll          equ     00001h  ; control C status for function 11
357  =  0002                pcm_ctls        equ     00002h  ; disable control S-Q
358  =  0004                pcm_rout        equ     00004h  ; raw output, no tabs or control P
359  =  0008                pcm_ctlc        equ     00008h  ; disable control C
360  =                      ;pcm_rsrv       equ     00040h  ; used by CP/M 3.0
361  =  0080                pcm_ctlo        equ     00080h  ; disable control O
362  =  0300                pcm_rsx         equ     00300h  ; 2 bits used by RSX for status
363
```

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

0000            q_link          equ     word ptr 0
0002            q_net           equ     byte ptr q_link + word
0003            q_org           equ     byte ptr q_net + byte
0004            q_flags         equ     word ptr q_org + byte
0006            q_name          equ     byte ptr q_flags + word
000L            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
```

```
                                ;*****************************************************
                                ;*
                                ;*      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 ! 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_nocli             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 1 include mem.def
                ;*******************************************************
                ;*
                ;*      Memory Descriptor Formats
                ;*
                ;*******************************************************

                        ; format while on MFL or MAL

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

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

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

                        ; format of spb (share parameter block)

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

```
                        eject ; include mpb.def
                    ;*****************************************************
                    ;*
                    ;*      Memory Parameter Block
                    ;*
                    ;*****************************************************

0000                mpb_start       equ     word ptr 0
0002                mpb_min         equ     word ptr mpb_start + word
0004                mpb_max         equ     word ptr mpb_min + word
0006                mpb_pdadr       equ     word ptr mpb_max + word
0008                mpb_flags       equ     word ptr mpb_pdadr + word

000A                mpblen          equ     mpb_flags + word


                        ; mpb_flags definition (also ms_flags)

0001                mf_load         equ     00001h
0002                mf_share        equ     00002h
0004                mf_code         equ     00004h
                    ; mf_data       equ     00008h
                    ; mf_extra      equ     00010h
                    ; mf_stack      equ     00020h
0040                mf_uda_only     equ     00040h

                    ;*****************************************************
                    ;*
                    ;*      Memory Free Parameter Block
                    ;*
                    ;*****************************************************

0000                mfpb_start      equ     word ptr 0
0002                mfpb_pd         equ     word ptr mfpb_start + word
0004                mfpblen         equ     mfpb_pd + word

                    ;*****************************************************
                    ;*
                    ;*      MAU Free Request
                    ;*
                    ;*      +----+----+----+----+----+----+
                    ;*      |   mau   |   sat   |  start  |
                    ;*      +----+----+----+----+----+----+
                    ;*
                    ;*****************************************************

0000                maf_mau         equ     word ptr 0
0002                maf_sat         equ     word ptr maf_mau + word
0004                maf_start       equ     word ptr maf_sat + word
0006                maflen          equ     maf_start + word
```

```
585  =
586  =                              eject | include memif.mem
587  =                          ;*****************************************************
588  =                          ;*
589  =                          ;*       Memory Management Module Interface
590  =                          ;*
591  =                          ;*****************************************************
592  =
593  =                                  CSEG
594  =                                  org     0
595  =
596  =0000 E93F00          0042        jmp init
597  =0003 E95700          005B        jmp entry
598  =
599  =                                                      ;3 variables set by GENSYS
600  =0006 0000                 sysdat          dw      0
601  =  0005                    supervisor      equ     (offset $)
602  =0008                              rw      2
603  =
604  =                                  org     0Ch
605  =000C 06                   dev_ver         db      6       ;development system data version
606  =                                                          ;set in sysdat.fmt
607  =
608  =000D 434F50595249                 db      'COPYRIGHT (C) 1983,'
609          474854202843
610          29203139383320
611          2C
612  =0020 204469474954                 db      ' DIGITAL RESEARCH '
613          414C20524553
614          454152434820
615  =0032 585858582D30                 db      'XXXX-0000-'
616          30303032D
617  =003C 363534333231         serial          db      '654321'
618  =
619  =                          ;====
620  =                          init:
621  =                          ;====
622  =0042 CB                           retf
623  =
624  =                          ;*****************************************************
625  =                          ;*
626  =                          ;*       MEM function table
627  =                          ;*
628  =                          ;*****************************************************
629  =
630  =0043 B700                 functab dw      maxmem_entry    ; 53 - Get Max Memory
631  =0045 E300                         dw      absmax_entry    ; 54 - Get Abs Max Mem
632  =0047 0E01                         dw      cpmalloc_entry  ; 55 - Alloc Mem
633  =0049 3301                         dw      cpmabsa_entry   ; 56 - Alloc Abs Mem
634  =004B 9901                         dw      cpmfree_entry   ; 57 - Free Mem
635  =004D E801                         dw      freeall_entry   ; 58 - Free All Mem
636  =004F 2602                         dw      malloc_entry    ; 129 - Rel Mem Rqst
637  =0051 5F04                         dw      mfree_entry     ; 130 - Memory Free
```

```
                        dw      share_entry     ; share memory
                        dw      maualloc_entry  ; Alloc from MAU
                        dw      maufree_entry   ; Free from MAU
                        dw      mlalloc_entry   ; Alloc from Mem List
                        dw      mlfree_entry    ; Free from Mem List

                ;=====  ================
                entry:  ; MEM entry point
                ;=====  ================
                ;       entry:  CX = memory function number
                ;               DX,BX parameters
                ;       exit:   AX=BX=return code
                ;               CX=error code if BX=0ffffh

                ;       The memory manager is a mutually exclusive code
                ;       section, only one process may be in the memory code
                ;       at one time.  However, a process already in the memory manager
                ;       can call the memory manager ENTRY: point again.
                ;       Once a process gets into the memory code it cannot
                ;       be terminated.

0050 E81100     0071    call m_sync                     ;obtain memory system
0053 B500D1E1           mov ch,0 | shl cx,1             ;and make sure TEMPKEEP is on
0057 8BF1               mov si,cx
0059 2EFF944300         call cs:functab[si]
005E E82400     0092    call m_unsync
0061 8BC3               mov ax,bx                       ;last time out, BX,CX preserved
0063 CB                 retf

                m_sync:
                ;------

0064 535152             push bx | push cx | push dx     ;save entry parameters
0067 BB4106             mov bx,offset mem_spb
006A B91502             mov cx,f_sync                   ;obtain memory system
006D E83F00     00AC    call osif                       ;reetrancy stops here ...
0070 FE064006           inc mem_cnt                     ;keep track how many
                                                        ;times through memory entry
0074 803E40060175 0081  cmp mem_cnt,1 | jne msl
             06 0081
0083 B91702               mov cx,f_no_abort             ;cannot abort while in
0086 E81E00     00AC      call osif                     ;MEM, call NO_ABORT_ENTRY
                msl:                                    ;on 1st time through only
0089 595A5B             pop dx | pop cx | pop bx
008C C3                 ret

                ;       Note: the MEM_CNT variable is set to 1 by TERMINATE in the RTM
                ;       and back to 0 by the TERM_ACT in the dispatcher.

                m_unsync:
                ;--------
0090 FE0E4006           dec mem_cnt                     ;only release when last time
```

```
691
692  =0096 7513          00A8          jnz mu_ret                      ;out of memory system
693  =0098 5351                          push bx ! push cx
694  =009A BA4106                        mov dx,offset mem_spb
695  =009D B91602                        mov cx,f_unsync
696  =00A0 E80900        00AC            call osif                       ;reentrancy starts here ...
697  =00A3 B91802                        mov cx,f_ok_abort
698  =00A6 E80300        00AC            call osif                       ;exit no_abort region
699  =00A9 5958                          pop cx ! pop bx
700  =                           mu_ret:
701  =00AB C3                            ret
702  =
703  =                           ;====           ================
704  =                           osif:           ; O.S. interface
705  =                           ;====           ================
706  =00AC 2EFF1E0900C3                  callf cs:dword ptr .supervisor ! ret
707  =
708  =                           ;======         ================
709  =                           xiosif:         ; XIOS interface
710  =                           ;======         ================
711  =00B2 FF1E2800C3                    callf dword ptr .xiosmod ! ret
712  =
713
```

```
                        eject ! include cpmmem.mem
                    ;*****************************************************
                    ;*                                                   *
                    ;*      MEM Entry Points                             *
                    ;*                                                   *
                    ;*****************************************************

                        ; Format of Memory Control Block used
                        ; in CP/M-86 Memory Calls (53 - 58)
                        ;
                        ;     +-----------+-----------+-----+
                        ; MCB |   Base    |   Length  | ext |
                        ;     +-----------+-----------+-----+
                        ;

0000                    mcb_base        equ     word ptr 0
0002                    mcb_length      equ     word ptr mcb_base + word
0004                    mcb_ext         equ     byte ptr mcb_length + word
0005                    mcblen          equ     mcb_ext + byte


                    ;=============           ======================
                    maxmem_entry:               ; 53 - Get Max Memory
                    ;=============           ======================
                    ;       input:  DX = address of MCB in u_wrkseg
                    ;       output: BX = 0ffffh if failure
                    ;                    0h if success
                    ;               CX = Error Code
                    ;               mcb_ext = 0 if no additional mem
                    ;                         1 if more available

0037 89F2                       mov si,dx
0039 1E268E1E2E00               push ds ! mov ds,u_wrkseg
003F 8B54021F                   mov dx,mcb_length[si] ! pop ds
0043 29C029D8                   sub ax,ax ! mov bx,ax
0047 E83200E304     017C        call getmemory ! jcxz mm_gm
004C B9FFFFC3                       mov bx,0ffffh ! ret
0050 1E268E1E2E00       mm_gm:  push ds ! mov ds,u_wrkseg
0056 895402                     mov mcb_length[si],dx
0059 8904                       mov mcb_base[si],ax
005B C6440401                   mov mcb_ext[si],1
005F 1F29DBC3                   pop ds ! sub bx,bx ! ret


                    ;=============
                    absmax_entry:               ; 54 - Get Abs Max Mem
                    ;=============
                    ; Allocate the largest absolute memory region which
                    ; is less than or equal mcb_length
                    ;       input:  DX = address of MCB in u_wrkseg
                    ;       output: BX = 0ffffh if failure
                    ;                    0h if success
                    ;               CX = Error Code
```

```
                
                =
                =00E3 8BF2                      mov si,dx
                =00E5 1E268E1E2E00              push ds ! mov ds,u_wrkseg
                =00EB 8B04                      mov ax,mcb_base[si]
                =00ED 8B54021F                  mov dx,mcb_length[si] ! pop ds
                =00F1 2BDB                      sub bx,bx
                =00F3 E884000E304    017C       call getmemory ! jcxz am_gm
                =00F6 BBFFFFC3                      mov bx,0ffffh ! ret
                =00FC 1E268E1E2E00      am_gm:  push ds ! mov ds,u_wrkseg
                =0102 895402                    mov mcb_length[si],dx
                =0105 8904                      mov mcb_base[si],ax
                =0107 1F2BDBC3                  pop ds ! sub bx,bx ! ret
                =
                =                       ;===============
                =                       cpmalloc_entry:         ; 55 - Alloc Mem
                =                       ;===============
                =                       ; Allocate a memory region which is equal to mcb_length
                =                       ;
                =                       ;       input:  DX = address of MCB in u_wrkseg
                =                       ;       output: BX = 0ffffh if failure
                =                       ;                    0h if success
                =                       ;               CX = Error Code
                =
                =0108 8BF2                      mov si,dx
                =010D 1E268E1E2E00              push ds ! mov ds,u_wrkseg
                =0113 2BC9895C02                sub ax,ax ! mov bx,mcb_length[si]
                =0118 8BD31F                    mov dx,bx ! pop ds
                =011B E85E00E304    017C        call getmemory ! jcxz ca_gm
                =0120 BBFFFFC3                      mov bx,0ffffh ! ret
                =0124 1E268E1E2E00      ca_gm:  push ds ! mov ds,u_wrkseg
                =012A 895402                    mov mcb_length[si],dx
                =0120 8904                      mov mcb_base[si],ax
                =012F 1F2BDBC3                  pop ds ! sub bx,bx ! ret
                =
                =
                =                       ;==============
                =                       cpmabsa_entry:          ; 56 - Alloc Abs Mem
                =                       ;==============
                =                       ; Allocate an absolut memory region which is
                =                       ; equal to mcb_length.  Note:  For CP/M-86
                =                       ; compatibility, this function must return success
                =                       ; if the memory is already allocated because
                =                       ; the GDI MAX functions allocate memory in MP/M, CCP/M
                =                       ; and not in CP/M.
                =                       ;
                =                       ;       input:  DX = address of MCB in u_wrkseg
                =                       ;       output: AX = 0ffffh if failure
                =                       ;                    0h if success
                =                       ;               CX = Error Code
                =
                =0133 8BF2                      mov si,dx
                =0135 1E268E1E2E00              push ds ! mov ds,u_wrkseg
```

```
                                    mov ax,mcb_base[si]
                                    mov bx,mcb_length[si]
                                    mov dx,bx ! pop ds
                                    ;
                                        ; See if We already own this memory
                                        ; ST -> MCB in U_WRKSEG
                                        ; AX=Base, BX=Length, DX=Length
                                    ;
                                    xor cx,cx
                                    mov di,rlr
                                    add di,p_mem-ms_link
                            caa_n:  mov di,ms_link[di]
                                    cmp di,0 ! je caa_g
                                        cmp ms_start[di],ax ! jne caa_n
                                            cmp ms_length[di],bx ! jbe caa_gm
                                                mov cx,e_no_memory
                                                mov bx,0ffffh ! ret
                                    ;
                                        ; ST -> MCB in U_WRKSEG
                                        ; AX=Base, BX=Length, DX=Length
                            caa_g:  call getmemory ! jcxz caa_gm
                                        mov bx,0ffffh ! ret
                                    ;
                                        ; Successful allocation
                            caa_gm: push ds ! mov ds,u_wrkseg
                                    mov mcb_length[si],dx
                                    mov mcb_base[si],ax
                                    pop ds ! sub bx,bx ! ret
                                    ;
                            getmemory:
                                    ;---------
                                    ;       input:  AX = start
                                    ;               BX = min
                                    ;               DX = max
                                    ;       output: AX = start
                                    ;               DX = length
                                    ;               CX = error code
                                    ;         preserve SI
                                    ;
                                    push si
                                    sub cx,cx ! push cx ! push cx ! push dx ! push bx
                                    push ax ! mov dx,sp
                                    push ds ! mov cx,ss ! mov ds,cx
                                    mov cx,f_malloc ! call osif
                                    pop ds ! pop ax ! pop dx ! pop bx ! pop bx ! pop bx
                                    pop si ! ret
                                    ;
                            ;==============
                            cpmfree_entry:          ; 57 - Free Mem
                            ;==============
                            ; Free memory as specified in MCB
                                    ;       input:  DX = offset of MCB in u_wrkseg
```

```
                        ;               mcb_ext = 0ffh = free all but load mem
                        ;                       else as specified by mcb_base.
                        ;               mcb_base = seg addr of memory segment
                        ;                       to free.  IF in middle of
                        ;                       existing segment then just free
                        ;                       the end of the segment.
                        ;
=0199 1E268E1E2E00              push ds ! mov ds,u_wrkseg
=019F 8BF2                      mov si,dx
=01A1 8A4404                    mov al,mcb_ext[si]
=01A4 8B141F                    mov dx,mcb_base[si] ! pop ds
=01A7 3CFF752A      01D5        cmp al,0ffh ! jne free_memory
=01AB B81E6800          cpmf_root:  mov bx,rlr
=01AF 83C316                    add bx,p_mem-ms_link
=01B2 8B37              cpmf_next:  mov si,ms_link[bx]
=01B4 83FE007505    01BE        cmp si,0 ! jne try_seg
=01B9 2BDB8BC0F3                sub bx,bx ! mov cx,bx ! ret
=01BE F74406010074  01C9 try_seg:   test ms_flags[si],mf_load ! jz free_seg
         04         01C9
=01C5 8BDE8B5E9     01B2            mov bx,si ! jmps cpmf_next
=01C9 8B5402            free_seg:   mov dx,ms_start[si]
=01CC 56E805005E    01D5            push si ! call free_memory ! pop si
=01D1 E3D8          01AB            jcxz cpmf_root
=01D3 EBDD          01B2                jmps cpmf_next
                        ;
                        free_memory:
                        ;-----------
                        ;       input:  DX = start
                        ;       output: BX = 0,0ffffh (success,fail)
                        ;               CX = Error Code
                        ;
=01D5 1E161F                    push ds ! push ss ! pop ds
=01D8 2BC95152                  sub cx,cx ! push cx ! push dx
=01DC 8BD4                      mov dx,sp
=01DE B98200E8C8FF  00AB        mov cx,f_memfree ! call osif
=01E4 5A591F                    pop dx ! pop dx ! pop ds
=01E7 C3                        ret
                        ;
                        ;=============
                        freeall_entry:          ; 58 - Free All Mem
                        ;=============
                        ;  Free all memory except LOAD UDA and LDSTK
                        ;       if a transient program calls this, it must free up the
                        ;       all of the perceived memory.  The UDA and LDSTK is not
                        ;       perceived by CPM compatible programs.  The UDA cannot be
                        ;       freed, since it is part of the process descriptor, until
                        ;       termination in the dispatcher.
                        ;
=01E8 8B3e6800                  mov si,rlr
=01EC 8B4410                    mov ax,p_uda[si]
=01EF 83C616                    add si,p_mem-ms_link
=01F2 8B34              fam_n:  mov si,ms_link[si]
```

```
926
927  =
928  =                                    ; See if anymore memory to free
929  =01F4 83FE007428     0221          cmp si,0 | je fam_e
930  =01F9 8B5C02                       mov bx,ms_start[si]
931  =
932  =                                          ; See if UDA above start
933  =01FC 3BD8771A       021A          cmp bx,ax | ja fam_a              ;AX=UDA Segment
934  =0200 8BCB                         mov cx,bx
935  =0202 034C04                       add cx,ms_length[si]
936  =
937  =                                          ; See if below start+len
938  =0205 3BC87211       021A          cmp cx,ax | jb fam_a
939  =
940  =                                          ; WE HAVE A UDA !!!
941  =                                          ; see if we already trimmed it
942  =
943  =0209 F744064000                     test ms_flags[si],mf_udaonly
944  =020E 75E2           01F2            jnz fam_n
945  =0210 814C064000                     or ms_flags[si],mf_udaonly
946  =
947  =                                                ; trim the memory above the UDA
948  =
949  =0215 051600                        add ax,(lstklen+ulen)/16        ;CX=# paragraphs
950  =0218 8BD8                          mov bx,ax
951  =
952  =                                    ; Free the segment or after the UDA+LOSTK
953  =021A 8BD3           fam_a:         mov dx,bx
954  =021C E886FF         01D5           call free_memory
955  =021F EBC7           01E8           jmps freeall_entry
956  =0221 33DB8BCBC3     fam_e:  xor bx,bx | mov cx,bx | ret
957
```

```
                        eject | include memory.mem
                ;*****************************************************
                ;*
                ;*      MPB entry Points
                ;*
                ;*      Each process descriptor points to a linked
                ;*      list of MDs that describe memory segments that
                ;*      it owns.  The MDs note the starting paragraph
                ;*      and a MAU that the Memory Segment is in.
                ;*
                ;*      Format of MDs on p_mem lists:
                ;*
                ;*      +----+----+----+----+----+----+----+----+----+----+
                ;*      |  link   |  start  |  length |  flags  |   mau   |
                ;*      +----+----+----+----+----+----+----+----+----+----+
                ;*
                ;*      link    link field for p_mem list
                ;*      start   starting paragraph of memory segment
                ;*      length  length in paragraphs of memory segment
                ;*      flags   load,code,and share as in MPB
                ;*      mau     offset of MAU in SYSDAT that segment is
                ;*              allocated from
                ;*
                ;*****************************************************

                ;=============
                malloc_entry:
                ;=============
                ; Allocate Memory - memory is allocated from MAUs.
                ; First we try to allocate memory from MAUs that has
                ; memory segments already allocated by this process.
                ; If that fails, we try to create a new MAU from the
                ; Memory Free List (MFL).  If both fails, an error is
                ; is returned to the user.
                ;
                ;       Input:  DX = MPB in u_wrkseg
                ;       Output: AX = 0 if successful (unused memory)
                ;                  = 1 if successful (shared code)
                ;                  = 0ffffh if failure
                ;               CX = error code
                ;
                ; Format of MPB:
                ;
                ;       +----+----+----+----+----+----+----+----+----+----+
                ;       |  start  |   min   |   max   |  pdadr  |  flags  |
                ;       +----+----+----+----+----+----+----+----+----+----+
                ;
                ;       start - if non-zero, an absolute request
                ;       min -   minimum length that will satisfy the request
                ;       max -   maximum length wanted
                ;               We will try to allocate the maximum
                ;               but will be satisfied with as little as the
```

```
1011
1012  =                         ;               minimum.
1013  -                         ;       pdadr - Process Descriptor to allocate memory to.
1014  =                         ;               Note - PD can not be a current process unless
1015  =                         ;               it is the calling process.
1016  =                         ;               The Calling process must explicitly release
1017  =                         ;               the memory if the PD never becomes a process.
1018  =                         ;               Otherwise, memory is released upon termination.
1019  =                         ;               if pdadr is 0, then calling process allocates
1020  =                         ;               the memory.
1021  =                         ;       flags - 00001h Load     This segment initialized from
1022  =                         ;                               a disk file
1023  =                         ;               00002h Shared   This is a sharable segment
1024  =                         ;               00004h Code     This is a Code segment
1025  =
1026  =                                 ; Get a MD for use in the p_mem list
1027  =
1028  =                                                       ; DX -> MPB (u_wrkseg)
1029  =0226 52E8A5075A    09CF          push dx ! call getmd ! pop dx   ; BX -> MD
1030  =0229 E304          0231          jcxz mall_gmd
1031  =022D B8FFFFFFC3                      mov bx,0ffffh ! ret
1032  =                           mall_gmd:
1033  =                                 ; fill PDADR field in case its zero
1034  =
1035  =0231 06268E062E00                push es ! mov es,u_wrkseg               ;save UDA
1036  =0237 8BF426884506                mov di,dx ! mov ax,es:mpb_pdadr[di]
1037  =023D 3D00007507    0249          cmp ax,0 ! jne mall_gpd
1038  =0242 A16800                          mov ax,rlr
1039  =0245 26894506                        mov es:mpb_pdadr[di],ax
1040  =                           mall_gpd:
1041  =0249 07                          pop es                                ;ES=UDA
1042  =024A 3B0668007448  0298          cmp ax,rlr ! je mall_pdv1
1043  =
1044  =0250 5053                            push ax ! push bx
1045  =0252 5257                            push dx ! push di
1046  =0254 BB5906                          mov bx,offset thrd_spb                ;ok to call sync on
1047  =0257 B91502                          mov cx,f_sync                         ;thread after sync-ing
1048  =025A E84FFE        00AC              call osif                             ;on memory
1049  =025D 5F5A                            pop di ! pop dx
1050  =025F 5B58                            pop bx ! pop ax
1051  =0261 BE7000                          mov si,(offset thrdrt)-p_thread
1052  =
1053  =0264 8B7402              mall_pdnxt: mov si,p_thread[si]
1054  =0267 83FE0074IB    0287              cmp si,0 ! je mall_pdver
1055  =026C 3BF075F4      0264                  cmp si,ax ! jne mall_pdnxt
1056  =0270 3B3668007411  0287                      cmp si,rlr ! je mall_pdver
1057  =0276 B92200                              mov cx,e_active_pd
1058  =0279 53                                  push bx                           ; save MD addr
1059  =027A BF5906                              mov bx,offset thrd_spb
1060  =027D B91602                              mov cx,f_unsync
1061  =0280 E829FE        00AC                  call osif
1062  =0283 5B                                  pop bx                            ; restore MD addr
1063  =0284 E94300        02CF                  jmp mall_err
```

```
1064
1065  =                             mall_pdver:
1066  =0287 5053                            push ax ! push bx
1067  =0289 5257                            push dx ! push di
1068  =028B B95906                          mov bx,offset thrd_spb
1069  =028E B91602                          mov cx,f_unsync
1070  =0291 E813FE          00A8            call osif
1071  =0294 5F5A                            pop di ! pop dx
1072  =0296 5B58                            pop bx ! pop ax
1073  =
1074  =                                         ; verify MIN <= MAX
1075  =
1076  =                             mall_pdv1:
1077  =0298 06268E062E00                    push es ! mov es,u_wrkseg
1078  =029E 268B4D02                        mov cx,es:mpb_min[di]
1079  =02A2 263B4D04                        cmp cx,es:mpb_max[di]
1080  =02A6 7604            02AC           jbe mpb_ok
1081  =02A8 26894D04                            mov es:mpb_max[di],cx
1082  =                             mpb_ok:
1083  =
1084  =                                         ; Make sure total allocation <= MAP
1085  =
1086  =02AC 8BF083C616                      mov si,ax ! add si,(p_mem - ms_link)
1087  =02B1 28C956                          sub cx,cx ! push si
1088  =                             max_chk_nxt:
1089  =02B4 8B34                            mov si,ms_link[si]
1090  =02B6 83FE007405      02C0            cmp si,0 ! je max_chk_done
1091  =02BB 034C04                              add cx,ms_length[si]
1092  =02BE EBF4            02B4            jmps max_chk_nxt
1093  =                             max_chk_done:
1094  =02C0 5EA14C00                        pop si ! mov ax,mmp
1095  =02C4 2BC1                            sub ax,cx
1096  =02C6 3D00007508      02D6            cmp ax,0 ! jne not_zero
1097  =02CA 07                      no_min: pop es
1098  =02CB B90300                  mall_nomem: mov cx,e_no_memory
1099  =02CE E82D07          09FE    mall_err:   call freemd         ; Free the Memory Descriptor
1100  =02D2 B8FFFFCB                        mov bx,0ffffh ! ret ; CX=Error Code set previously
1101  =                             not_zero:
1102  =02D5 263B450272EF    02C8            cmp ax,es:mpb_min[di] ! jb no_min
1103  =02DC 263B45047304    02E6            cmp ax,es:mpb_max[di] ! jae max_ok
1104  =02E2 26894504                            mov es:mpb_max[di],ax
1105  =                             max_ok:
1106  =
1107  =                                         ; initialize local variables
1108  =
1109  =02E6 57BF2A06                        push di ! mov di,offset beststart
1110  =02EA B90B002BC0                      mov cx,11 ! sub ax,ax
1111  =02EF 2E8E060600                      mov es,sysdat
1112  =02F4 F3ABSF                         rep stosw ! pop di
1113  =
1114  =02F7 2BC9                           sub cx,cx
1115  =02F9 07                              pop es
1116  =                                         ; try to allocate memory
```

```
1117
1118  =
1119  =                             mall_next:
1120  =02FA 8B34                            mov si,ms_link[si]
1121  =02FC 83FE007418    031C              cmp si,0 ! je mall_ml
1122  =0301 3346087AF4    02FA                  cmp cx,ms_mau[si] ! je mall_next
1123  =                                                 ; here's a MAU we haven't tried...
1124  =                                                 ; CX=last MAU tried
1125  =                                                 ; DX=MPB adr in u_wrkseg
1126  =                                                 ; BX=New MD
1127  =                                                 ; SI=current MD
1128  =0306 515256                                  push cx ! push dx ! push bx ! push si
1129  =0309 8B5C08                                  mov bx,ms_mau[si]
1130  =030C E83E00        034D                      call mall_try_mau
1131  =030F 5E5A59                                  pop si ! pop bx ! pop dx
1132  =                                                 ; Plus DI=MAU address
1133  =0312 83F900597501 02FA                       cmp cx,0 ! pop cx ! jne mall_next
1134  =                                                 ; exact allocation
1135  =                                                 ; succesful allocation
1136  =0317 E92F00        0348                      jmp mall_linkit
1137  =
1138  =                             mall_ml:                ; We must create a new MAU from MFL
1139  =031A 5352                           push bx ! push dx
1140  =031C 268F1E2E00                     mov ds,u_wrkseg
1141  =0321 BB5400B90803                    mov bx,offset mfl ! mov cx,f_mlalloc
1142  =0327 E8B0FD        00DA              call osif
1143  =032A 2E8E1E0600                      mov ds,sysdat
1144  =                                             ;BX=MAU, (New MD,MPB on stack)
1145  =032F E305          0336              jcxz mall_ml0
1146  =0331 5A5B                              pop dx ! pop bx
1147  =0333 E91300        0348              jmp mall_linkit
1148  =                             mall_ml0:               ; We have a new MAU
1149  =                                             ;place MAU on MAL
1150  =0336 8B367600                        mov si,mal
1151  =033A 8937                            mov m_link[bx],si
1152  =033C 891E7600                        mov mal,bx
1153  =                                             ;allocate memory
1154  =0340 5A57                            pop dx ! push dx
1155  =0342 2BF6                            sub si,si
1156  =0344 E85500        039C              call mall_try_mau
1157  =0347 5A5B                            pop dx ! pop bx
1158  =                                             ;DI=MAU, BX=New MD, DX=MPB in u_wrkseg
1159  =
1160  =                             mall_linkit:
1161  =                                             ;BX -> MD
1162  =                                             ;DX -> MPB in u_wrkseg
1163  =
1164  =0349 833E2A0600750355              cmp beststart,0 ! jne yesmem
1165        03
1166  =
1167  =                             if ccpm
1168  =034E E977FF        02CC                jmp mall_nomem
1169  =                             endif
```

```
1170
1171  =
1172  =                         if mpm
1173  =
1174  =                             ; We couldn't find any memory ...
1175  =                             ; lets take something off the SCL
1176  =
1177  =                         pushf ! cli
1178  =                         mov di,scl
1179  =                         cmp di,0 ! jne take_one_off
1180  =
1181  =                                 ;  No where else to try, giveup.
1182  =                             popf ! jmp mall_nomem
1183  =
1184  =                             ; take first SHARE PD off SCL
1185  =                         take_one_off:
1186  =                         mov ax,p_thread[di]
1187  =                         mov scl,ax
1188  =                         popf
1189  =
1190  =                             ; Free its memory
1191  =                             ;  We can assume only one memory segment
1192  =
1193  =                         push bx ! push dx
1194  =                         mov bx,p_mem[di]
1195  =                         push di ! push ds
1196  =                         push di ! push ms_start[bx]
1197  =                         push ss ! pop ds
1198  =                         mov dx,sp ! mov cx,f_memfree
1199  =                         call osif
1200  =                         pop ax ! pop ax
1201  =                         pop ds ! pop di
1202  =
1203  =                             ;  There should be no error.
1204  =                             ;  Put SHARE PD on PUL
1205  =                         pushf ! cli
1206  =                         mov ax,pul
1207  =                         mov p_link[di],ax
1208  =                         mov pul,di
1209  =                         popf
1210  =
1211  =                             ;  Let's assume we just released a partition
1212  =                             ;  and try the ML Alloc again.
1213  =
1214  =                         pop dx ! pop bx
1215  =                         jmp mall_ml
1216  =                         endif
1217  =
1218  =                             ; We have memory ...
1219  =                         yesmem:
1220  =0355 06268E062E00                push es ! mov es,u_wrkseg
1221  =0359 8BFA                        mov di,dx
1222  =035B A12A06                      mov ax,beststart
```

```
1223
1224  =0360 268905                      mov es:mpb_start[di],ax
1225  =0363 894702                      mov ms_start[bx],ax
1226  =0366 A12C06                      mov ax,bestlen
1227  =0369 26894502                    mov es:mpb_min[di],ax
1228  =036D 26894504                    mov es:mpb_max[di],ax
1229  =0371 894704                      mov ms_length[bx],ax
1230  =0374 26884508                    mov al,es:mpb_flags[di]
1231  =0378 894706                      mov ms_flags[bx],ax
1232  =037B 8B362E06                    mov si,bestsi
1233  =037F 83FE007507    0389          cmp si,0 ! jne mall_10
1234  =0384 26887506                        mov si,es:mpb_pdadr[di]
1235  =0388 83C616                          add si,(p_mem-ms_link)
1236  =038B 89048907            mall_10:mov ax,ms_link[si] ! mov ms_link[bx],ax
1237  =038F 891C                        mov ms_link[si],bx
1238  =0391 8B3E3006                    mov di,bestmau
1239  =0395 897F08                      mov ms_mau[bx],di
1240  =0398 0728C93809C3                pop es ! sub cx,cx ! mov bx,cx ! ret
1241  =
1242  =                           mall_try_mau:
1243  =                           ;------------
1244  =                           ;       input:  BX -> MAU
1245  =                           ;               DX -> MPB in u_wrkseg
1246  =                           ;               SI = MS Root that MAU came from
1247  =                           ;       output: CX = 0, found exact allocation
1248  =                           ;                    else, exact not found
1249  =
1250  =                                   ; save root,mau
1251  =039C 89363406                    mov currsi,si
1252  =03A2 891E3206                    mov currmau,bx
1253  =
1254  =                                   ; copy user's MPB into local MPB
1255  =03A6 521E06                      push dx ! push ds ! push es
1256  =03A9 268E1E2E00                  mov ds,u_wrkseg
1257  =03AE 2E8E060600                  mov es,sysdat
1258  =03B3 8BF28F3606                  mov si,dx ! mov di,offset currmpb
1259  =03B8 B90500F3A5                  mov cx,5 ! rep movsw
1260  =03BD 071F                        pop es ! pop ds
1261  =
1262  =03BF 8A3606                      mov dx,offset currmpb
1263  =03C2 B90903E8E4FC 00AC           mov cx,f_maualloc ! call osif
1264  =
1265  =03C8 5F                          pop di
1266  =03C9 E301          03CC          jcxz chkbest
1267  =03CB C3                              ret
1268  =                           chkbest:
1269  =03CC 0626E0052E00                push es ! mov es,u_wrkseg
1270  =03D2 8E3606                      mov si,offset currmpb
1271  =03D5 8B4404                      mov ax,mpb_max[si]
1272  =03D8 28C9                        sub cx,cx
1273  =03DA 263B4504                    cmp ax,es:mpb_max[di]
1274  =03DE 07                          pop es
1275  =03DF 740F          03F0          jz replacebest
```

```
1276
1277  =03E1 8908008B1E32                  mov cx,3 ! mov bx,currmau
1278        06
1279  =03E8 8BFE                          mov di,si
1280  =03EA 3B052C067625 0415             cmp ax,bestlen ! jbe freeworst
1281  =                           replacebest:
1282  =03F0 8F2A06                        mov di,offset beststart
1283  =03F3 8F1E3006                      mov bx,bestmau
1284  =03F7 E81800        0415            call freeworst
1285  =03FA BE3506                        mov si,offset currmpb
1286  =03FD 8B04A32A06                    mov ax,mpb_start[si] ! mov beststart,ax
1287  =0402 8B4404A32C06                  mov ax,mpb_max[si] ! mov bestlen,ax
1288  =0408 A13206A33006                  mov ax,currmau ! mov bestmau,ax
1289  =040E A13406A32E06                  mov ax,currsi ! mov bestsi,ax
1290  =0414 C3                    savret: ret
1291  =                           freeworst:      ; DI->Start, CX=Return Code, BX=MAU
1292  =0415 833D0074FA    0414            cmp word ptr [di],0 ! je savret
1293  =
1294  =                                           ; free worst memory
1295  =041A 511E                          push cx ! push ds
1296  =041C FF35FF35                      push word ptr [di] ! push word ptr [di]
1297  =0420 53161F                        push bx ! push ss ! pop ds
1298  =0423 8B04B90A03                    mov dx,sp ! mov cx,f_maufree
1299  =0428 E881FC        008C            call osif
1300  =
1301  =                                           ; if MAU empty, free MAU
1302  =042B 83F90075291   0459            cmp cx,0 ! jne mflret
1303  =0430 83FBFF7524    0459                cmp bx,0ffffh ! jne mflret
1304  =0435 585B581F                          pop bx ! pop ax ! pop ax ! pop ds
1305  =
1306  =                                               ; take off MAL
1307  =                                               ; BX = MAU address
1308  =0439 8BC38B7600                    mfl:    mov ax,bx ! mov bx,(offset mal)-m_link
1309  =043E 8BF3391C                      mfl1:   mov si,bx ! mov bx,m_link[si]
1310  =0442 3BD875F8      043E                    cmp bx,ax ! jne mfl1
1311  =0446 50                                        push ax
1312  =0447 8B07390400                                mov ax,m_link[bx] ! mov m_link[si],ax
1313  =044B 5B                                        pop bx
1314  =                                               ; Replace into MFL
1315  =044C B90C05                                    mov cx,f_mlfree
1316  =044F 8BD3B95A00                                mov dx,bx ! mov bx,offset mfl
1317  =0454 E855FC59C3    00AC                        call osif ! pop cx ! ret
1318  =
1319  =0459 585B58                mflret:    pop bx ! pop ax ! pop ax
1320  =045C 1F59C3                            pop ds ! pop cx ! ret
1321  =
1322  =                           ;============
1323  =                           mfree_entry:            ; 130 - Memory Free
1324  =                           ;============
1325  =                           ; Free the memory segment with the given segment addr.
1326  =                           ;       input:  DX = MFPB in u_wrkseg
1327  =                           ;       output: BX = 0 if successful
1328  =                           ;                  = 0ffffh on failure
```

```
1329
1330  =                         ;               CX = error code
1331  =                         ;
1332  =                         ; Memory Free Parameter Block (MFPB)
1333  =                         ;
1334  =                         ;       +----+----+----+----+
1335  =                         ;       |  start  |  pdadr  |
1336  =                         ;       +----+----+----+----+
1337  =                         ;
1338  =                         ;       start - starting paragraph of area to be freed.
1339  =                         ;       pdadr - PD to free memory from.  If 0, then calling
1340  =                         ;               process.  If non-zero, the PD must not be
1341  =                         ;               a current process.
1342  =
1343  =045F 06268E062E00                push es ! mov es,u_wrkseg
1344  =0465 8BF2                        mov si,dx
1345  =0467 26895C02                    mov bx,es:mfpb_pd[si]
1346  =046B 268B14                      mov dx,es:mfpb_start[si]
1347  =046E 07                          pop es
1348  =                                                 ; BX = pdadr
1349  =                                                 ; DX = start paragraph
1350  =046F 83FB007506      047A        cmp bx,0 ! jne mfree_chkpd
1351  =0474 8B1E6800                        mov bx,rlr
1352  =0478 EB3F            0489            jmps mfree_gl
1353  =                           mfree_chkpd:
1354  =
1355  =047A 5352                       push bx ! push dx
1356  =047C 8B5906                      mov bx,offset thrd_spb
1357  =047F B91502                      mov cx,f_sync
1358  =0482 E827FC          00AC        call osif
1359  =0485 5A5B                        pop dx ! pop bx
1360  =
1361  =0487 BE7000                      mov si,(offset thrdrt)-p_thread
1362  =                           mfree_nxtpd:
1363  =048A 8B7402                      mov si,p_thread[si]
1364  =048D 83FE00741A      04AC        cmp si,0 ! je mfree_gotpd
1365  =0492 3BF375F4        048A            cmp si,bx ! jne mfree_nxtpd
1366  =0496 3B3668007410    04AC                cmp si,rlr ! je mfree_gotpd
1367  =049C 8B5906                              mov bx,offset thrd_spb
1368  =049F B91602                              mov cx,f_unsync
1369  =04A2 E807FC          00AC                call osif
1370  =04A5 BBFFFF                              mov bx,0ffffh
1371  =04A8 B92200                              mov cx,e_active_pd
1372  =04AB C3                                  ret
1373  =                           mfree_gotpd:
1374  =04AC 5352                        push bx ! push dx
1375  =04AE 8B5906                      mov bx,offset thrd_spb
1376  =04B1 B91602                      mov cx,f_unsync
1377  =04B4 E8F5FB          00AC        call osif
1378  =04B7 5A5B                        pop dx ! pop bx
1379  =
1380  =                           mfree_gl:
1381  =04B9 8D7716                      lea si,p_mem[bx]
```

```
1382
1383  =                         mfree_next:
1384  =0480 8BDE8B37                    mov bx,si ! mov si,ms_link[bx]
1385  =0484 83FE007422    04E7          cmp si,0 ! je mfree_err
1386  =0489 395402742A    04F6              cmp ms_start[si],dx ! je mfree_it
1387  =048E 77F0          0480                  ja mfree_next
1388  =0490 8B4402                              mov ax,ms_start[si]
1389  =0493 034404                              add ax,ms_length[si]
1390  =0496 3BC276E6      0480                  cmp ax,dx ! jbe mfree_next
1391  =049A 5256                                    push dx ! push si
1392  =049C E81300        04EE                      call mfree_it
1393  =049F 5E5A                                    pop si ! pop dx
1394  =04A1 83F900750A    04B0                      cmp cx,0 ! jne mfree_next
1395  =                         mfree_exit:
1396  =04A6 29DB89C8C3                  sub bx,bx ! mov cx,bx ! ret
1397  =                         mfree_err:
1398  =04AD BBFFFF890300                mov bx,0ffffh ! mov cx,e_no_memory
1399  =04B3 C3                          ret
1400  =
1401  =                         mfree_it:
1402  =                         ;--------
1403  =                         ;       input:  DX = root
1404  =                         ;               SI = MD ([bx])
1405  =                         ;               DX = segment to free
1406  =                         ;       output: AX = 0,0ffffh (success,failure)
1407  =                         ;               CX = Error Code
1408  =
1409  =04B4 535652                      push bx ! push si ! push dx
1410  =                                 ;push MAF structure
1411  =04B7 52FF7402                    push dx ! push ms_start[si]
1412  =04BB FF7408                      push ms_mau[si]
1413  =04BE 8BD4161F                    mov dx,sp ! push ss ! pop ds
1414  =04C2 B90A03E8AAFB  00AC          mov cx,f_maufree ! call osif
1415  =04C8 8BEB                        mov bp,bx                       ;if bp=0,MAU not empty
1416  =04CA 2E8E1F0600                  mov ds,sysdat
1417  =                                     ;pop MAF structure
1418  =04CF 5858                        pop ax ! pop ax ! pop ax
1419  =04D1 5A5E5B                      pop dx ! pop si ! pop bx
1420  =                                     ;DX=segment to free
1421  =                                     ;AX=root
1422  =                                     ;SI=MD ( [BX] )
1423  =04D4 83F900                      cmp cx,0
1424  =04D7 7503          04E7          jne mfree_err
1425  =04D9 3B5402740B    04E7              cmp dx,ms_start[si] ! je mfree_off
1426  =                                         ;decrease length
1427  =04DE 295402                              sub dx,ms_start[si]
1428  =04E1 895404                              mov ms_length[si],dx
1429  =                         mfree_r:
1430  =04E4 EBC1          04E2                  jmps mfree_exit
1431  =                                         ;take off p_mem list
1432  =04E6 88048907            mfree_off:  mov ax,ms_link[si] ! mov ms_link[bx],ax
1433  =                                         ;free MD
1434  =04EA FF740855                        push ms_mau[si] ! push bp
```

```
1435
1436  =0529 8BDE8C404    09F2            mov bx,si ! call freemd
1437  =052E 5D5A                         pop bp ! pop dx
1438  =                                          ;free MAU if empty
1439  =0530 83FD0074AD   C4E2            cmp bp,0 ! je mfree_exit
1440  =                                          ;find it on MAL
1441  =0535 BF7600                           mov di,(offset mal)-m_link
1442  =0538 8BF78B3C          mfree_nmal:    mov si,di ! mov di,m_link[si]
1443  =053C 3BFA75F8     0538                cmp di,dx ! jne mfree_nmal
1444  =                                          ;release from MAL
1445  =0540 8B058904                             mov ax,m_link[di] ! mov m_link[si],ax
1446  =0544 C7050000                             mov m_link[di],0
1447  =                                          ;release to MFL
1448  =0548 BB5A00B90C03                         mov bx,offset mfl ! mov cx,f_mlfree
1449  =054E E958F8       00AC                    jmp osif
1450
```

```
1451
1452  =                         eject | include share.mem
1453  =                       ;*****************************************************
1454  =                       ;*
1455  =                       ;*      Shared Memory Routines
1456  =                       ;*
1457  =                       ;*****************************************************
1458  =
1459  =                       ;============
1460  =                       share_entry:            ; share memory
1461  =                       ;============
1462  =                       ;       input:  DX->SPB in u_wrkseg
1463  =                       ;
1464  =                       ;           +-----+-----+-----+-----+-----+-----+
1465  =                       ;       SPB |    OPD    |    RPD    |   START   |
1466  =                       ;           +-----+-----+-----+-----+-----+-----+
1467  =                       ;
1468  =
1469  =                               ; Obtain new MD
1470  =
1471  =0551 52E87A045A   09CF         push dx | call getmd | pop dx
1472  =0556 E303         0558         jcxz se_c
1473  =0558 E99400       05EF             jmp se_err2
1474  =055B 8BE8                 se_c:   mov bp,bx                       ; BP = New MD
1475  =
1476  =055D 1E268E1E2E00              push ds | mov ds,u_wrkseg
1477  =0563 8BFA8B1D                  mov di,dx | mov bx,spb_opd[di]  ; BX = Owner PD
1478  =0567 8B7502                    mov si,spb_rpd[di]              ; SI = Requestor PD
1479  =056A 8B5504                    mov dx,spb_start[di]            ; DX = start paragraph
1480  =056D 1F                        pop ds
1481  =
1482  =056E 83FB007504   0577         cmp bx,0 | jne se_r
1483  =0573 BB1E6800                      mov bx,rlr
1484  =0577 83FE007504   0580 se_r:   cmp si,0 | jne se_s
1485  =057C BB366800                      mov si,rlr
1486  =0580 3BF3746F      05F3 se_s:   cmp si,bx | je se_ge
1487  =
1488  =0584 8D7F16                       lea di,(p_mem-ms_link)[bx]
1489  =0587 8B39                 se_go:   mov di,ms_link[di]
1490  =0589 83FF00745F    05EC            cmp di,0 | je se_err
1491  =058E 39550275F4    0587                cmp ms_start[di],dx | jne se_go
1492  =
1493  =                                           ; DI = Owner MS to share
1494  =                                           ; SI = Requestor PD
1495  =                                           ; DX = Start
1496  =                                           ; BP = New MD
1497  =0593 8BDD                              mov bx,bp
1498  =0595 895702                            mov ms_start[bx],dx
1499  =0598 8B4504894704                      mov ax,ms_length[di] | mov ms_length[bx],ax
1500  =059E 8B4506894706                      mov ax,ms_flags[di] | mov ms_flags[bx],ax
1501  =05A4 8B4508894708                      mov ax,ms_mau[di] | mov ms_mau[bx],ax
1502  =05AA 8BFB8BD8                          mov di,bx | mov bx,ax
1503  =05AE 1E8E5F02                          push ds | mov ds,m_start[bx]
```

```
1504
1505  =0502 2BDB3A0E0000                    sub bx,bx ! mov cl,.0
1506  =0508 80F900742A    05E6 se_n:        cmp cl,0 ! je se_err1
1507  =050B 83C305FEC9                          add bx,satlen ! dec cl
1508  =050E 391775F2      0508                  cmp sat_start[bx],dx ! jne se_n
1509  =0512 FE4704                              inc sat_nall[bx]
1510  =                                                 ; place new MS on RPD p_mem
1511  =0515 1F8D5C16                            pop ds ! lea bx,p_mem[si]
1512  =0519 8B5508                              mov dx,ms_mau[di]
1513  =051C 8B37                  se_m:         mov si,ms_link[bx]
1514  =051E 83FE007409    0527                  cmp si,0 ! je se_link
1515  =0523 3B54037404    052C                      cmp dx,ms_mau[si] ! je se_link
1516  =0528 8BDEEBF0      051C                          mov bx,si ! jmps se_m
1517  =052C 8935                  se_link:      mov ms_link[di],si
1518  =052E 893F                                mov ms_link[bx],di
1519  =0530 EB0D          053F                  jmps se_ge
1520  =0532 1F83DE80604 09F2 se_err1:       pop ds ! mov bx,bp ! call freemd
1521  =0537 B90300                se_err:   mov cx,e_no_memory
1522  =053A 83CFFFC3              se_err2:  mov bx,0ffffh ! ret
1523  =053E 2BDB8BCBC3            se_ge:  sub bx,bx ! mov cx,bx ! ret
1524
```

```
                        eject ! include mau.mem
                ;*****************************************************
                ;*                                                   *
                ;*      Memory Allocation Unit Routines              *
                ;*                                                   *
                ;*      Memory Allocation Unit                       *
                ;*              A Memory Allocation Unit is a contiguous
                ;*      area of memory that is described by a MD.    *
                ;*      At the beginning of this area is a Suballocation
                ;*      table that describes areas within it that are
                ;*      allocated or free                            *
                ;*                                                   *
                ;*           +----+----+----+----+----+----+----+----+
                ;*      MD   |  Link   |  Start  | Length  |  Plist  |
                ;*           +----+----+----+----+----+----+----+----+
                ;*                                                   *
                ;*      Link - Link field for placing MAUs on Linked
                ;*              lists                                *
                ;*      Start - Segment Address of the area covered  *
                ;*      Length - Length of area                      *
                ;*      Plist - used for a linked list of partitions
                ;*              that are included in this area.      *
                ;*                                                   *
                ;*      Suballocation Table                          *
                ;*              The m_start field is the segment     *
                ;*      address of the suballocation table (offset 0).
                ;*      The first entry of the table is special and  *
                ;*      has the following format.                    *
                ;*                                                   *
                ;*           +----+----+----+----+----+              *
                ;*      SAT0 |ents|   reserved        |              *
                ;*           +----+----+----+----+----+              *
                ;*                                                   *
                ;*              satl_ents - number of SAT entries    *
                ;*                      not including SAT0           *
                ;*      Subsequent entries have the following format *
                ;*                                                   *
                ;*           +----+----+----+----+----+              *
                ;*      SAT  |  Start  |  Length |nall|              *
                ;*           +----+----+----+----+----+              *
                ;*                                                   *
                ;*              start - start address of this contiguous
                ;*                      piece of memory              *
                ;*              length - of this piece of memory     *
                ;*              nall - number of times allocated     *
                ;*                      0 = free  2 = share          *
                ;*                                                   *
                ;*****************************************************

0000            sat_start       equ     word ptr 0
0002            sat_length      equ     word ptr sat_start + word
0004            sat_nall        equ     byte ptr sat_length + word
```

```
                                satlen          equ     sat_nall + byte
                                ;===============
                                maualloc_entry:         ; Alloc from MAU
                                ;===============
                                ; Allocate a piece of memory from an MAU
                                ;
                                ;       input:  BX = address of MAU
                                ;               DX = MPB in u_wrkseg
                                ;               mpb_start = abs addr (0 = any)
                                ;               mpb_min = minimum necessary
                                ;               mpb_max = maximum wanted
                                ;
                                ;       output: BX =     0h if success
                                ;                    0ffffh if failure
                                ;               CX = Error Code
                                ;               mpb_start = start address
                                ;               mpb_min = mcb_max = length
                                
 05F8 06268E062E00              push es ! mov es,u_wrkseg
                                
                                        ; if absolute, see if within MAU
 05FE 8BF2268B0C                mov si,dx ! mov cx,es:mpb_start[si]
 0603 83F900740E    0616        cmp cx,0 ! je maua_rel
 0608 8B4702                        mov ax,m_start[bx]
 060B 3BC87246      0655            cmp cx,ax ! jb maua_err
 060F 034704                        add ax,m_length[bx]
 0612 3BC8733F      0655            cmp cx,ax ! jae maua_err
                                maua_rel:
 0616 8E5F02                    mov ds,m_start[bx]
                                maua_start:
 0619 2BDB8AD3                  sub bx,bx ! mov dl,bl                   ;BX->SAT entry
                                                                        ;DL = SAT number
                                maua_nsat:
 061D FEC283C305                inc dl ! add bx,satlen
 0622 3A1600007720  0655        cmp dl,.0 ! ja maua_err                 ;?out of table entries?
 0628 833F007428    0655        cmp sat_start[bx],0 ! je maua_err       ;?end of used table?
 062D 807F040075EA  0619            cmp sat_nall[bx],0 ! jne maua_nsat  ;?already allocated?
 0633 268B0E                        mov cx,es:mpb_start[si]
 0636 83F90074220   065D            cmp cx,0 ! je maua_nabs             ;?abs req?
 063B 8B07                              mov ax,sat_start[bx]
 063D 3BC8720C      0619                cmp cx,ax ! jb maua_nsat        ;?in this area?
 0641 034702                                add ax,sat_length[bx]
 0644 3BC87305      0619                    cmp cx,ax ! jae maua_nsat
 0648 3B0F7411      065D                        cmp cx,sat_start[bx] ! je maua_nabs ;?exact?
                                                sub cx,sat_start[bx]
 064C 2B0F                                      push si
 064E 56                                        call mau_split                  ;make exact
 064F E83500        0708                        pop si
 0652 5F                                        jmps maua_start
 0653 EBC4          0619
                                maua_err:
 0655 B8FFFFB90300              mov bx,0ffffh ! mov cx,e_no_memory
```

```
1631
1632  =0658 EB45          0642          jmps maua_out
1633  =                           maua_nabs:
1634  =065D 268B4C02                    mov cx,es:mpb_min[si]
1635  =0661 8B4702                      mov ax,sat_length[bx]
1636  =0664 3BC877E5      0610          cmp cx,ax ! ja maua_nsat
1637  =0668 268B4C04                        mov cx,es:mpb_max[si]
1638  =066C 3BC87316      0686              cmp cx,ax ! jae maua_alloc
1639  =0670 807F04007408  0681                  cmp sat_nall[bx],0 ! je maua_splitit
1640  =0676 268B4408                                mov ax,es:mpb_flags[si]
1641  =067A 2502007409    0610                      and ax,mf_share ! jz maua_nsat
1642  =067F EB05          0686                          jmps maua_alloc
1643  =0681 56E8B300      0708  maua_splitit:       push si ! call mau_split ! pop si
1644  =0686 8B4702              maua_alloc:       mov ax,sat_length[bx]
1645  =0689 268944402                           mov es:mpb_min[si],ax
1646  =068D 26894404                            mov es:mpb_max[si],ax
1647  =0691 8B07                                mov ax,sat_start[bx]
1648  =0693 268904                              mov es:mpb_start[si],ax
1649  =0696 FF4704                              inc sat_nall[bx]
1650  =0699 E89400        0730                  call mau_collapse
1651  =069C 2BDB8BC8                            sub bx,bx ! mov cx,bx
1652  =06A0 EB00          06A2                  jmps maua_out
1653  =                           maua_out:
1654  =06A2 2E8E1E0600                  mov ds,sysdat
1655  =06A7 07C3                        pop es ! ret
1656  =
1657  =                           ;==============
1658  =                           maufree_entry:          ; Free from MAU
1659  =                           ;==============
1660  =                           ; free SAT in given MAU
1661  =                           ;       input:  DX = address of MAF in u_wrkseg
1662  =                           ;       output: CX = 0 if successful
1663  =                           ;                  CX = 0ffffh on failure
1664  =                           ;                  if CX = 0,
1665  =                           ;                      BX = 0 if MAU still in use
1666  =                           ;                      BX = 0ffffh if MAU is empty
1667  =                           ;                  if CX = 0ffffh
1668  =                           ;                      BX = 0ffffh
1669  =
1670  =06A9 06268E062E00                push es ! mov es,u_wrkseg
1671  =06AF 8BF2                        mov si,dx
1672  =06B1 268B1C                      mov bx,es:maf_mau[si]
1673  =06B4 8E5F02                      mov ds,m_start[bx]
1674  =
1675  =                                       ; ES:SI -> MAF
1676  =                                       ; DS: 0 -> SAT table
1677  =
1678  =06B7 2BDB8BC8                    sub bx,bx ! mov cx,bx
1679  =06BB 268B5402                    mov dx,es:maf_sat[si]
1680  =
1681  =                                       ; DS:BX -> current SAT
1682  =                                       ; CX = SAT entry number
1683  =                                       ; DX = MAF_SAT start
```

```
1684
1685  =
1686  =                       mauf_nsat:
1687  =06CF 83C30541                  add bx,satlen ! inc cx
1688  =06C3 3B0E00007732 06FB        cmp cx,0 ! ja mauf_err
1689  =06C9 3817                      cmp dx,sat_start[bx]            ; This SAT?
1690  =06CB 75F2            06BF      jne mauf_nsat
1691  =06CD 26395404                  cmp dx,es:maf_start[si]         ; Freeing all of it?
1692  =06D1 7420            06F3      je mauf_free
1693  =06D3 807F0401                  cmp sat_nall[bx],1              ; Is it shared?
1694  =06D7 7402            06D8      je mauf_nsha
1695  =06D9 EB26            0701          jmps mauf_out               ; Can't split it
1696  =                                                               ; since its shared.
1697  =                                                               ; But, no error
1698  =
1699  =06DB 268B4C04          mauf_nsha:      mov cx,es:maf_start[si]     ; Not shared,
1700  =06DF 8B07                              mov ax,sat_start[bx]
1701  =06E1 034702                            add ax,sat_length[bx]
1702  =06E4 3BC17213        06FB              cmp ax,cx ! jb mauf_err     ; Within range?
1703  =
1704  =                                               ; must split this SAT
1705  =                                               ; this can only be done if not shared
1706  =
1707  =06E8 2BCAE81800      0708                  sub cx,dx ! call mau_split
1708  =06EC 2BC9880D9EB03   06F6                  sub cx,cx ! mov bx,cx ! jmps mauf_c
1709  =06F3 FE4F04          mauf_free:      dec sat_nall[bx]
1710  =06F6 E83700E806      0730 mauf_c:    call mau_collapse ! jmps mauf_out
1711  =                     mauf_err:
1712  =06FB BBFFFF890300            mov bx,0ffffh ! mov cx,e_no_memory
1713  =                     mauf_out:
1714  =0701 072E8E1E0600            pop es ! mov ds,sysdat
1715  =0707 C3                      ret
1716  =
1717  =                     mau_split:              ;split SAT into 2
1718  =                     ;---------              -----------------
1719  =                     ; new SAT element starting at split boundary
1720  =                     ; with nall=0, old has same nall.
1721  =                     ;       input:  BX = address of SAT element
1722  =                     ;               CX = length to split at
1723  =                     ;       output: BX = address of original SAT element
1724  =
1725  =070B 5351                    push bx ! push cx
1726  =070A 83C305E8E200    07F2    add bx,satlen ! call mau_insert
1727  =0710 83F90059                cmp cx,0 ! pop cx
1728  =0714 587515          072C    pop bx ! jne maus_err           ;create hole if error
1729  =0717 807F05885702              lea di,satlen[bx] ! mov dx,sat_length[bx]
1730  =071D 2BD1895502                sub dx,cx ! mov sat_length[di],dx
1731  =0722 8B0317                    mov dx,cx ! add dx,sat_start[bx]
1732  =0726 8915C6450400              mov sat_start[di],dx ! mov sat_nall[di],0
1733  =                     maus_err:
1734  =072C 894F02                  mov sat_length[bx],cx
1735  =072F C3                      ret
1736  =
```

```
                        
                        mau_collapse:           ;collapse adjacent unallocated SATs
                        ;------------           -----------------------------------
                        ; collapse adjacent unallocated SATs and holes in table
                        ; if possible
                        ;       return: BX = 0 if not empty
                        ;                  = 0ffffh if empty
                        ;               CX = 0
                        
0736 28D389C8                   sub bx,bx ! mov cx,bx
                        mauc_nsat:
0734 41                         inc cx
0735 3B0E00007469 07A4          cmp cx,.0 ! je mauc_mtchk
073B 83C3058D7F05                 add bx,satlen ! lea di,satlen[bx]
0741 833F00745E   07A4            cmp sat_start[bx],0 ! je mauc_mtchk
0746 833D007459   07A4            cmp sat_start[di],0 ! je mauc_mtchk
074B 807F0400752C 077D            cmp sat_nall[bx],0 ! jne mauc_notfree
                                        ;this SAT is free
0751 8B07034702                       mov ax,sat_start[bx] ! add ax,sat_length[bx]
0756 3B057407     0761                cmp ax,sat_start[di] ! je mauc_fnohole
                                        ;followed by a hole
075A 8B052B07                         mov ax,sat_start[di] ! sub ax,sat_start[bx]
075E 894702                           mov sat_length[bx],ax
                        mauc_fnohole:
                                        ;free SAT., no hole
0761 807D040075CD 0734              cmp sat_nall[di],0 ! jne mauc_nsat
                                        ;followed by free SAT
0767 8B4502                           mov ax,sat_length[di]
076A 014702                           add sat_length[bx],ax
076D 5351                             push bx ! push cx
076F 8BFE85400    07C8                mov bx,di ! call mau_release
0774 5953                             pop cx ! pop bx
0776 83EB0549                         sub bx,satlen ! dec cx
077A E9B7FF       0734                jmp mauc_nsat
                        mauc_notfree:
                                        ;allocated SAT
077D 8B07034702                     mov ax,sat_start[bx] ! add ax,sat_length[bx]
0782 3B0574AE     0734              cmp ax,sat_start[di] ! je mauc_nsat
                                        ;followed by a hole
0786 807D0400740C 0798              cmp sat_nall[di],0 ! je mauc_nfhole
                                        ;next SAT is allocated
078C 5351                             push bx ! push cx
078E 8BFB85F00    07F2                mov di,bx ! call mau_insert
0793 5959                             pop cx ! pop bx
0795 E99CFF       0734                jmp mauc_nsat
                        mauc_nfhole:            ;Alloc SAT, followed by hole
                                        ;next SAT is free
079d 8B052B07                       mov ax,sat_start[di] ! sub ax,sat_start[bx]
079C 2905014502                     sub sat_start[di],ax ! add sat_length[di],ax
07A1 E990FF       0734              jmp mauc_nsat
                        
                        mauc_mtchk:
07A4 BDFFFF                     mov bp,0ffffh
```

```
1790
1791  =0747 33C933D3                xor cx,cx ! xor bx,bx
1792  =                         mauc_mtn:
1793  =074B 41                      inc cx
1794  =074C 80BE00007411 07C3       cmp cx,.0 ! je mauc_out
1795  =0752 83C305                      add bx,satlen
1796  =0755 833F007409   07C3           cmp sat_start[bx],0 ! je mauc_out
1797  =075A 807F046074EB 0748               cmp sat_nall[bx],0 ! je mauc_mtn
1798  =0760 BD0000                              mov bp,0
1799  =                         mauc_out:
1800  =07C3 8BDD33C9C3              mov bx,bp ! xor cx,cx ! ret
1801  =
1802  =                         mau_release:
1803  =                         ;-----------
1804  =                         ;       input: BX = SAT to release
1805  =
1806  =07C8 06BCD88EC0              push es ! mov ax,ds ! mov es,ax
1807  =07CD B80500                  mov ax,satlen
1808  =0700 F6260000                mul byte ptr .0
1809  =0704 5083C828C8              push ax ! mov cx,ax ! sub cx,bx
1810  =0709 8BFB83C305              mov di,bx ! add bx,satlen
1811  =07DE 8BF3F3A4                mov si,bx ! rep movs al,al
1812  =07E2 58C747020000            pop bx ! mov sat_length[bx],0
1813  =07E8 C7070000                mov sat_start[bx],0
1814  =07EC C6470400                mov sat_nall[bx],0
1815  =07F0 07C3                    pop es ! ret
1816  =
1817  =                         mau_insert:
1818  =                         ;-----------
1819  =                         ;       input:  BX = SAT to place new SAT in front
1820  =                         ;       output: CX = 0 if successful
1821  =
1822  =07F2 B80500F62600            mov ax,satlen ! mul byte ptr .0
1823        09
1824  =07F9 8BF0A9FFFF              mov si,ax ! mov cx,0ffffh
1825  =07FE 833C0075lA   0810       cmp sat_start[si],0 ! jne maui_r
1826  =0803 8BCE2BC8                    mov cx,si ! sub cx,bx
1827  =0807 4E8D7C05                    dec si ! lea di,satlen[si]
1828  =080B 061E07                      push es ! push ds ! pop es
1829  =080E FDF3A4FC                    std ! rep movsb ! cld
1830  =0812 0728C9                      pop es ! sub cx,cx
1831  =0815 890F                        mov sat_start[bx],cx
1832  =0817 894F02                      mov sat_length[bx],cx
1833  =081A 884F04                      mov sat_nall[bx],cl
1834  =081D C3                  maui_r: ret
1835
```

```
1836
1837  =                         eject 1 include ml.mem
1838  =                       ;*****************************************************
1839  =                       ;*
1840  =                       ;*      Memory List Functions
1841  =                       ;*
1842  =                       ;*****************************************************
1843  =
1844  =                       ;==============
1845  =                       mlalloc_entry:          ; Alloc from Mem List
1846  =                       ;==============
1847  =                       ; Create a MAU from a Memory List.  Memory List is a
1848  =                       ; linked list of MDs in memory order (low first).  It
1849  =                       ; is assumed that units in the Memory List are paragraphs
1850  =                       ;       input:  DX = address of MPB in u_wrkseg
1851  =                       ;               BX = ML root in SYSDAT
1852  =                       ;       output: BX = address of MAU
1853  =                       ;                  = 0ffffh if error
1854  =                       ;               CX = Error Code
1855  =
1856  =0815 8BF2                      mov si,dx
1857  =0820 29C089C88AFF              sub ax,ax ! mov cx,ax ! mov dx,0ffffh
1858        FF
1859  =                                     ; AX = MD w/best score
1860  =                                     ; DX = best score
1861  =                                     ; CX = # partitions used
1862  =                                     ; SI -> MPB in u_wrkseg
1863  =0827 5051                      push ax ! push cx
1864  =0829 833F007420    084E mla_nmd:cmp m_link[bx],0 ! je mla_found
1865  =082E 535256                        push bx ! push dx ! push si
1866  =0831 E82F00        0863            call mla_value
1867  =0834 5E5A58                        pop si ! pop dx ! pop bx
1868  =0837 3BC473DF      0841            cmp cx,dx ! jae mla_next
1869  =083B 8BD189C8                          mov dx,cx ! pop cx ! mov cx,ax
1870  =083F 5889C3                            pop ax ! mov ax,bx
1871  =0843 5051                              push ax ! push cx
1872  =0845 83FA007404    084E                cmp dx,0 ! je mla_found
1873  =084A 8B1F                mla_next:   mov bx,m_link[bx]
1874  =084C EBDB          0829              jmps mla_nmd
1875  =                         mla_found:
1876  =084E 5958                      pop cx ! pop ax
1877  =0850 3D000007407   0859        cmp ax,0 ! je mla_err
1878  =0855 89D889C1                      mov bx,ax ! mov ax,cx
1879  =0859 E99F00        08FB            jmp mla_makemau
1880  =                         mla_err:
1881  =085C BBFFFF890300              mov bx,0ffffh ! mov cx,e_no_memory
1882  =0862 C3                  mla_out:ret
1883  =
1884  =                         mla_value:
1885  =                         ;---------
1886  =                         ;       input:  BX = ML element Root
1887  =                         ;               SI = MPB in u_wrkseg
1888  =                         ;       output: CX = value (0=perfect,0ffffh=no fit)
```

```
1889
1890  =                         ;               AX = # of elements
1891  =
1892  =0863 0626BF062E00               push es ! mov es,u_wrkseg
1893  =0867 29C08F0000                 sub ax,ax ! mov dx,0
1894  =                                        ; AX = # of partitions
1895  =                                        ; DX = total length so far
1896  =086E 8B3F                       mov di,m_link[bx]
1897  =0870 83FF007450    08C5         cmp di,0 ! je mlav_err
1898  =                                        ; we have a list
1899  =0875 26833C007416 0891              cmp es:mpb_start[si],0 ! je mlav_next
1900  =                                            ;an absolute request
1901  =087D 8B400283C10A                   mov cx,m_start[di] ! add cx,(2*satlen)
1902  =0881 26390C723F    08C5             cmp es:mpb_start[si],cx ! jb mlav_err
1903  =                                            ;it starts after our start
1904  =0886 03400483E90A                       add cx,m_length[di] ! sub cx,(2*satlen)
1905  =088C 26390C7334    08C5                 cmp es:mpb_start[si],cx ! jae mlav_err
1906  =
1907  =                                            ;this list satisfies ABS requirement
1908  =0891 26834C04          mlav_next:   mov cx,es:mpb_max[si]
1909  =0895 3BCA7233      08CC             cmp cx,dx ! jb mlav_chk
1910  =0899 3D0000740F    08AD mlav_add:       cmp ax,0 ! je mlav_adl
1911  =089E 50                                 push ax
1912  =089F 8B4702                             mov ax,m_start[bx]
1913  =08A2 034704                             add ax,m_length[bx]
1914  =08A5 3B4502587402 08AD                  cmp ax,m_start[di] ! pop ax ! je mlav_adl
1915  =08AB EB0D          08BA                 jmps mlav_chkmin
1916  =08AD 40035504          mlav_adl:        inc ax ! add dx,m_length[di]
1917  =08B1 8BDF8B3F                           mov bx,di ! mov di,m_link[bx]
1918  =08B5 83FF0075D7    0891                 cmp di,0 ! jne mlav_next
1919  =08BA 268B4C02          mlav_chkmin:         mov cx,es:mpb_min[si]
1920  =08BE 83C10A                             add cx,(2*satlen)
1921  =08C1 3BCA760E      08D3                 cmp cx,dx ! jbe mlav_calc
1922  =08C5 B9FFFF2BC0        mlav_err:            mov cx,0ffffh ! sub ax,ax
1923  =08CA 07C3                               pop es ! ret
1924  =08CC 83C10A            mlav_chk:    add cx,(2*satlen)
1925  =08CF 3BCA77C6      0899             cmp cx,dx !  ja mlav_add
1926  =
1927  =                         mlav_calc:       ; HERE IS WHERE WE GIVE THIS SET A VALUE.
1928  =                                          ; IT CAN SATIFY THE REQUEST.
1929  =
1930  =                                          ; ALGORITHM:
1931  =                                          ;       #partitions = #partitions + 1 if next
1932  =                                          ;                 to contiguous memory
1933  =                                          ;       value = (abs(max-length) SHR 4) +
1934  =                                          ;                 ((#partitions-1) SHL 2)
1935  =                                          ; idea is that 1K = 1 partition to avoid
1936  =                                          ; gobbling all of our small partitions.
1937  =                                          ; we deal with 256 byte memory units so we
1938  =                                          ; won't overflow a word.
1939  =
1940  =                                          ; AX = # partitions
1941  =                                          ; DX = memory size
```

```
1942
1943  =
1944  =08D3 5048                        push ax ! dec ax
1945  =08D5 B102                        mov cl,2
1946  =08D7 D3E0                        shl ax,cl
1947  =08D9 268B4C04                    mov cx,es:mpb_max[si]
1948  =08DD 83C108                      add cx,(2*satlen)
1949  =
1950  =                           ;         if MAX memory is FFFF, SAT adjustment
1951  =                           ;         wraps around so force FFFF.
1952  =
1953  =08E0 83F90A7303    08E8           cmp cx,(2*satlen) ! jae mlav_added
1954  =08E5 B9FFFF                          mov cx,0FFFFH
1955  =                           mlav_added:
1956  =
1957  =08E8 3BCA7602      08EE           cmp cx,dx ! jbe mlav_ab
1958  =08EC 87D1                          xchg dx,cx
1959  =08EE 2BD1B104D3EA          mlav_ab:  sub dx,cx ! mov cl,4 ! shr dx,cl
1960  =08F4 03D08BCA                      add dx,ax ! mov cx,dx
1961  =08F8 5807C3                        pop ax ! pop es ! ret
1962  =
1963  =                           mla_makemau:
1964  =                           ;-----------
1965  =                           ;       input:  BX = address of ML element Root
1966  =                           ;               AX = number of elements
1967  =                           ;       output: BX = MAU address
1968  =
1969  =08FB 3D01007F12    0912           cmp ax,1 ! jg mlam_getmd
1970  =0900 8B378B0C                      mov si,m_link[bx] ! mov cx,m_link[si]
1971  =0904 890F8BDE                      mov m_link[bx],cx ! mov bx,si
1972  =0908 2BC9894F06                    sub cx,cx ! mov m_plist[bx],cx
1973  =090D 890F                          mov m_link[bx],cx
1974  =090F E94000        0952           jmp mla_initmau
1975  =                           mlam_getmd:
1976  =0912 5053                        push ax ! push bx
1977  =0914 E8B300        09CF           call getmd
1978  =0917 8BFB5B58                     mov di,bx ! pop bx ! pop ax
1979  =091A E303          0920           jcxz mlam_gotmd
1980  =091C 2BDBC3                          sub bx,bx ! ret
1981  =                           mlam_gotmd:
1982  =0920 C7050000C745                mov m_link[di],0 ! mov m_length[di],0
1983        040000
1984  =0929 8B378B7506                    mov si,m_link[bx] ! mov m_plist[di],si
1985  =092E 894C02894D02                  mov cx,m_start[si] ! mov m_start[di],cx
1986  =                           mlam_next:
1987  =0934 894C04014D04                  mov cx,m_length[si] ! add m_length[di],cx
1988  =093A 483B00007404  0944           dec ax ! cmp ax,0 ! je mlam_nomore
1989  =0940 8B34EBF0      0934               mov si,m_link[si] ! jmps mlam_next
1990  =                           mlam_nomore:
1991  =0944 53                          push bx
1992  =0945 8B1C                        mov bx,m_link[si]
1993  =0947 C7040000                    mov m_link[si],0
1994  =094B 5E891C                      pop si ! mov m_link[si],bx
```

```
1995
1996  =094E 89DF2BC9                mov bx,di ! sub cx,cx ! ;jmp mla_initmau
1997  =
1998  =                         mla_initmau:
1999  =                         ;----------
2000  =                         ;       input:  BX = address of MAU
2001  =                         ;       output: DX = address of MAU
2002  =                         ;               CX = 0
2003  =
2004  =0952 8B57028B4F04              mov dx,m_start[bx] ! mov cx,m_length[bx]
2005  =0958 51068EC2268C0            push cx ! push es ! mov es,dx ! sub ax,ax
2006  =095E 8BF8B90500              mov di,ax ! mov cx,((32*satlen)/2)
2007  =0963 F3AB07                  rep stos ax ! pop es
2008  =0966 598BDA                  pop cx ! mov ds,dx
2009  =0969 C6050001F               mov byte ptr .0,31
2010  =096C 83E90A83C20A            sub cx,(2*satlen) ! add dx,(2*satlen)
2011  =0974 8B05008914              mov si,satlen ! mov sat_start[si],dx
2012  =0979 894C02C64404            mov sat_length[si],cx ! mov sat_null[si],0
2013        00
2014  =0980 2E8E1F0600              mov ds,sysdat
2015  =0985 2BC9C3                  sub cx,cx ! ret
2016  =
2017  =                         ;============
2018  =                         mltree_entry:           ; Free from Mem List
2019  =                         ;============
2020  =                         ;       input:  DX = MAU address
2021  =                         ;               BX = ML Root
2022  =
2023  =0988 8BF2                  mov si,dx
2024  =098A 837C0600741D 09A9        cmp m_plist[si],0 ! je mlf_freeone
2025  =0990 8BFE                      mov di,si
2026  =0992 8B7406                    mov si,m_plist[si]
2027  =0995 83FE00740C    09A6 mlf_nmd:   cmp si,0 ! je mlf_endfree
2028  =099A FF345357                      push m_link[si] ! push bx ! push di
2029  =099E E81400        0995            call ml_insert
2030  =09A1 5F5B5EEBEF    0995            pop di ! pop bx ! pop si ! jmps mlf_nmd
2031  =                         mlf_endfree:
2032  =09A6 8BD3E84700    09F2            mov bx,di ! call freemd
2033  =09AB EB03          09B0            jmps mlf_exit
2034  =                         mlf_freeone:
2035  =09AD E80500        0985            call ml_insert
2036  =                         mlf_exit:
2037  =09B0 2BDB8BCBC3              sub bx,bx ! mov cx,bx ! ret
2038  =
2039  =                         ml_insert:
2040  =                         ;---------
2041  =                         ;       input:  SI = address of MD to insert
2042  =                         ;               BX = ML Root
2043  =
2044  =0985 8B3F                  mov di,m_link[bx]
2045  =0987 83FF00740C    09C8        cmp di,0 ! je mli_ins
2046  =098C 8B4502                    mov ax,m_start[di]
2047  =098F 3B44027704    09C8        cmp ax,m_start[si] ! ja mli_ins
```

```
2048
2049  =09C4 89FBBED      0995              mov bx,di ! jmps ml_insert
2050  =09C6 88078937     ml_ins:mov ax,m_link[bx] ! mov m_link[bx],si
2051  =09CC 8904                    mov m_link[si],ax
2052  =09CE C3                      ret
2053
```

```
2054
2055  =                         eject ! include util.mem
2056  =                       ;*****************************************************
2057  =                       ;*
2058  =                       ;*       MEM Common Functions
2059  =                       ;*
2060  =                       ;*****************************************************
2061  =
2062  =
2063  =                       getmd:          ; get MD from MDUL
2064  =                       ;-----          ------------------
2065  =                       ;       output: bX = MD address
2066  =                       ;                    0 if none found
2067  =                       ;               CX = Error Code
2068  =
2069  =09CF 9CFA                      pushf ! cli
2070  =0901 891200881E58              mov cx,e_no_umd ! mov bx,mdul
2071        00
2072  =0905 83FB007413   09F0         cmp bx,0 ! je gmd_ret
2073  =0909 33C9                          xor cx,cx
2074  =090B 8B3739365800                  mov si,m_link[bx] ! mov mdul,si
2075  =090F 890F894F02                    mov m_link[bx],cx ! mov m_start[bx],cx
2076  =0914 894F04894F06                  mov m_length[bx],cx ! mov m_plist[bx],cx
2077  =09F0 9DC3              gmd_ret:popf ! ret
2078  =
2079  =
2080  =                       freemd:         ; put MD on MDUL
2081  =                       ;------         ----------------
2082  =                       ;       input:  bX = MD address
2083  =
2084  =09F2 9CFA                      pushf ! cli
2085  =09F4 8B365800891E              mov si,mdul ! mov mdul,bx
2086        5800
2087  =09FC 893709C3                  mov m_link[bx],si ! popf ! ret
2088
```

```
2089
2090  =                       eject ! include patch.cod
2091  =                     ;*****************************************************
2092  =                     ;*
2093  =                     ;*      PATCH AREA  --  128 bytes long
2094  =                     ;*
2095  =                     ;*****************************************************
2096  =
2097  =                     patch:
2098  =0A00 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;00-0f
2099  =0A06 909090909090            nop ! nop ! nop ! nop ! nop ! nop
2100  =0A0C 90909090                nop ! nop ! nop ! nop
2101  =
2102  =0A10 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;10-1f
2103  =0A16 909090909090            nop ! nop ! nop ! nop ! nop ! nop
2104  =0A1C 90909090                nop ! nop ! nop ! nop
2105  =
2106  =0A20 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;20-2f
2107  =0A26 909090909090            nop ! nop ! nop ! nop ! nop ! nop
2108  =0A2C 90909090                nop ! nop ! nop ! nop
2109  =
2110  =0A30 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;30-3f
2111  =0A36 909090909090            nop ! nop ! nop ! nop ! nop ! nop
2112  =0A3C 90909090                nop ! nop ! nop ! nop
2113  =
2114  =0A40 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;40-4f
2115  =0A46 909090909090            nop ! nop ! nop ! nop ! nop ! nop
2116  =0A4C 90909090                nop ! nop ! nop ! nop
2117  =
2118  =0A50 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;50-5f
2119  =0A56 909090909090            nop ! nop ! nop ! nop ! nop ! nop
2120  =0A5C 90909090                nop ! nop ! nop ! nop
2121  =
2122  =0A60 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;60-6f
2123  =0A66 909090909090            nop ! nop ! nop ! nop ! nop ! nop
2124  =0A6C 90909090                nop ! nop ! nop ! nop
2125  =
2126  =0A70 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;70-7f
2127  =0A76 909090909090            nop ! nop ! nop ! nop ! nop ! nop
2128  =0A7C 90909090                nop ! nop ! nop ! nop
2129  =
2130
```

```
2131
2132  =                         eject ! include udu.fmt
2133  =                       ;*********************************************************
2134  =                       ;*
2135  =                       ;*      User Data Area - The User Data Area is an
2136  =                       ;*      extension of the process descriptor but it
2137  =                       ;*      travels with the user.  It contains info
2138  =                       ;*      that is needed only while in context.
2139  =                       ;*
2140  =                       ;*      While in the operating system, The Extra
2141  =                       ;*      Segment register points to the beginning
2142  =                       ;*      of the User Data Area.
2143  =                       ;*
2144  =                       ;*********************************************************
2145  =
2146  =  0060                 ud_insys        equ     60h
2147  =
2148  =                               eseg
2149  =                               org     0
2150  =
2151  =0000                   u_dparam        rw      1       ; arg to dispatch
2152  =
2153  =                       ;       this area overlays part of BDOS
2154  =0002                   u_dma_ofst      rw      1       ; BDOS dma offset
2155  =0004                   u_dma_seg       rw      1       ; bDOS dma segment
2156  =0006                   u_func          rb      1       ; actual function number
2157  =0007                   u_searchl       rb      1       ; BDOS search length
2158  =0008                   u_searcha       rw      1       ; BDOS search FCb offset
2159  =000A                   u_searchabase   rw      1       ; BDOS search user's segment
2160  =000C                   u_dcnt          rw      1       ; BDOS directory count
2161  =000E                   u_dblk          rw      1       ; BDOS directory block #
2162  =0010                   u_error_mode    rb      1       ; BDOS error mode
2163  =0011                   u_mult_cnt      rb      1       ; BDOS multi-sector count
2164  =0012                   u_df_password   rb      8       ; BDOS default password
2165  =001A                   u_pd_cnt        rb      1       ; BDOS process count
2166  =  0019                 ud_ovl_len      equ     (offset $)-(offset u_dma_ofst)
2167  =                       ;       end of overlay area
2168  =
2169  =
2170  =001B                   u_in_int        rb      1
2171  =001C                   u_sp            rw      1       ; save register area
2172  =001E                   u_ss            rw      1
2173  =0020                   u_ax            rw      1
2174  =0022                   u_bx            rw      1
2175  =0024                   u_cx            rw      1
2176  =0026                   u_dx            rw      1
2177  =0028                   u_di            rw      1
2178  =002A                   u_si            rw      1
2179  =002C                   u_bp            rw      1
2180  =002E                   u_wrkseg        rw      1       ; curr seg addr of buf
2181  =0030                   u_retseg        rw      1       ; usr ES return
2182  =0032                   u_ds_sav        rw      1       ;\
2183  =0034                   u_stack_sp      rw      1       ; usr stack segment
```

```
2184
2185  =0036                u_stack_ss      rw      1       ; usr stack pointer
2186  =0038                u_ivectors      rw      4       ; save int 0,1
2187  =0040                u_unused        rw      2       ;
2188  =0044                u_ivectors2     rw      4       ; save int 3,4
2189  =0048                u_es_sav        rw      1       ; > Used during interrupts
2190  =004A                u_flag_sav      rw      1       ;/
2191  =
2192  =0050                u_initcs        rw      1
2193  =0052                u_initds        rw      1
2194  =0054                u_inites        rw      1
2195  =0056                u_initss        rw      1
2196  =0058                u_os_ip         rw      1       ; O.S. vec save
2197  =005A                u_os_cs         rw      1
2198  =005C                u_debug_ip      rw      1       ; RTS,Debug Vector Save
2199  =005E                u_debug_cs      rw      1
2200  =0060                u_insys         rb      1       ; # times through user_entry
2201  =0061                u_stat_sav      rb      1       ; used during interrupts
2202  =0062                u_conccb        rw      1       ; default console's CCB addr
2203  =0064                u_lstccb        rw      1       ; default list devices CCB addr
2204  =0066                u_delim         rb      1       ; delimiter for user function 9
2205  =
2206  =                    ;       org     ulen
2207  =                    ;u_8087         rw      47      ; 8087 save area
2208  =                    ;                               ; see dispatcher, loader and terminate
2209
```

```
2210
2211  =                         eject ! include lstk.fmt
2212  =                       ;***************************************************
2213  =                       ;*
2214  =                       ;*       Load Stack Format
2215  =                       ;*
2216  =                       ;***************************************************
2217  =
2218  =  0060                 lstklen         equ     96 * byte
2219  =
2220  =                               DSEG    ,   '
2221  =                               org     lstklen - 10
2222  =
2223  =0056                   ls_sp           rw      0
2224  =0056                   ls_offset       rw      1
2225  =0058                   ls_cseg         rw      1
2226  =005A                   ls_flags        rw      1
2227  =005C                   ls_roffset      rw      1
2228  =005E                   ls_rcseg        rw      1
2229
```

```
2230
2231  =                         eject ! include sysdat.dat
2232  =                       ;*****************************************************
2233  =                       ;*
2234  =                       ;*      System Data Area
2235  =                       ;*
2236  =                       ;*****************************************************
2237  =
2238  =                               CSEG
2239  =                               org     0ch
2240  =                       ;dev_ver
2241  =0000 06                               db      6       ;development system data version
2242  =                                                       ;SYSDAT.CON has 16 byte code segment
2243  =
2244  =                               DSEG
2245  =                               org     0
2246  =
2247  =                               ;
2248  =                               ;This data is initialized by GENCCPM
2249  =                               ;
2250  =
2251  =                               ;Module Table - contains the FAR CALL addresses
2252  =                               ;       of each module for their initialization
2253  =                               ;       and entry routines.
2254  =                               ;
2255  =                               ;       +---+---+---+---+---+---+---+---+
2256  =                               ;       |     entry     |   initialize  |
2257  =                               ;       +---+---+---+---+---+---+---+---+
2258  =                               ;
2259  =                               ;               entry   init
2260  =                               ;               -----   ----
2261  =
2262  =  0000                 module_table    equ     dword ptr (offset $)
2263  =  0000                 supmod          equ     (offset $)
2264  =0000 030000000000              dw      3,0,    0,0     ;SUP
2265        0000
2266  =
2267  =  0008                 rtnmod          equ     (offset $)
2268  =0008 030000000000              dw      3,0,    0,0     ;RTM
2269        0000
2270  =
2271  =  0010                 memmod          equ     (offset $)
2272  =0010 030000000000              dw      3,0,    0,0     ;MEM
2273        0000
2274  =
2275  =  0018                 ciomod          equ     (offset $)
2276  =0018 030000000000              dw      3,0,    0,0     ;CIO
2277        0000
2278  =
2279  =  0020                 bdosmod         equ     (offset $)
2280  =0020 030000000000              dw      3,0,    0,0     ;BDOS
2281        0000
2282  =
```

```
2283
2284  =  0028                  xiosmod         equ     (offset $)
2285  =0028 0300000000000C                     dw      0C03H,0,        0C00H,0 ;XIOS
2286        0000
2287  =
2288  =  0030                  netmod          equ     (offset $)
2289  =0030 030000000000                       dw      3,0,    0,0     ;NET
2290        0000
2291  =
2292  =  0036                  dispatcher      equ     (offset $)
2293  =0036 00000000                           dw      0,0             ;far dispatcher (does IRET)
2294  =
2295  =  003C                  rtm_pdisp       equ     (offset $)
2296  =003C 00000000                           dw      0,0             ;far dispatcher (does RETF)
2297  =
2298  =                                ; location in memory of MP/M-86 or CCP/M-86
2299  =
2300  =0040 0810               osseg           dw      1008h   ;1st parag. of MP/M-86 or CCP/M
2301  =0042 0000               rspseg          dw      0       ;segment of first RSP
2302  =0044 0000               endseg          dw      0       ;1st parag. outside of MP/M or CCP/M
2303  =
2304  =0046 3F                 module_map      db      03fh    ;bit map of modules that exist
2305  =                                                        ; in this system.  low order bit
2306  =                                                        ; corresponds to 1st module in
2307  =                                                        ; module table.  If bit is on,then
2308  =                                                        ; module exists.
2309  =
2310  =                                ; some answers to GENCCPM questions
2311  =
2312  =0047 04                 ncns            db      4       ;# system console devices
2313  =0048 01                 nlst            db      1       ;# system list devices
2314  =0049 05                 nccb            db      5       ;# character control blocks
2315  =004A 20                 nflags          db      20h     ;# flags
2316  =004B 01                 srchdisk        db      1       ;system disk
2317  =004C 0040               mmp             dw      04000h  ;Max Memory per process
2318  =004E 00                 nslaves         db      0       ;Number of network requestors
2319  =004F 00                 dayfile         db      0       ;if 0ffh,display command info
2320  =0050 01                 tempdisk        db      1       ;Temporary disk
2321  =0051 3C                 tickspersec     db      60      ;Number of ticks per second
2322  =
2323  =                                ; data lists created by GENCCPM
2324  =
2325  =0052 0000               free_root       dw      0       ;locked unused list
2326  =0054 0000               ccb             dw      0       ;addr. Console Ctrl Blk Table
2327  =0056 0000               flags           dw      0       ;addr. Flag Table
2328  =0058 2000               mdul            dw      020h    ;Mem descr. Unused List
2329  =005A 0000               mfl             dw      0       ;Memory Free List
2330  =005C 1400               pul             dw      014h    ;Proc. descr. Unused List
2331  =005E 2000               qul             dw      020h    ;QCB Unused List
2332  =                                                        ;MAU for queue buffer info
2333  =0060 0000               qmau            dw      0       ;link
2334  =0062 0000                               dw      0       ;start segment
2335  =0064 0004                               dw      400h    ;length
```

```
2355
2356  =0066 0000                        dw      0       ;plist
2357  =
2358  =                         ;
2359  =                         ;This data is initialized at Assembly time
2360  =                         ;
2361  =
2362  =0068 7704        rlr             dw      initpd  ;Ready List Root
2363  =006A 0000        ulr             dw      0       ;Delay List Root
2364  =006C 0000        drl             dw      0       ;Dispatcher Ready List
2365  =006E 0000        plr             dw      0       ;Poll List Root
2366  =0070 0000        scl             dw      0       ;Shared Code List
2367  =0072 7704        thrdrt          dw      initpd  ;Process Thread Root
2368  =0074 9401        qlr             dw      mxloadqd;Queue List Root
2369  =0076 0000        mal             dw      0       ;Memory Alloc List
2370  =
2371  =                         ; Version Information
2372  =
2373  =0078 0000        version         dw      unknown ;addr. version str in SUP code segment
2374  =                                         ;set by GENCCPM if CCP/M
2375  =                 if mpm
2376  =                 bvernum         dw      01130h  ;MPM-86 w/BDOS v3.0
2377  =                 osvernum        dw      01121h  ;MPM-86 V2.1
2378  =                 endif
2379  =
2380  =                 if ccpm
2381  =007A 3114        bvernum         dw      01431h  ;CCP/M w/BDOS 3.1
2382  =007C 2014        osvernum        dw      01420h  ;CCP/M V2.0
2383  =                 endif
2384  =
2385  =                         ; Time of Day Structure
2386  =
2387  =007E             tod             rw      0
2388  =007E 7E06        tod_day         dw      067EH   ;day since 1/1/78 (09 Jul 82)
2389  =0080 12          tod_hr          db      12h     ;hour of day
2390  =0081 00          tod_min         db      00h     ;minute of hour
2391  =0082 00          tod_sec         db      00h     ;second of minute
2392  =
2393  =                         ; info from XIOS
2394  =
2395  =0083 00          ncondev         db      0               ;# console devs in XIOS
2396  =0084 00          nlstdev         db      0               ;# character devs in XIOS
2397  =0085 00          nciodev         db      0               ;# character i/o devices
2398  =                                                         ; supported by XIOS.
2399  =0086 0000        lcb             dw      0               ; list control block address
2400  =0088 0000        openvec         dw      0               ; open file vector
2401  =008A 20          lock_max        db      20h             ; Max Locked Records/process
2402  =008B 20          open_max        db      20h             ; Max Open Files/process
2403  =008C 0000        owner8087       dw      0               ; no one owns it initially
2404  =008E             rsv             rw      1               ; RESERVED
2405  =0090 FF          cmod            db      0ffh            ; BDOS Compatibility
2406  =0091 00          ndp8087         db      false           ; true 8087 exits
2407  =                                                         ; (Numeric Data Processor)
```

```
2389
2390  =0092 00000000          err_intercept   dw      0,0             ; BDOS does a callf here
2391  =                                                               ; to print error msgs,
2392  =                                                               ; if second word is <> 0
2393  =0096 4106              slr             dw      offset mem_spb  ; Sync List Root
2394  =0098 000000000000              0000    dw      0,0,0,0         ; RESERVED
2395
2396  =
2397  =
2398  =                               ; SYSENT Table - MP/M-86, CCP/M-86 system function information
2399  =                               ;       The supervisor calls the appropriate module
2400  =                               ;       through this table.
2401  =                               ;
2402  =                               ;        Low Byte  High Byte
2403  =                               ;       +----+----+--------+
2404  =                               ;       |function |flgs|mod|
2405  =                               ;       +----+----+--------+
2406  =                               ;
2407  =                               ;       flgs    - 001h - network intercept
2408  =                               ;                 if on, the network module is called
2409  =                               ;                 first, on return, either the function
2410  =                               ;                 is called or it is considered complete
2411  =                               ;                 depending on the return.
2412  =                               ;       mod     - module number (0-15)
2413  =                               ;       function- function to call within module
2414  =
2415  =                               ;       note: sup function 0 returns not the
2416  =                               ;                 implemented error code to the caller,
2417  =                               ;                 and sup function 1 returns the illegal
2418  =                               ;                 function error code.
2419  =
2420  =                               ; standard CPM-2 functions
2421  =
2422  =                               ;       func,   module
2423  =
2424  =                               org     ((offset $) + 1) and 0fffeh
2425  =
2426  =00A0 0002              sysent  db      0,      rtm     ;  0-system reset
2427  =00A2 0004                      db      0,      cio     ;  1-conin
2428  =00A4 0104                      db      1,      cio     ;  2-conout
2429  =00A6 0001                      db      0,      sup     ;  3-raw conin/aux in
2430  =00A8 0001                      db      0,      sup     ;  4-raw conout/aux out
2431  =00AA 0404                      db      4,      cio     ;  5-list out
2432  =00AC 0504                      db      5,      cio     ;  6-raw conio
2433  =00AE 0001                      db      0,      sup     ;  7-getiobyte
2434  =00B0 0001                      db      0,      sup     ;  8-setiobyte
2435  =00B2 0604                      db      6,      cio     ;  9-conwrite
2436  =00B4 0704                      db      7,      cio     ; 10-conread
2437  =00B6 0804                      db      8,      cio     ; 11-constat
2438  =00B8 0201                      db      2,      sup     ; 12-get version
2439  =00BA 0005                      db      0,      bdos    ; 13-diskreset
2440  =00BC 0105                      db      1,      bdos    ; 14-diskselect
2441  =00BE 0205                      db      2,      bdos    ; 15-file open
```

```
2442
2443  =00C0 0305                db      3,      bdos    ; 16-file close
2444  =00C2 0405                db      4,      bdos    ; 17-search first
2445  =00C4 0505                db      5,      bdos    ; 18-search next
2446  =00C6 0605                db      6,      bdos    ; 19-file delete
2447  =00C8 0705                db      7,      bdos    ; 20-file read seq
2448  =00CA 0805                db      8,      bdos    ; 21-file write seq
2449  =00CC 0905                db      9,      bdos    ; 22-file make
2450  =00CE 0A05                db      10,     bdos    ; 23-file rename
2451  =00D0 0B05                db      11,     bdos    ; 24-login vector
2452  =00D2 0C05                db      12,     bdos    ; 25-get def disk
2453  =00D4 0D05                db      13,     bdos    ; 26-set dma
2454  =00D6 0E05                db      14,     bdos    ; 27-get alloc vector
2455  =00D8 0F05                db      15,     bdos    ; 28-write protect
2456  =00DA 1005                db      16,     bdos    ; 29-get r/O vector
2457  =00DC 1105                db      17,     bdos    ; 30-set file attr.
2458  =00DE 1205                db      18,     bdos    ; 31-get disk parm block
2459  =00E0 1305                db      19,     bdos    ; 32-user code
2460  =00E2 1405                db      20,     bdos    ; 33-file read random
2461  =00E4 1505                db      21,     bdos    ; 34-file write random
2462  =00E6 1605                db      22,     bdos    ; 35-file size
2463  =00E8 1705                db      23,     bdos    ; 36-set random record
2464  =00EA 1805                db      24,     bdos    ; 37-reset drive
2465  =00EC 1905                db      25,     bdos    ; 38-access drive
2466  =00EE 1A05                db      26,     bdos    ; 39-free drive
2467  =00F0 1B05                db      27,     bdos    ; 40-file write random w/zero fill
2468  =
2469  =                                         ;CPM-3 extensions
2470  =
2471  =00F2 0001                db      0,      sup     ; 41-Test and Write (NOT IMPLEMENTED)
2472  =                                         ; Would be BDOS func # 28
2473  =00F4 1C05                db      28,     bdos    ; 42-Lock Record
2474  =00F6 1D05                db      29,     bdos    ; 43-Unlock Record
2475  =00F8 1E05                db      30,     bdos    ; 44-Set Multi-sector
2476  =00FA 1F05                db      31,     bdos    ; 45-Set Bdos Error Mode
2477  =00FC 2005                db      32,     bdos    ; 46-Get Disk Free Space
2478  =00FE 0C01                db      12,     sup     ; 47-Chain to Program
2479  =                                         ; In CP/M-86 BDOS func # 34
2480  =0100 2105                db      33,     bdos    ; 48-Flush Buffers
2481  =0102 0101                db      1,      sup     ; 49-
2482  =
2483  =                                         ;CPM-86 extensions
2484  =
2485  =0104 0301                db      3,      sup     ; 50-call xios
2486  =0106 2205                db      34,     bdos    ; 51-set dma base
2487  =0108 2305                db      35,     bdos    ; 52-get dma
2488  =010A 0003                db      0,      mem     ; 53-get max mem
2489  =010C 0103                db      1,      mem     ; 54-get abs max mem
2490  =010E 0203                db      2,      mem     ; 55-alloc mem
2491  =0110 0303                db      3,      mem     ; 56-alloc abs mem
2492  =0112 0403                db      4,      mem     ; 57-free mem
2493  =0114 0503                db      5,      mem     ; 58-free all mem
2494  =0116 0401                db      4,      sup     ; 59-load
```

```
2495
2496  =0118 0101                db      1,      sup     ; 60-
2497  =011A 0101                db      1,      sup     ; 61-
2498  =011C 0101                db      1,      sup     ; 62-
2499  =011E 0101                db      1,      sup     ; 63-
2500  =
2501  =                                         ;CP/NET functions
2502  =
2503  =0120 4007                db      64,     net     ; 64-network login
2504  =0122 4107                db      65,     net     ; 65-network logoff
2505  =0124 4207                db      66,     net     ; 66-network send msg
2506  =0126 4307                db      67,     net     ; 67-network rcv msg
2507  =0128 4407                db      68,     net     ; 68-network status
2508  =012A 4507                db      69,     net     ; 69-get network config addr
2509  =
2510  =                                         ;CP/M-3 extensions
2511  =
2512  =012C 2405                db      36,     bdos    ; 98-Reset Alloc Vector
2513  =012E 2505                db      37,     bdos    ; 99-Truncate File
2514  =0130 2605                db      38,     bdos    ;100-Set Dir Label
2515  =0132 2705                db      39,     bdos    ;101-Return Dir Label
2516  =0134 2805                db      40,     bdos    ;102-Read File XFCB
2517  =0136 2905                db      41,     bdos    ;103-Write File XFCB
2518  =0138 2A05                db      42,     bdos    ;104-Set Date and Time
2519  =013A 2B05                db      43,     bdos    ;105-Get Date and Time
2520  =013C 2C05                db      44,     bdos    ;106-Set Default Password
2521  =013E 0D01                db      13,     sup     ;107-Return Serial Number
2522  =0140 0001                db      0,      sup     ;108-(not implemented)
2523  =0142 1904                db      25,     cio     ;109-Get/Set Console Mode
2524  =0144 1A04                db      26,     cio     ;110-Get/Set Output Delimiter
2525  =0146 1B04                db      27,     cio     ;111-Print Block
2526  =0148 1C04                db      28,     cio     ;112-List Block
2527  =
2528  =                                         ; MP/M functions
2529  =
2530  =014A 0603                db      6,      mem     ;128-mem req
2531  =014C 0603                db      6,      mem     ;129-(same function as 128)
2532  =014E 0703                db      7,      mem     ;130-mem free
2533  =0150 0102                db      1,      rtm     ;131-poll device
2534  =0152 0202                db      2,      rtm     ;132-flag wait
2535  =0154 0302                db      3,      rtm     ;133-flag set
2536  =0156 0402                db      4,      rtm     ;134-queue make
2537  =0158 0502                db      5,      rtm     ;135-queue open
2538  =015A 0602                db      6,      rtm     ;136-queue delete
2539  =015C 0702                db      7,      rtm     ;137-queue read
2540  =015E 0802                db      8,      rtm     ;138-cond. queue read
2541  =0160 0902                db      9,      rtm     ;139-queue write
2542  =0162 0A02                db      10,     rtm     ;140-cond. queue write
2543  =0164 0B02                db      11,     rtm     ;141-delay
2544  =0166 0C02                db      12,     rtm     ;142-dispatch
2545  =0168 0D02                db      13,     rtm     ;143-terminate
2546  =016A 0E02                db      14,     rtm     ;144-create process
2547  =016C 0F02                db      15,     rtm     ;145-set priority
```

```
2548
2549  =016E 0904                    db      9,      cio     ;146-console attach
2550  =0170 0A04                    db      10,     cio     ;147-console detach
2551  =0172 0B04                    db      11,     cio     ;148-set def console
2552  =0174 0C04                    db      12,     cio     ;149-console assign
2553  =0176 0501                    db      5,      sup     ;150-CLI
2554  =0178 0601                    db      6,      sup     ;151-call RPL
2555  =017A 0701                    db      7,      sup     ;152-parse filename
2556  =017C 0D04                    db      13,     cio     ;153-get def console
2557  =017E 0801                    db      8,      sup     ;154-sysdat addr
2558  =0180 0901                    db      9,      sup     ;155-time of day
2559  =0182 1002                    db      16,     rtm     ;156-get PD addr
2560  =0184 1102                    db      17,     rtm     ;157-abort process
2561  =
2562  =                             ; MPM II extensions
2563  =
2564  =0186 0F04                    db      15,     cio     ;158-attach list
2565  =0188 1004                    db      16,     cio     ;159-detach list
2566  =018A 1104                    db      17,     cio     ;160-set list dev
2567  =018C 1204                    db      18,     cio     ;161-Cond. Attach list
2568  =018E 1304                    db      19,     cio     ;162-Cond. Attach Console
2569  =0190 0801                    db      11,     sup     ;163-MP/M Version Number
2570  =0192 1404                    db      20,     cio     ;164-get list dev
2571  =
2572  =
2573  =                             ; Initialized Queues
2574  =
2575  =                             org     ((offset $) + 1) and 0fffeh
2576  =
2577  =0194 7408            mxloadqd        dw      mxdiskqd
2578  =0196 0000                            db      0,0
2579  =0198 0300                            dw      qf_keep+qf_mx
2580  =019A 40584C6F6164                    db      '@MXLoad  '
2581        2020
2582  =01A2 000001000000                    dw      0,1,0,0,1,0,0
2583        000001000000
2584        0000
2585  =01B0 0000            mxloadqpb       db      0,0
2586  =01B2 940101000000                    dw      mxloadqd,1,0
2587  =                     ;               db      '@MXLoad  '
2588  =
2589  =                             ; Data Used by Load Program
2590  =
2591  =                             org     ((offset $) + 1) and 0fffeh
2592  =
2593  =01B8 0000            lod_uda         dw      0
2594  =01BA 0000            lod_lstk        dw      0
2595  =01BC 0000            lod_based       dw      0
2596  =01BE 0000            lod_nldt        dw      0
2597  =01C0 0000            lod_pd          dw      0
2598  =01C2                 lod_fcb         rs      36
2599  =01E6 0000            lod_indma       dw      0
2600  =01E8 00              lod_nrels       db      0
```

```
2601
2602  =01E9 00              lod_chain       db      0
2603  =01EA 00              lod_user        db      0
2604  =01EB 00              lod_disk        db      0
2605  =01EC 00              lod_fifty       db      0
2606  =01ED 00              lod_8080        db      0
2607  =01EE 00              lod_1byte       db      0
2608  =01EF 0000            lod_fixrec      dw      0
2609  =01F1 0000            lod_fixrecl     dw      0
2610  =01F3                 lod_dma         rb      dskrecl
2611  =0273                 ldtab           rb      ldtabsiz
2612  =
2613  =031D                 cli_dma_ofst    rw      1
2614  =031F                 cli_dma_seg     rw      1
2615  =0321                 cli_pflag       rw      1
2616  =0323                 cli_chain       rb      1
2617  =0324                 cli_term        rb      1
2618  =0325                 cli_dma         rb      dskrecl         ;dma buffer
2619  =
2620  =                                     ;copy of user's clicb
2621  =03A5                 cli_net         rb      1               ;net
2622  =03A6                 cli_ppd         rw      1               ;parent PD
2623  =03A8                 cli_cmdtail     rb      129             ;command sent
2624  =0429                                 rb      1
2625  =
2626  =042A                 cli_fcb         rb      fcblen+1        ;internal FCB
2627  =
2628  =0448 0000            cli_cuspqpb     db      0,0             ;QPB of command
2629  =044D 00000000                        dw      0,0
2630  =0451 A603                            dw      offset cli_ppd
2631  =0453 242424242424                    db      '$$$$$$$$'
2632        2424
2633  =
2634  =0456 0000            cli_acb         db      0,0             ;cns,match
2635  =045D 0000                            dw      0               ;pd
2636  =045F 242424242424                    db      '$$$$$$$$'      ;name
2637        2424
2638  =
2639  =0467 A803            cli_pcb         dw      offset cli_cmdtail      ;parse
2640  =0469 2A04                            dw      offset cli_fcb          ;ctl bk
2641  =
2642  =0468 0000            cli_pd          dw      0       ;pd of load prog
2643  =046D 0000            cli_err         dw      0       ;error return
2644  =046F 0000            cli_bpage       dw      0       ;base page
2645  =0471 01              cli_lddsk       db      1       ;load disk
2646  =
2647  =                             ;parent information
2648  =
2649  =0472 00              cli_cns         db      0       ;pd.p_cns save
2650  =0473 00              cli_user        db      0       ;pd.p_dsk save
2651  =0474 00              cli_dsk         db      0       ;pd.p_user save
2652  =0475 00              cli_err_mode    db      0       ;u_error_mode save
2653  =0476 00              cli_dfil        db      0       ;dayfile flag
```

```
2654
2655  =
2656  =                                ;
2657  =                                ;System Initialization Variables
2658  =                                ;
2659  =
2660  =0477 0000                initpd          dw      0               ;link
2661  =0479 0000                                dw      0               ;thread
2662  =047B 00                                  db      ps_run          ;stat
2663  =047C 01                                  db      1               ;prior
2664  =047D 0500                                dw      pf_sys+pf_kernal;flag
2665  =047F 496E69742020                        db      'Init   '       ;name
2666        2020
2667  =0487 0000                                dw      unknown         ;uda segment
2668  =0489 00                                  db      0               ;disk
2669  =048A 00                                  db      0               ;user
2670  =048B 0000                                db      0,0             ;ldsk,luser
2671  =048D 0000                                dw      0               ;mem
2672  =048F 0000                                dw      0               ;dvract
2673  =0491 0000                                dw      0               ;wait
2674  =0493 0000                                db      0,0             ;org,net
2675  =0495 0000                                dw      0               ;parent
2676  =0497 00                                  db      0               ;cns
2677  =0498 00                                  db      0               ;abort
2678  =0499 0000                                db      0,0             ;cin,cout
2679  =049B 00                                  db      0               ;lst
2680  =049C 000000                              db      0,0,0           ;sf3,4,5
2681  =049F                                     rb      4               ;reserved
2682  =04A3 00000000                            dw      0,0             ;pret,scratch
2683  =
2684  =
2685  =                                ;User Data Area of Init process
2686  =                                ;paragraph aligned
2687  =
2688  =                                org     ((offset $)+0fh) AND 0fff0h
2689  =0480                     inituda         rb      ulen
2690  =0580                     init_tos        rw      0
2691  =                                org     offset inituda + ud_insys
2692  =0510 01                                  db      1               ;keep the SUP from doing stack
2693  =                                org     offset init_tos         ;switches
2694  =
2695  =
2696  =                                ; RTM data
2697  =                                ; is word aligned from init uda
2698  =
2699  =0582 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2700  =0588 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2701  =058E CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2702  =
2703  =0582 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2704  =0588 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2705  =058E CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
2706  =0594 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
```

```
2707
2708  =
2709  =05DA CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2710  =05E0 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2711  =05E6 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2712  =05EC CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2713  =
2714  =05F2 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2715  =05F8 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2716  =05FE CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2717  =0604 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2718  =
2719  =060A CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2720  =0610 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2721  =0616 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2722  =061C CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
2723  =
2724  =0622                 dsptchtos       rw      0
2725  =
2726  =0622 00              inuisp          db      false   ;?currently in dispatch?
2727  =0623 00              intflag         db      0       ;if 0, interrupts not enabled -
2728  =                                                     ;not implemented
2729  =0624 0000            es_sav          dw      0       ;(staying word aligned)
2730  =0626 0000            bx_sav          dw      0
2731  =0628 0000            ax_sav          dw      0
2732  =
2733  =                             ; MEM Data
2734  =
2735  =062A 0000            beststart       dw      0
2736  =062C 0000            bestlen         dw      0
2737  =062E 0000            bestsi          dw      0
2738  =0630 0000            bestmau         dw      0
2739  =0632 0000            currmau         dw      0
2740  =0634 0000            currsi          dw      0
2741  =0636 000000000000    currmpb         dw      0,0,0,0,0
2742        00000000
2743  =
2744  =
2745  =                     ;       SYNC Parameter Blocks
2746  =
2747  =                     ;       The MEM ENTRY: point uses the following for
2748  =                     ;       mutual exclusion and recursion.
2749  =
2750  =0640 00              mem_cnt         db      0       ;how many times a process has recursivly
2751  =                                                     ;called the memory manager
2752  =
2753  =0641 4906            mem_spb         dw      q_spb   ;link   Mem Sync Parameter Block
2754  =0643 0000                            dw      0       ;owner
2755  =0645 0000                            dw      0       ;wait
2756  =0647 0000                            dw      0       ;next
2757  =
2758  =                     ;       The queue sub-system in the RTM uses the following
2759  =                     ;       structure for mutual exclusion
```

```
2760
2761  =
2762  =0649 5106              q_spb           dw      cli_spb ;link   Queue Sync Parameter Block
2763  =064B 0000                              dw      0       ;owner
2764  =064D 0000                              dw      0       ;wait
2765  =064F 0000                              dw      0       ;next
2766  =
2767  =
2768  =                       ;       The CLI uses the CLI_SPB for mutual exclusion
2769  =
2770  =0651 5906              cli_spb         dw      thrd_spb;link   CLI Sync Parameter Block
2771  =0653 0000                              dw      0       ;owner
2772  =0655 0000                              dw      0       ;wait
2773  =0657 0000                              dw      0       ;next
2774  =
2775  =                       ;       When the thread is accessed, the THRD_SPB must be owned
2776  =                       ;       first.
2777  =
2778  =0659 4A06              thrd_spb        dw      msg_spb ;link   Thread Sync Parameter Block
2779  =065B 0000                              dw      0       ;owner
2780  =065D 0000                              dw      0       ;wait
2781  =065F 0000                              dw      0       ;next
2782  =
2783  =                       ; Currently the order in which the SYNCs must be obtained if
2784  =                       ; more than one is needed is:
2785  =
2786  =                       ;       CLI
2787  =                       ;       QUEUE                   ;called by CLI for RSPs
2788  =                       ;       MEM                     ;called by make queue
2789  =                       ;       THREAD                  ;used from the MEM module
2790  =
2791  =                       ; The SYNCs must be released in reverse order
2792  =                       ; MSG_SPB is used by the BDOS to protect the BDOS error message
2793  =                       ; buffer.  MSG_SPB is in DATA.BDO
2794
```

```
2795
2796  =                         eject 1 include data.bdo
2797  =
2798  =                 ;*****************************************************
2799  =                 ;*
2800  =                 ;*      bdos data area
2801  =                 ;*
2802  =                 ;*****************************************************
2803  =
2804  =  FFFF          CCPMOFF equ     true
2805  =
2806  =                 if BCPM
2807  =
2808  =                 ;
2809  =                 ;       8086 variables that must reside in code segment
2810  =                 ;
2811  =                 cseg    $
2812  =                 ;
2813  =                 axsave          dw      0       ; register saves
2814  =                 ss_save         dw      0
2815  =                 sp_save         dw      0
2816  =                 stack_begin     dw      endstack
2817  =                 ;
2818  =                 ;       variables in data segment:
2819  =                 ;
2820  =                 dseg    cpmsegment
2821  =                         org     bdosoffset+bdoscodesize
2822  =
2823  =                 header  rs      128
2824  =                         rs      72
2825  =
2826  =                 pag0    dw      0               ;address of user's page zero
2827  =                 ip0     db      0               ;initial page value for ip register
2828  =                 ;
2829  =                 ;     memory control block
2830  =                 ;
2831  =                 umembase        dw      0       ;user'sbase for memory request
2832  =                 umemlg  dw      0               ;length of memory req
2833  =                 contf   db      0               ;flag indicates added memory is avail
2834  =                 ;
2835  =                 ;
2836  =                 hold_info       dw      0       ;save info
2837  =                 hold_spsave     dw      0       ;save user sp during program load
2838  =                 hold_sssave     dw      0       ;save user ss during program load
2839  =
2840  =                 mod8080         db      0
2841  =                 ;
2842  =                 ;       byte i/o variables:
2843  =                 ;
2844  =                 compcol db      0               ;true if computing column position
2845  =                 strtcol db      0               ;starting column position after read
2846  =                 column  db      0               ;column position
2847  =                 listcp  db      0               ;listing toggle
```

```
2848
2849  =                       kbchar  db      0               ;initial key char = 00
2850  =
2851  =                       endif
2852  =
2853  =                               dseg
2854  =                       if BNPM
2855  =                               org     0c00h   ;org for MP/M system
2856  =                       endif
2857  =
2858  =                       if BNPM and CCPMOFF
2859  =                               org     0300h   ;org for CCP/M system
2860  =                       endif
2861  =
2862  =                       if BNPM
2863  =0800 0000              rlog            dw      0       ;removeable logged-in disks
2864  =0802 0000              tlog            dw      0       ;removeable disk test login vector
2865  =0804 0000              ntlog           dw      0       ;new tlog vector
2866  =
2867  =                       endif
2868  =
2869  =0806 0000              rodsk           dw      0       ;read only disk vector
2870  =0808 0000              dlog            dw      0       ;logged-in disks
2871  =
2872  =080A 00                open_fnd        db      0               ;open file found in srch_olist
2873  =
2874  =                       ;the following variables are set to zero upon entry to file system
2875  =
2876  =080B 00                fcbdsk          db      0               ;disk named in fcb
2877  =080C 00                parlg           db      0               ;length of parameter block copied
2878  =080D 0000              aret            dw      0               ;adr value to return
2879  =  080D                 lret            equ     byte ptr aret   ;low(aret)
2880  =080F 00                resel           db      0               ;reselection flag
2881  =0810 00                rd_dir_flag     db      0               ;must read/locate directory record
2882  =0811 00                comp_fcb_cks    db      0               ;compute fcb checksum flag
2883  =0812 00                search_user0    db      0               ;search user 0 for file (open)
2884  =0813 00                make_xfcb       db      0               ;make & search xfcb flag
2885  =0814 00                find_xfcb       db      0               ;search find xfcb flag
2886  =0815 0000              xdcnt           dw      0               ;empty directory dcnt
2887  =0817 00                DIR_CNT         DB      0               ;DIRECT I/O COUNT
2888  =0818 00                MULT_NUM        DB      0               ;MULTI-SECTOR COUNT used in bdos
2889  =0819 00                olap_type       db      0               ;record lock overlap type
2890  =081A 00                ff_flag         db      0               ;0ffh xios error return flag
2891  =081B 00                err_type        db      0               ;type of error to print if not zero
2892  =081C                   zerolength      rb      4               ;reserved bytes
2893  =  0010                 zerolength      equ     (offset $)-(offset fcbdsk)
2894  =0820 01                mult_sec        db      1               ;multi sector count passed to xios
2895  =0821 00                RELOG           DB      0               ;AUTOMATIC RELOG SWITCH
2896  =
2897  =0822 00                BLK_OFF         DB      0               ;RECORD OFFSET WITHIN BLOCK
2898  =0823 0000              blk_num         dw      0               ;block number
2899  =                       ;
2900  =0825 2708              ua_lroot        dw      offset ua0      ;unallocated block list root
```

```
2901
2902  =                       ;
2903  =0827 20030000          ua0     dw      offset ua1,0
2904  =082B FF00                      db      0ffh,0
2905  =082D 32030000          ua1     dw      offset ua2,0
2906  =0831 FF00                      db      0ffh,0
2907  =0833 39030000          ua2     dw      offset ua3,0
2908  =0837 FF00                      db      0ffh,0
2909  =0839 3F040000          ua3     dw      offset ua4,0
2910  =083D FF00                      db      0ffh,0
2911  =083F 45030000          ua4     dw      offset ua5,0
2912  =0843 FF00                      db      0ffh,0
2913  =0845 00000000          ua5     dw      0,0
2914  =0849 FF00                      db      0ffh,0
2915  =
2916  =084B                   HASH            RB      4               ;HASH CODE WORK AREA
2917  =084F 00                HASHL           DB      0               ;HASH SEARCH LENGTH
2918  =
2919  =0850 0B                LOG_FXS         db      11              ;number of log_fxs entrys
2920  =                                       ;       fxi15,fxi17,fxi19,fxi22,fxi23
2921  =0851 0204060908                        db      02h  ,04h  ,06h  ,09h  ,0ah
2922  =                                       ;       fxi30,fxi35,fxi99,fxi100,fxi102,fxi103
2923  =0856 111625262829                      db      11h  ,16h  ,25h  ,26h   ,28h   ,29h
2924  =085C 00000000                          db      0,0,0,0         ;reserved
2925  =0860 07                RW_FXS          db      7               ;number of rw_fxs entrys
2926  =                                       ;       fxi20,fxi21,fxi33,fxi34,fxi40,fxi42,fxi43
2927  =0861 0708141518 1C                     db      07h  ,08h  ,14h  ,15h  ,1bh  ,1ch  ,1dh
2928          1D
2929  =0868 0000                              db      0,0             ;reserved
2930  =086A 02                SC_FXS          db      2               ;number of sc_fxs entrys
2931  =                                       ;       fxi16,fxi18
2932  =086B 0305                              db      03h  ,05h
2933  =086D 0000                              db      0,0             ;reserved
2934  =
2935  =086F 00                DEBLOCK_FX      DB      0               ;DEBLOCK FUNCTION #
2936  =0870 00                PHY_OFF         DB      0               ;RECORD OFFSET WITHIN PHYSICAL RECORD
2937  =0871 0000              CUR_BCBA        DW      0               ;CURRENT BCB OFFSET
2938  =0873 0000              ROOT_BCBA       DW      0               ;ROOT BCB OFFSET
2939  =0875 0000              EMPTY_BCBA      DW      0               ;EMPTY BCB OFFSET
2940  =
2941  =                       if BMPM
2942  =
2943  =0877 0000              P_LAST_BCBA     DW      0               ;PROCESS'S LAST BCB OFFSET
2944  =0879 00                P_BCB_CNT       DB      0               ;PROCESS BCB COUNT IN BCB LIST
2945  =
2946  =                       endif
2947  =
2948  =087A 00                fx_intrn        db      0               ;internal BDOS function number
2949  =
2950  =087B 0000              TRACK           DW      0       ;BCB RECORD'S TRACK
2951  =087D 0000              SECTOR          DW      0       ;BCB RECORD'S SECTOR
2952  =
2953  =                       ;       seldsk,usrcode are initialized as a pair
```

```
2954
2955  =087F 00          seldsk      db      0       ;selected disk num
2956  =0880 00          usrcode     db      0       ;curr user num
2957  =
2958  =0881 0000        info        dw      0       ;info adr
2959  =0883 0000        searcha     dw      0       ;search adr
2960  =
2961  =                 ;the following variable order is critical
2962  =
2963  =                 ;variables copied from uda for mp/m                   x
2964  =
2965  =                 ;variables included in fcb checksum for mp/m and cp/m     x
2966  =
2967  =                 ;variables used to access system lock list for mp/m       x
2968  =
2969  =0885 0000        dma_ofst    dw      0       ;dma offset           1
2970  =0887 0000        dma_seg     dw      0       ;dma segment          2
2971  =0889 00          func        db      0       ;bdos function #      3
2972  =088A 00          searchl     db      0       ;search len           4
2973  =
2974  =                 if BMPM
2975  =
2976  =088B 0000        searchaofst dw      0       ;search adr ofst      5
2977  =088D 0000        searchabase dw      0       ;search adr base      6
2978  =
2979  =                 endif
2980  =
2981  =088F 0000        dcnt        dw      0       ;directory counter    7
2982  =0891 0000        dblk        dw      0       ;directory block      8 ?? - not used - ??
2983  =0893 00          error_mode  db      0       ;bdos error mode      9
2984  =0894 00          mult_cnt    db      0       ;bdos multi-sector cnt 10
2985  =0895             df_password rb      8       ;process default pw   11
2986  =
2987  =                 if BMPM
2988  =
2989  =089D 00          pd_cnt      db      0       ;bdos process cnt     12 1
2990  =
2991  =                 endif
2992  =
2993  =089E 00          high_ext    db      0       ;fcb high extent bits     2
2994  =089F 00          xfcb_read_only db   0       ;xfcb read only flag      3
2995  =08A0 FF          curdsk      db      0ffh    ;current disk             4 1
2996  =
2997  =                 if CPM
2998  =
2999  =08A1 00          packed_dcnt db      0       ;packed dblk+dcnt           2
3000  =08A2 00                      db      0
3001  =08A3 00                      db      0
3002  =08A4 0000        pdaddr      dw      0       ;process descriptor addr    3
3003  =
3004  =                 endif
3005  =
3006  =                         org ((offset $) + 1) and 0fffeh
```

```
3007
3008  =
3009  =                         ;       curtrka - alloca are set upon disk select
3010  =                         ;       (data must be adjacent)
3011  =
3012  =0836 0000               curtrka         dw      0       ;ptr to cur dir max val
3014  =0838 0000               drvlbla         dw      0       ;drive label data byte addr
3014  =083A 0000               buffa           dw      0       ;ptr to dir dma addr
3015  =083C 0000               dpbaddr         dw      0       ;curr disk param block addr
3016  =083E 0000               checka          dw      0       ;curr checksum vector addr
3017  =0840 0000               alloca          dw      0       ;curr alloc vector addr
3018  =0842 0000               DIR_BCBA        DW      0       ;DIRECTORY BUFFER CONTROL BLOCK ADDR
3019  =0844 0000               DAT_BCBA        dw      0       ;DATA BUFFER CONTROL BLOCK ADDR
3020  =0846 0000               HASH_SEG        dw      0       ;HASH TABLE SEGMENT
3021  =  000C                  addlist         equ     12      ;"$-buffa" = addr list size
3022  =
3023  =                         ;       sectpt - offset obtained from disk parm block at dpbaddr
3024  =                         ;       (data must be adjacent)
3025  =
3026  =0848 0000               sectpt          dw      0       ;sectors per track
3027  =084A 00                 blkshf          db      0       ;block shift factor
3028  =084B 00                 blkmsk          db      0       ;block mask
3029  =084C 00                 extmsk          db      0       ;extent mask
3030  =084D 0000               maxall          dw      0       ;max alloc num
3031  =084F 0000               dirmax          dw      0       ;max dir num
3032  =0851 0000               dirblk          dw      0       ;reserved alloc bits for dir
3033  =0853 0000               chksiz          dw      0       ;size of checksum vector
3034  =0855 0000               offsetv         dw      0       ;offset tracks at beginning
3035  =0857 00                 PHYSHF          DB      0       ;PHYSICAL RECORD SHIFT FACTOR
3036  =0858 00                 PHYMSK          DB      0       ;PHYSICAL RECORD MASK
3037  =0859                    endlist         rs      0       ;end of list
3038  =  0011                  dpblist         equ     (offset endlist)-(offset sectpt)
3039  =                                                        ;size
3040  =
3041  =                         ;       local variables
3042  =
3043  =0859                    common_dma      rb      16      ;copy of user's dma 1st 16 bytes
3044  =0869 00                 make_flag       db      0       ;make function flag
3045  =086A 00                 actual_rc       db      0       ;directory ext record count
3046  =086B 00                 save_xfcb       db      0       ;search xfcb save flag
3047  =086C 00                 save_mod        db      0       ;open_reel module save field
3048  =086D 00                 pw_mode         db      0       ;password mode
3049  =086E 00                 attributes      db      0       ;fcb interface attributes hold byte
3050  =
3051  =                         if BMPX
3052  =
3053  =                         ;       number of lock list items required for lock operation
3054  =086F 010002020101       required_table  db      1,0,2,2,1,1,2,2,1,1,2,2,1,1,2,2
3055          020201010202
3056          01010202
3057  =
3058  =0881 00                 chk_olist_flag  db      0       ;check | test olist flag
3059  =0882 0000               lock_sp         dw      0       ;lock stack ptr
```

```
3060
3061  =08F2 00              lock_shell      db      0       ;lock shell flag
3062  =08F3 00              check_fcb_ret   db      0       ;check_fcb return switch
3063  =08F4 00              lock_unlock     db      0       ;lock | unlock function flag
3064  =08F5 00              incr_pdcnt      db      0       ;increment process_cnt flag ??
3065  =08F6 00              free_mode       db      0       ;free lock list entries flag ??
3066  =                                                     ;1=free entries for curdsk
3067  =                                                     ;0=free all entries
3068  =08F7 0000            cur_pos         dw      0       ;current position in lock list
3069  =08F9 0000            prv_pos         dw      0       ;previous position in lock list
3070  =
3071  =08FB 00              dont_close      db      0       ;inhibit actual close flag
3072  =08FC 00              open_cnt        db      0       ;process open file count
3073  =08FD 00              lock_cnt        db      0       ;process locked record count
3074  =08FE 0000            file_id         dw      0       ;address of file' lock list entry
3075  =0900 00              set_ro_flag     db      0       ;set drive r/o flag
3076  =0901 00              check_disk      db      0       ;disk reset open file check flag
3077  =0902 00              flushed         db      0       ;lock list open file flush flag
3078  =
3079  =                     ;free_root, lock_max, open_max initialized by sysgen
3080  =
3081  =0903 5200                            dw      offset free_root
3082  =0905 0000            open_root       dw      0       ;lock list open file list root
3083  =0907 0000            lock_root       dw      0       ;lock list locked record list root
3084  =
3085  =                     endif
3086  =
3087  =0909 0000            sdcnt           dw      0       ;saved dcnt of file's 1st fcb
3088  =090B 0000            sdcnt0          dw      0       ;saved dcnt (user 0 pass)
3089  =
3090  =                     if BCPM
3091  =
3092  =                     chain_flag      db      0       ;chain flag ??
3093  =                     tod             db      0ffh,0ffh,0ffh,0ffh ??
3094  =
3095  =                     endif
3096  =
3097  =
3098  =090D 00              rmf             db      0       ;read mode flag for open$reel
3099  =090E 00              wflag           db      0       ;xios/bios write flag
3100  =090F 00              dirloc          db      0       ;directory flag in rename, etc.
3101  =0910 00              linfo           db      0       ;low(info)
3102  =0911 00              dminx           db      0       ;local for diskwrite
3103  =0912 00              single          db      0       ;set true if single byte
3104  =                                                     ;alloc map
3105  =0913 00              rcount          db      0       ;record count in curr fcb
3106  =0914 00              extval          db      0       ;extent num and extmsk
3107  =0915 00              vrecord         db      0       ;curr virtual record
3108  =0916 00              ADRIVE          DB      0       ;CURRENT DISK - must preceed arecord
3109  =0917 0000            arecord         dw      0       ;curr actual record
3110  =0919 00                              db      0       ;curr actual record high byte
3111  =091A 0000            ARECORD1        dw      0       ;curr actual block# * blkmsk
3112  =091C 0000            arec            dw      0       ;curr actual directory record
```

```
3113
3114  =0918 0000              CUR_dma_seg     dw      0
3115  =091A 0000              CUR_DMA         dw      0
3116  =
3117  =                       ;       local variables for directory access
3118  =
3119  =091C 00                dptr            db      0       ;directory pointer 0,1,2,3
3120  =  008F                  ldcnt           equ     byte ptr dcnt   ;low(dcnt)
3121  =091D 00                user_zero_pass  db      0       ;search user zero flag
3122  =
3123  =                       ;       shell variables
3124  =
3125  =091E 0000              shell_si        dw      0       ;bdos command offset
3126  =0920 000000            shell_rr        db      0,0,0   ;r0,r1,r2 save area
3127  =
3128  =                       ;       special 8086 variables:
3129  =
3130  =0923 0000              uda_save        dw      0       ;user data area saved value
3131  =0925 0000              parametersegment dw     0       ;user parameter segment
3132  =0927 0000              returnseg       dw      0       ;user return segment
3133  =
3134  =                       ;       error messages
3135  =
3136  =0929 00                err_drv         db      0
3137  =092A 0000              err_pd_addr     dw      0
3138  =
3139  =092C 0D0A43502F4D      dskmsg          db      13,10,'CP/M Error On '
3140        204572726F72
3141        204F6E20
3142  =0942 20342000          dskerr          db      ' : ',0
3143  =
3144  =0946 0000600969090     xerr_list       dw      0,permsg,rodmsg,rofmsg,selmsg
3145        78098709
3146  =0950 8309C709DC09              dw      xe5,xe6,xe7,xe8,xe9,xe10,xe11,xe3
3147        8B09FF09100A
3148        290A9509
3149  =
3150  =0960 446973683A2049    permsg          db      'Disk I/O',0
3151        2F4F00
3152  =0969 526561642F4F      rodmsg          db      'Read/Only Disk',0
3153        6E6C79204469
3154        736B00
3155  =0978 526561642F4F      rofmsg          db      'Read/Only File',0
3156        6E6C79204669
3157        6C6500
3158  =0987 496E76616C69      selmsg          db      'Invalid Drive',0
3159        642044726976
3160        6500
3161  =
3162  =0995 46696C652043      xe3             db      'File Opened in Read/Only Mode',0
3163        6F70656E656420
3164        696E20526561
3165        642F4F6E6C79
```

```
3166
3167                    20496F646500
3168  =
3169  =0935 46696C652043        xe5             db      'File Currently Open',0
3170                    757272656E74
3171                    6C792047F065
3172                    6E00
3173  =0967 436C6F736520        xe6             db      'Close Checksum Error',0
3174                    436865636B73
3175                    756D20457272
3176                    6F7200
3177  =090C 50617373776F        xe7             db      'Password Error',0
3178                    726420457272
3179                    6F7200
3180  =09Eb 46696C652041        xe8             db      'File Already Exists',0
3181                    6C7265616479
3182                    204578697374
3183                    7309
3184  =09Fr 496C6C656761        xe9             db      'Illegal ? in FCB',0
3185                    6C203F20696F
3186                    204643B200
3187  =0A10 4F70656E2046        xe10            db      'Open File Limit Exceeded',0
3188                    696C65204C69
3189                    6D6974204578
3190                    636565646564
3191                    00
3192  =0A29 4E6F20526F6F        xe11            db      'No Room in System Lock List',0
3193                    6D20696E2053
3194                    797374656D20
3195                    4C6F636B204C
3196                    69737400
3197  =
3198  =0A45 0D0A00              crlf_str        db      13,10,0
3199  =0A48 0D0A42646F73        pr_fx           db      13,10,'Bdos Function = '
3200                    2046756E6374
3201                    696F6E203D20
3202  =0A5A 202020              pr_fx1          db      '   '
3203  =0A5D 2046696C6520        pr_fcb          db      ' File = '
3204                    3D20
3205  =0A65                     pr_fcb1         rs      12
3206  =0A71 00                  pr_fcb2         db      0
3207  =
3208  =0A72 0D0A44697361B       deniedmsg       db      13,10,'Disk reset denied, Drive ' .
3209                    207265736574
3210                    2064656E6965
3211                    642C20447269
3212                    765520
3213  =0A8D 003A                deniedrv        db      0,':'
3214  =0A8F 20436F6E736F        deniedcns       db      ' Console '
3215                    6C6520
3216  =0A98 00                  deniedcns       db      0
3217  =0A99 2050726F6772        deniedprg       db      ' Program '
3218                    616D20
```

```
3219
3220  =0AA2 313233343536      deniedprc       db      '12345678',0
3221        373800
3222  =
3223  =                       if BNPM
3224  =
3225  =                       ;       bdos stack switch variables and stack
3226  =                       ;       used for all bdos disk functions
3227  =
3228  =                               org     ((offset $) + 1) and 0fffeh
3229  =
3230  =                               ; 69 word bdos stack
3231  =
3232  =0AAC CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3233  =0AB2 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3234  =0AB8 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3235  =0ABE CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3236  =0AC4 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3237  =0ACA CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3238  =0AD0 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3239  =0AD6 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3240  =0ADC CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3241  =0AE2 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3242  =0AE8 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3243  =0AEE CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3244  =0AF4 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3245  =0AFA CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3246  =0B00 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3247  =0B06 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3248  =0B0C CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3249  =0B12 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3250  =0B18 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3251  =0B1E CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3252  =0B24 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3253  =0B2A CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3254  =0B30 CCCCCCCCCCCC              dw      0cccch,0cccch,0cccch
3255  =0B36                   bdosstack       rw      0
3256  =
3257  =0B36                   save_sp         rw      1
3258  =0B38                   ss_save         rw      1
3259  =0B3A                   sp_save         rw      1
3260  =
3261  =                       ;       local buffer area:
3262  =
3263  =0B3C                   info_fcb        rb      40      ;local user FCB
3264  =0B64                   save_fcb        rb      16      ;fcb save area for xfcb search
3265  =
3266  =0B74 0000              mxdiskqd        dw      0       ;link
3267  =0B76 0000                              db      0,0     ;net,orq
3268  =0B78 0300                              dw      qf_keep+qf_mx   ;flags (MX queue)
3269  =0B7A 4D5864697368              db      'MXdisk  '
3270        2020
3271  =0B82 00000100                  dw      0,1     ;msglen,nmsgs
```

```
3272
3273  =0086 00000000                    dw      0,0     ;nq,dq
3274  =008A 01000000                    dw      1,0     ;msjcnt,out
3275  =008F 0000                        dw      0       ;buffer ptr
3276  =
3277  =0091 00              mxdiskqpb   db      0       ;flgs
3278  =0092 00                          db      0       ;nat
3279  =0093 7408                        dw      mxdiskqd        ;qaddr
3280  =0095 0100                        dw      1       ;nmsgs
3281  =0097 0000                        dw      0       ;buffer
3282  =0099 4D5846697368                db      'MXdisk  '
3283        2020
3284  =
3285  =                       ;       error Message sync param block for mutual exclusion
3286  =
3287  =00A0 0000              msg_spb     dw      0       ;link   Error Message sync parameter block
3288  =00A2 0000                          dw      0       ;owner
3289  =00A4 0000                          dw      0       ;wait
3290  =00A6 0000                          dw      0       ;next
3291  =                       endif
3292  =
3293  =                       if BCPM
3294  =
3295  =                       ;
3296  =                       ;       special 8086 variables:
3297  =                       ;
3298  =                       ioloc           db      0       ;iobyte
3299  =                       user_parm_seg   dw      0       ;holds user parameter seg during load
3300  =                       nallocmem       db      0       ;no. of allocated memory segments
3301  =                       ncrmem          db      0       ;no. of available memory segments
3302  =                       crmem           dw      0,0     ;memory table (16 elements)
3303  =                                       dw      0,0,0,0,0,0
3304  =                                       dw      0,0,0,0,0,0,0,0
3305  =                                       dw      0,0,0,0,0,0,0,0
3306  =                                       dw      0,0,0,0,0,0,0,0
3307  =                       ;
3308  =                       mem_stack_length        equ     40
3309  =                       memstack        rs      mem_stack_length
3310  =                                       ;8 possible allocations
3311  =                       stbase  equ     word ptr 0
3312  =                       stlen   equ     word ptr 2
3313  =                       ccpflag equ     byte ptr 4
3314  =                       nccpalloc       db      0       ;number of current ccp allocations
3315  =                       mem_stk_ptr     dw      0       ;current memory stack location
3316  =
3317  =                       stackarea       rw      ssize   ;stack size
3318  =                       endstack        rb      0       ;top of stack
3319  =                       ;
3320  =                       endif
3321  =                                       org     0fffh
3322  =                       if CCPROFF
3323  =                                       org     0bfffh
3324  =                       endif
```

```
3325
3326  =05FF 00                  db      0
3327
```

```
3328
3329                                    eject 1 end
3330
3331
3332  END OF ASSEMBLY.  NUMBER OF ERRORS:    0.  USE FACTOR: 54%
```

```
ABSMAXENTRY       0033 L   631   759#
ACTUALRC          03CA V  3045#
ADDLIST           000C N  3021#
ADRIVE            0916 V  3106#
ALLOCA            08B0 V  3017#
AMOM              00FC L   774   776#
ARECORD           0917 V  3109#
ARECORD1          091A V  3111#
ARET              08CD V  2879# 2879
ATTRIBUTES        08DE V  3049#
AXSAV             0628 V  2731#
BCPM              0000 N    69#   70  2805  3090  3293
BDOS              0005 N    91#  234  2439  2440  2441  2443  2444  2445  2446  2447
                            2448  2449  2450  2451  2452  2453  2454  2455  2456  2457
                            2458  2459  2460  2461  2462  2463  2464  2465  2466  2467
                            2473  2474  2475  2476  2477  2490  2486  2487  2512  2513
                            2514  2515  2516  2517  2518  2519  2520
BDOSADD           0020 N  2279#
BDOSNUMBIT        0008 N   100#
BDOSSTACK         0B36 V  3255#
BESTLEN           062C V  1226  1280  1287  2736#
BESTRAD           0630 V  1238  1283  1288  2738#
BESTSY            062E V  1232  1289  2737#
BESTSTART         062A V  1109  1164  1222  1282  1286  2735#
BLKMSK            08BB V  3028#
BLKNUM            0823 V  2898#
BLKOFF            0822 V  2897#
BLKSHF            08BA V  3027#
BMPM              FFFF N    70# 2854  2858  2862  2941  2974  2987  2997  3051  3223
BUFFA             08AA V  3014#
BVERNUM           007A V  2357# 2362#
BXSAV             0626 V  2730#
CAAG              0164 L   833   841#
CAAGM             016D L   835   841   845#
CAAN              014C L   832#  834
CAGM              0124 L   795   797#
CCB               0054 V  2326#
CCPM              FFFF N    67# 1167  2361
CCPMOFF           FFFF N  2804# 2858  3322
CDRMAXA           08A6 V  3012#
CHECKA            08AE V  3016#
CHECKDISK         0901 V  3076#
CHECKFORRET       08F3 V  3062#
CHKPBST           03CC L  1266  1268#
CHKVECTFLAG       08FF V  3058#
CHKSIZ            08C3 V  3033#
CIO               0004 N    90#  225   226   227   228   229  2427  2428  2431  2432
                            2435  2436  2437  2523  2524  2525  2526  2549  2550  2551
                            2552  2556  2564  2565  2566  2567  2568  2570
CIOMOD            0018 N  2275#
CIOMODBIT         0010 N   101#
CLIACB            045B V  2634#
CLIBPAGE          046F V  2644#
```

```
CLICRLIN        0323 V  2616#
CLICMDTAIL      03A6 V  2625# 2639
CLICNS          0472 V  2649#
CLICUSPOPN      044E V  2628#
CLIDFIL         0476 V  2653#
CLIDMA          0325 V  2618#
CLIDMADFST      031D V  2613#
CLIDMASEG       031F V  2614#
CLIDSK          0474 V  2651#
CLIERR          046D V  2645#
CLIERRMODE      0475 V  2652#
CLIFCB          042A V  2626# 2640
CLILDDSK        0471 V  2645#
CLINET          03A5 V  2621#
CLIPGM          0467 V  2639#
CLIPD           046B V  2642#
CLIPFLAG        0321 V  2615#
CLIPPD          03A6 V  2622# 2630
CLISP9          0651 V  2762  2770#
CLITERM         0324 V  2617#
CLIUSER         0473 V  2650#
CMOD            0090 V  2386#
COMMONDMA       08C9 V  3043#
COMPTOCKS       0611 V  2882#
CPMABSAENTRY    0133 L   633   804#
CPMALLOCENTRY   010E L   632   782#
CPMBIGEXT       0102 L   888#  893   897
CPMFREEENTRY    0199 L   634   869#
CPMFRODT        01AB L   886#  896
CRLFSTR         0345 V  3198#
CS              SREG V   663   706
CURBUSA         0871 V  2937#
CURDMA          0920 V  3115#
CURDMASEG       091E V  3114#
CURDSK          08B0 V  2995#
CURFCS          08F7 V  3068#
CURRECD         0632 V  1252  1277  1288  2739#
CURRMPB         0636 V  1258  1262  1270  1285  2741#
CURRSI          0634 V  1251  1289  2740#
DATECPA         08B4 V  3019#
DAYFILE         004F V  2319#
DBLK            0891 V  2982#
DCNT            088F V  2981# 3120
DEBLOCKFK       066F V  2935#
DEBUGINT        0CE1 N    56#
DENIEDCMS       0498 V  3216#
DENIEDDRV       0A9D V  3215#
DENIEDMSG       0A72 V  3208#
DENIEDPRC       0AA2 V  3220#
DEVVER          000C V   605#
DFPASSWORD      0895 V  2985#
DIRBUSA         08B2 V  3018#
DIRBLK          08C1 V  3032#
```

```
DIRCNT          0b17 V  2837#
DIRLOC          090F V  3100#
DIRMAX          089F V  3031#
DISPATCHER      0038 N  2292#
DLOG            0808 V  2870#
DLR             006A V  2344#
DMADEST         0885 V  2969#
DMASEG          0887 V  2970#
DMINX           0911 V  3102#
DONTCLOSE       08FB V  3071#
DPBADDR         08AC V  3015#
DPBLIST         0011 N  3038#
DPIR            0922 V  3119#
DREC            091C V  3112#
DRL             006C V  2345#
DRVLOLA         08A8 V  3013#
DS              SREG V   747   747   752   752   756   770   770   772   776
                         776   779   792   792   794   797   797   800   819   819
                         823   845   845   848   863   863   865   881   881   884
                         905   905   909  1140  1143  1195  1197  1201  1255  1256
                        1260  1295  1297  1304  1320  1413  1416  1476  1476  1480
                        1503  1503  1511  1520  1608  1654  1673  1714  1806  1828
                        2008  2014
DSKERR          0942 V  3142#
DSKMSG          0932 V  3139#
DSKRECL         0080 N    49#  2610  2618
DSPTCHTOS       0622 V  2724#
EABORT          0028 N   497#
EACTIVEPD       0022 N   491#  1057  1371
EBADENTRY       0002 N   459#
EBADFNAME       0018 N   481#
EBADFTYPE       0019 N   482#
EBADLOAD        001C N   485#
EBADOPEN        001E N   487#
EBADREAD        001D N   486#
EFIXUPREC       0029 N   498#
EFLAGOVRUN      0005 N   462#
EFLAGUNDERUN    0006 N   463#
EILLCNS         0013 N   476#
EILLDISK        0017 N   480#
EILLFLAG        0004 N   461#
EILLLST         0025 N   494#
EILLMD          001B N   484#
EILLPASSWD      0026 N   495#
EMPTYBCBA       0875 V  2939#
ENCLIP          0016 N   479#
ENCLIQ          0010 N   473#
ENDLST          08C9 V  3037#  3038
ENDSEG          0044 V  2302#
ENDATTACH       0024 N   493#
ENDCHAR         001A N   485#
ENDCHSWITCH     0015 N   478#
ENDCSEG         0021 N   490#
```

```
ENDMEMORY          0003 N   460#  876  1098  1398  1521  1630  1712  1881
ENDMINIC           0027 N   496#
ENDPD              000C N   469#
ENDPDNAME          0014 N   477#
ENDQUEUE           0008 N   465#
ENDQD              0007 N   464#
ENDQUEUE           0009 N   466#
ENDTIMPLEMENTED    0001 N   458#
ENDTOMER           0020 N   489#
ENDUMD             0012 N   475# 2070
ENTRY              005D L   597   646#
ENULLCMD           001F N   488#
EPDNOTERM          0023 N   492#
EQEMPTY            000E N   471#
EQFULL             000F N   472#
EQINUSE            000A N   467#
EQPROTECTED        000D N   470#
ERRDRV             092F V  3136#
ERRINTERCEPT       0092 V  2390#
ERRDRMODE          0893 V  2983#
ERRPDADDR          0930 V  3137#
ERRTYPE            0818 V  2891#
ES                 SREG V  1035  1035  1036  1039  1041  1077  1077  1078  1079  1081
                           1097  1102  1103  1104  1111  1115  1220  1220  1224  1227
                           1228  1230  1234  1240  1255  1257  1260  1269  1269  1273
                           1274  1343  1343  1345  1346  1347  1598  1598  1601  1617
                           1634  1637  1640  1645  1646  1648  1655  1670  1670  1672
                           1679  1691  1699  1714  1806  1806  1815  1828  1828  1830
                           1892  1892  1899  1902  1905  1908  1919  1923  1947  1961
                           2005  2005  2007
ESSAV              0624 V  2729#
EXTMSK             0A8C V  3029#
EXIVAL             0914 V  3106#
FALSE              0000 N    47#   66    69  2387  2726
FARA               021A L   933   938   953#
FARE               0221 L   929   956#
FARM               01F2 L   925#  944
FBDOSTERM          052D N   234#
FCALLBIOS          0032 N   149#
FCBDSK             080B V  2876# 2893
FCBLEN             0020 N    50# 2626
FCCONATTCH         00A2 N   194#
FCIOSTAT           0418 N   227#
FCIOTERM           0417 N   226#
FCLOAD             010E N   201#
FCONASSIGN         0095 N   181#
FCONATTACH         0092 N   177#
FCONDETACH         0093 N   179#
FCONOUT            0002 N   108#
FCONPRINT          040E N   225#
FCONREAD           000A N   116#
FCONWRITE          0009 N   115#
FCOREAD            008A N   169#
```

```
FCQWRITE          008C N   171#
FCREATEPROC       0090 N   175#
FDELIN            006E N   159#
FDISPATCH         009E N   173#
FFCLOSE           0010 N   122#
FFFLAG            031A V  2890#
FFINDPNAME        0214 N   208#
FFLAGSET          0085 N   164#
FFLAGWAIT         0084 N   163#
FFOPEN            000F N   121#
FFREADRDM         0021 N   140#
FFREADSEQ         0014 N   127#
FFREEALL          003A N   157#
FFRELMEM          0039 N   156#
FGETDEFDISK       0019 N   132#
FILEID            08FE V  3074#
FINDXFCB          0814 V  2885#
FINFLAGSET        0203 N   205#
FLAGMIN           0003 N    63#
FLAGS             0056 V  2327#
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  3077#
FMALLOC           0080 N   160#  664
FMAUXALLOC        0309 N   218# 1263
FMAUFREE          030A N   219# 1298  1414
FMEMFREE          0082 N   161#  903  1198
FMLALLOC          0308 N   220# 1141
FMLFREE           030C N   221# 1315  1448
FNAMSTZ           0008 N    53#
FNUABORT          0217 N   211#  679
FNUABURTSPEC      0219 N   213#
FUKABORT          0218 N   212#  697
FPARSEFILENAME    0098 N   184#
FQMAKE            0086 N   165#
FQOPEN            0087 N   166#
FQREAD            0089 N   168#
FQWRITE           0088 N   170#
FRCONIN           0402 N   228#
FRCONOUT          0403 N   229#
FREEALLENTRY      01E8 L   635   913#  955
FREEND            09F2 L  1099  1436  1520  2032  2080#
FREEMEMORY        01D5 L   885   895   899#  954
FREEMODE          08F6 V  3065#
FREEROOT          0052 V  2325# 3081
FREESEG           01C9 L   891   894#
FREEWORST         0415 L  1280  1284  1291#
FSETDMA           001A N   133#
FSETDMAB          0033 N   150#
```

```
FSETPRIOR       0091 N   176#
FSETRNDREC      0024 N   143#
FSHARE          0308 N   217#
FSLEEP          0212 N   206#
FSYNC           0215 N   209#  673  1047  1397
FTERMINATE      008F N   174#
FTYPSIZ         0003 N    54#
FUNC            0889 V  2971#
FUNCTAB         0043 V   630#  663
FUNSYNC         0216 N   210#  695  1060  1063  1368  1376
FUSERCODE       0020 N   139#
FWAKEUP         0213 N   207#
FXINTRN         087A V  2948#
GETMD           09CF L  1029  1471  1977  2063#
GETMEMORY       017C L   750   774   795   841   850#
GMDRET          09F0 L  2072  2077#
HASH            084B V  2916#
HASHL           084F V  2917#
HASHSEG         0836 V  3020#
HIGHEXT         089E V  2993#
INCRPDCNT       08F5 V  3064#
INDISP          0622 V  2726#
INFO            0881 V  2958#
INFOFCB         083C V  3263#
INIT            0042 L   596   620#
INITFD          0477 V  2343  2348  2660#
INITIOS         0530 V  2690# 2693
INITUDA         0480 V  2689# 2691
INITFLAG        0623 V  2727#
LCB             0086 V  2330#
LDCNT           08CF V  3120#
LDIAB           0273 V  2611#
LDIABSIZ        00AA N    64# 2611
LINFO           0910 V  3101#
LOCKCNT         08FD V  3073#
LOCKMAX         008A V  2382#
LOCKROOT        0907 V  3083#
LOCKSHELL       08F2 V  3061#
LOCKSP          08F0 V  3059#
LOCKUNLOCK      08F4 V  3063#
LODBOOT         01ED V  2606#
LODBASEP        01BC V  2595#
LODCHAIN        01E9 V  2602#
LODDISK         01EB V  2604#
LODDMA          01F3 V  2610#
LODFCB          01C2 V  2598#
LODFIFTY        01EC V  2605#
LODFIXREC       01EF V  2608#
LODFIXREC1      01F1 V  2609#
LODLOADMA       01E6 V  2599#
LODLOGIN        01EE V  2607#
LODLSTK         01BA V  2594#
LODNEXT         01BE V  2596#
```

```
LODNRELS          01E8 V  2600#
LODPD             01C0 V  2597#
LODUDA            0198 V  2593#
LODUSER           01EA V  2603#
LOGFXS            0850 V  2919#
LRET              0300 V  2879#
LSCSEG            0058 V  2225#
LSFLAGS           005A V  2226#
LSOFFSET          0056 V  2224#
LSRCSEG           005E V  2228#
LSROFFSET         005C V  2227#
LSSP              0056 V  2223#
LSTKLEN           0060 N   949  2218# 2221
MAFLEN            0006 N   583#
MAFMAU            0000 N   580#  581  1672
MAFSAT            0002 N   581#  582  1679
MAFSTART          0004 N   582#  583  1691  1699
MAKEFLAG          0809 V  3044#
MAKEXFCB          0813 V  2884#
MAL               0076 V  1150  1152  1308  1441  2350#
MALLERR           02CF L  1063  1099#
MALLOMD           0231 L  1030  1032#
MALLOPD           0249 L  1037  1040#
MALLOU            0388 L  1233  1236#
MALLLINKAT        0348 L  1136  1147  1160#
MALLML            0316 L  1121  1138# 1215
MALLMLU           0338 L  1145  1148#
MALLNEXT          02FA L  1119# 1122  1133
MALLWOMEN         02CC L  1098# 1168  1182
MALLOCENTRY       0226 L   636   985#
MALLPDNXT         0264 L  1053# 1055
MALLPDV1          0298 L  1042  1076#
MALLPOVER         0287 L  1054  1056  1065#
MALLTRYMAU        039E L  1130  1156  1242#
MAUALLOC          0686 L  1638  1642  1644#
MAUAERR           0655 L  1604  1606  1614  1615  1629#
MAUALLOCENTRY     05F8 L   640  1582#
MAUAMBS           065D L  1618  1623  1633#
MAUAMSAT          061D L  1612# 1616  1620  1622  1636  1641
MAUAOUT           06A2 L  1632  1652  1653#
MAUAREL           0616 L  1602  1607#
MAUASPLIT1        0681 L  1639  1643#
MAUASTART         0619 L  1609# 1628
MAUCFNORULE       0761 L  1756  1760#
MAUCMTCHA         0744 L  1749  1751  1752  1788#
MAUCMIN           07A5 L  1792# 1797
MAUCNFIDLE        0798 L  1776  1782#
MAUCNOTFREE       0770 L  1753  1771#
MAUCNSAT          0734 L  1747# 1762  1770  1774  1781  1786
MAUCOLLAPSE       0730 L  1650  1710  1738#
MAUCOUT           07C3 L  1794  1796  1799#
MAUFC             06F6 L  1708  1710#
MAUFERR           06FB L  1688  1702  1711#
```

```
MAUFREE          06F3 L  1692  1709#
MAUFRSAT         06BF L  1686# 1690
MAUFRSBA         06D6 L  1694  1699#
MAUFOUT          0701 L  1695  1710  1713#
MAUFREEENTRY     06A9 L   641  1658#
MAUINSERT        07F2 L  1726  1779  1817#
MAUIR            081D L  1825  1834#
MAURELEASE       07C8 L  1767  1802#
MAUSERR          072C L  1728  1733#
MAUSPLIT         0708 L  1626  1643  1707  1717#
MAXALL           083D V  3030#
MAXCHKDONE       02C0 L  1090  1093#
MAXCHKNXT        0294 L  1088# 1092
MAXMEMORY        00B7 L   630   737#
MAXOK            02E6 L  1103  1105#
MCBASE           0000 N   730#  731   754   771   778   799   821   847   884
MCBEXT           0004 N   732#  733   755   883
MCBLEN           0005 N   733#
MCBLENGTH        0002 N   731#  732   748   753   772   777   793   798   822   846
MDLEN            000A N   515#
MDUL             0058 V  2070  2074  2085  2095  2328#
MEM              0003 N    89#  217   218   219   220   221   501   586   715   959
                        1452  1526  1837  2055  2488  2489  2490  2491  2492  2493
                        2530  2531  2532
MEMCNT           0640 V   675   677   690  2750#
MEMMOD           0010 N  2271#
MEMMODBIT        0004 N    99#
MEMSPB           0641 V   672   694  2393  2753#
MFCODE           0004 N   554#
MFL              005A V  1141  1316  1448  2329#
MFLI             043E L  1309# 1310
MFLOAD           0001 N   552#  891
MFLRET           0459 L  1302  1303  1319#
MFPBLEN          0004 N   568#
MFPPD            0002 N   567#  568  1345
MFPBSTART        0000 N   566#  567  1346
MFREECHKPD       047A L  1350  1353#
MFREEENTRY       045F L   637  1323#
MFREEERR         04E7 L  1385  1397# 1424
MFREEEXIT        04F2 L  1395# 1430  1439
MFREEGI          0499 L  1352  1380#
MFREEGUTPD       04AC L  1364  1366  1373#
MFREELIT         04BE L  1386  1392  1401#
MFREENEXT        04BC L  1383# 1387  1390  1394
MFREENMAL        0538 L  1442# 1443
MFREENXTPD       048A L  1362# 1365
MFREEOFF         0521 L  1425  1432#
MFRER            051F L  1429#
MFSHARE          0002 N   553# 1641
MFUDADDLY        0040 N   558#  943   945
MLAERR           0356 L  1877  1880#
MLAFOUND         084E L  1864  1872  1875#
MLAINITMAU       0952 L  1974  1998#
```

```
MLALLOCENTRY    081E L   642  1845#
MLAMAKENAU      08FB L  1879  1963#
MLAMGETFD       0912 L  1969  1975#
MLAMGOTMD       0920 L  1979  1981#
MLAMNEXT        0934 L  1986# 1989
MLAMNOMORE      0944 L  1988  1990#
MLANEXT         0A4A L  1866  1873#
MLANMD          0829 L  1864# 1874
MLAOUT          0862 L  1882#
MLAVAR          08EE L  1957  1959#
MLAVADI         08FD L  1910  1914  1916#
MLAVADD         0899 L  1910# 1925
MLAVADDED       08E8 L  1953  1955#
MLAVALUE        0863 L  1866  1884#
MLAVCALC        0803 L  1921  1927#
MLAVCHK         08CC L  1909  1924#
MLAVCHKMIN      08BA L  1915  1919#
MLAVERR         08C5 L  1897  1902  1905  1922#
MLAVNEXT        0891 L  1899  1908# 1918
MLENGTH         0004 N   512#  513   523  1605  1904  1913  1915  1982  1987  1987
                        2004  2076
MLFENDFREE      09F6 L  2027  2031#
MLFEXIT         0930 L  2033  2036#
MLFFREECHK      09AD L  2024  2034#
MLFMMD          0995 L  2027# 2030
MLFREEENTRY     0988 L   643  2018#
MLINMS          09C8 L  2045  2047  2050#
MLINK           0000 N   510#  511   521  1151  1308  1309  1312  1312  1441  1442
                        1445  1445  1446  1864  1873  1896  1917  1970  1970  1971
                        1973  1982  1984  1989  1992  1993  1994  2028  2044  2050
                        2050  2051  2074  2075  2087
MLINSERT        0965 L  2029  2035  2039# 2049
MMGM            00D0 L   750   752#
MMP             004C V  1094  2317#
MODULEMAP       0046 V  2304#
MODULETABLE     0000 N  2262#
MPBFLAGS        0008 N   545#  547  1230  1640
MPBLEN          000A N   547#
MPBMAX          0004 N   543#  544  1079  1081  1103  1104  1228  1271  1273  1287
                        1637  1646  1908  1947
MPBMIN          0002 N   542#  543  1078  1102  1227  1634  1645  1919
MPBOK           02AC L  1080  1082#
MPBPDADR        0006 N   544#  545  1036  1039  1234
MPBSTART        0000 N   541#  542  1224  1236  1601  1617  1648  1899  1902  1905
MPLIST          0006 N   513#  514   524  1972  1984  2024  2026  2076
MPM             0000 N    66#   67  1172  2356
MSI             08BE L   677   681#
MSFLAGS         0006 N   524#  891   943   945  1231  1500  1500
MSGSPB          0BA0 V  2778  3287#
MSLENGTH        0004 N   523#  835   935  1091  1229  1389  1428  1499  1499
MSLINK          0000 N   521#  831   832   887   888   924   925  1086  1089  1120
                        1235  1236  1236  1237  1384  1432  1432  1488  1489  1513
                        1517  1518
```

```
MSKAU           0008 N   525# 1122  1129  1239  1412  1434  1501  1501  1512  1515
MSSTART         0002 N   522#  834   894   930  1196  1225  1386  1388  1411  1425
                      1427  1491  1493
MSTART          0002 N   511#  512   522  1503  1603  1608  1673  1901  1912  1914
                      1985  1985  2004  2046  2047  2075
MSYNC           0071 L   660   668#
MULTCNT         0694 V  2984#
MULTNUM         0818 V  2886#
MULTSEC         0820 V  2894#
MUNSYNC         0092 L   664   688#
MUNUSED         0008 N   514#  515   525
MURET           00AB L   692   700#
MXDTSKQD        0874 V  2577  3266# 3279
MXDISKQPB       0690 V  3277#
MXLOADQD        0194 V  2349  2577# 2585
MXLOADQPB       0180 V  2585#
NCCB            0049 V  2314#
NCMODEV         0085 V  2378#
NCNS            0047 V  2312#
NCONDEV         0083 V  2376#
NDP8087         0091 V  2387#
NET             0007 N    93# 2503  2504  2505  2506  2507  2508
NETMOD          0030 N  2288#
NETMODBIT       0040 N   103#
NFLAGS          004A V  2315#
NLST            0048 V  2313#
NLSTDEV         0084 V  2377#
NOMIN           02CB L  1097# 1102
NOTZERO         02D6 L  1096  1101#
NSLAVES         004L V  2318#
NTLOG           0804 V  2865#
OFFSETV         08C5 V  3034#
OLAPTYPE        0819 V  2889#
OPENCNT         08FC V  3072#
OPENEND         080A V  2872#
OPENMAX         0088 V  2383#
OPENROOT        0905 V  3082#
OPENVEC         0088 V  2381#
OSIF            00AC L   674   680   696   698   704#  864   908  1048  1061  1070
                      1142  1199  1263  1299  1317  1358  1369  1377  1414  1449
OSINT           00E0 N    55#   56
OSSEG           0040 V  2300#
OSVERNUM        007C V  2358# 2363#
OWNERBOOT       008C V  2384#
PABORT          0021 N   310#  311
PACKEDDCNT      08A1 V  2999#
PARAMETERSEGMENT 092B V 3131#
PARLO           080C V  2877#
PATCH           0A00 L  2090  2097#
PBCPCNT         0879 V  2944#
PCMIT           0001 N   356#
PCMCILC         0006 N   359#
PCMCILO         0080 N   361#
```

```
PCHCILS          0002 N   357#
PCMOD            0018 N   305#  306
PCMROUT          0004 N   358#
PCMRSX           0300 N   362#
PCMS             0020 N   309#  310
PCOMMODE         0022 N   311#  312
PDADDR           08A4 V  3002#
PDCNT            0890 V  2989#
PDLEN            0030 N    59#
PDSK             0012 N   300#  301
PERMSG           0960 V  3144  3150#
PF8087           8000 N   351#
PFACTIVE         0100 N   344#
PFCHILDABORT     0800 N   347#
PFCTLC           0080 N   343#
PFCTLD           0400 N   346#
PFDSKD           2000 N   349#
PFKEEP           0002 N   336#
PFMSMAL          0004 N   337#  2664
PFLAG            0006 N   297#  298
PFNOCTLS         1000 N   348#
PFNOKMD          4000 N   350#
PFPURE           0008 N   338#
PFRAW            0040 N   341#
PFRESOURCE       0020 N   340#
PFSYS            0001 N   335#  2664
PFTABLE          0010 N   339#
PFTEMPKEEP       0200 N   345#
PHYMSK           08C8 V  3036#
PHYOFF           0870 V  2936#
PHYSOF           08C7 V  3035#
PLASTADDR        0877 V  2943#
PLDSK            0014 N   302#  303
PLINK            0000 N   293#  294  1207
PLR              006E V  2346#
PLST             0024 N   312#  313
PLUSER           0015 N   303#  304
PMEM             0016 N   304#  305   831   887   924  1086  1194  1235  1381  1488
                        1511
PNAME            0008 N   296#  299
PNAMSIZ          0008 N    51#   52    53   299
PPARENT          001E N   308#  309
PPRET            002C N   313#  314
PPRTOP           0005 N   296#  297
PRFCS            0A5D V  3203#
PRFCS1           0A63 V  3205#
PRFX             0A48 V  3199#
PRFX1            0A5A V  3202#
PRVPDS           08F9 V  3069#
PSCTRMALT        0009 N   330#
PSCRATCH         002E N   314#  315
PSDELAY          0002 N   323#
PSDQ             0006 N   327#
```

```
PSFLAGWAIT      0006 N   329#
PSHQ            0007 N   328#
PSPOLL          0001 N   322#
PSRUN           0000 N   321# 2662
PSSLEEP         0005 N   326#
PSSWAP          0003 N   324#
PSSYNC          000A N   331#
PSTAT           0004 N   295#  296
PSTERM          0004 N   325#
PTHREAD         0002 N   294#  295  1051  1053  1186  1361  1363
PTRCNT          0019 N   306#  307
PUDA            0010 N   299#  300   923
PUL             005C V  1206  1208  2330#
PUSER           0015 N   301#  302
PWAIT           001A N   307#  308
PWMODE          08DD V  3048#
PWSCRTCH        002E N   315#
QHUF            001A N   409#  410
QOLEN           001C N   410#
QOQ             0012 N   405#  406
QFDEV           0040 N   421#
QFHIDE          0004 N   416#
QFKEEP          0002 N   415# 2579  3268
QFLAGS          0004 N   401#  402
QFMX            0001 N   414# 2579  3268
QFRPL           0020 N   420#
QFRSP           0008 N   418#
QFTABLE         0010 N   419#
QLINK           0000 N   398#  399
QLK             0074 V  2349#
QMAU            0060 V  2333#
QMSGCNT         0016 N   407#  408
QMSGLEN         000E N   403#  404
QMSGOUT         0018 N   408#  409
QNAME           0006 N   402#  403
QNAMSIZ         0008 N    52#  403   448
QNET            0002 N   399#  400
QNMSGS          0010 N   404#  405
QNQ             0014 N   406#  407
QORG            0003 N   400#  401
QPBBUFPTR       0006 N   446#  447
QPBFLGS         0000 N   442#  443
QPBLEN          0010 N   448#
QPBNAME         0008 N   447#  448
QPLNET          0001 N   443#  444
QPBNMSGS        0004 N   445#  446
QPBQADDR        0002 N   444#  445
QSPB            0649 V  2753  2762#
QUL             005E V  2331#
RCOUNT          0313 V  3105#
RDDIRFLAG       0310 V  2881#
RELOG           0321 V  2895#
REPLACEREST     03F0 L  1275  1281#
```

```
REQUIREDIABLE   08DF V  3054#
RESEL           080F V  2880#
RETURNSEC       092D V  3132#
RLOG            0800 V  2863#
RLR             0068 V   330   895   922  1038  1042  1056  1351  1366  1483  1485
                       2343#
RMF             090D V  3094#
RODMSG          0969 V  3144  3152#
RODSK           0806 V  2869#
ROFMSG          0978 V  3144  3155#
ROOTEC59        0873 V  2938#
RSPSEG          0042 V  2301#
RTM             0002 N    88#  205   206   207   208   209   210   211   212   213
                       2426  2533  2534  2535  2536  2537  2538  2539  2540  2541
                       2542  2543  2544  2545  2546  2547  2559  2560
RTMMOD          0008 N  2267#
RTMMODBIT       0002 N    98#
RTMPDISP        003C N  2295#
RWFXS           0860 V  2925#
SAILEN          0005 N  1507  1579# 1613  1687  1726  1729  1750  1750  1769  1795
                       1807  1810  1822  1827  1901  1904  1920  1924  1948  1953
                       2006  2010  2010  2011
SAILENGTH       0002 N  1576# 1577  1621  1635  1644  1701  1729  1730  1734  1755
                       1759  1764  1765  1773  1785  1812  1832  2012
SAINALL         0004 N  1509  1577# 1579  1616  1639  1649  1693  1709  1732  1753
                       1762  1776  1797  1814  1833  2012
SAISTART        0000 N  1508  1575# 1576  1615  1619  1623  1624  1647  1689  1700
                       1731  1732  1751  1752  1755  1756  1758  1758  1773  1774
                       1784  1784  1785  1796  1813  1825  1831  2011
SAVEFCB         0864 V  3264#
SAVEMOD         08DC V  3047#
SAVESP          0836 V  3257#
SAVEXFCB        08D8 V  3046#
SAVRET          0414 L  1290# 1292
SCFXS           086A V  2930#
SCL             0070 V  1178  1187  2347#
SDCNT           0909 V  3087#
SDCNT0          0908 V  3088#
SEARCHA         0883 V  2959#
SEARCHABASE     088D V  2977#
SEARCHADEST     088B V  2976#
SEARCHL         088A V  2972#
SEARCHUSER0     0812 V  2883#
SEC             055B L  1472  1474#
SECTOR          087D V  2951#
SECTPT          0898 V  3026# 3038
SEERR           05EC L  1490  1521#
SEERR1          05E6 L  1506  1520#
SEERR2          05FF L  1473  1522#
SEGE            05F3 L  1486  1519  1523#
SEGO            0587 L  1499# 1491
SELDSK          087F V  2955#
SELINK          05E0 L  1514  1515  1517#
```

```
SELMSG          0087 V  3144  3158#
SEM             0500 L  1513# 1516
SEN             0508 L  1506# 1508
SER             0577 L  1432  1484#
SERIAL          003C V   617#
SES             0580 L  1484  1486#
SETPUFLAG       0900 V  3075#
SHARLENTRY      0551 L   639  1460#
SHELLRK         092e V  3126#
SHELLST         0924 V  3125#
SINGLE          0912 V  3103#
SLR             0096 V  2393#
SPGOTO          0000 N   529#  530  1477
SPGRTD          0002 N   530#  531  1478
SPGSTART        0004 N   531# 1479
SPSAVE          0e3A V  2815# 3259#
SRCHDISK        004b V  2316#
SS              SREG V   863   905  1197  1297  1413
SSSAVE          0e38 V  2814# 3250#
SUP             0001 N    97#  200   201  2429  2430  2433  2434  2438  2471  2478
                        2481  2485  2494  2496  2497  2498  2499  2521  2522  2553
                        2554  2555  2557  2558  2569
SUPERVISOR      0008 N   601#  706
SUPMOD          0000 N  2263#
SUPSUBSIT       0001 N    97#
SYSDAT          0006 V   600# 1111  1143  1257  1416  1654  1714  2014  2231
SYSENT          0080 V  2426#
TEMPDISK        0050 V  2320#
THRDRT          0072 V  1051  1361  2346#
THRDSPD         0659 V  1046  1059  1068  1356  1367  1375  2770  2778#
TICKSPERSEC     0051 V  2321#
TLOG            0802 V  2864#
TOD             007E V  2368# 3093#
TODDAY          007e V  2369#
TODMK           0080 V  2370#
TODLEN          0005 N    60#
TODMIN          0081 V  2371#
TODSEC          0082 V  2372#
TRACK           0878 V  2950#
TRUE            FFFF N    46# 2804
TRYSEG          016E L   889   891#
UB0087LEN       015E N    58#
UA0             0827 V  2900  2903#
UA1             082D V  2903  2905#
UA2             0833 V  2905  2907#
UA3             0839 V  2907  2909#
UA4             083F V  2909  2911#
UA5             0845 V  2911  2913#
UALROOT         0825 V  2900#
UAX             0020 V  2173#
UBP             002C V  2179#
UBX             0022 V  2174#
UCONCCB         0062 V  2202#
```

```
UCX                0024 V  2175#
UDAOVLLEN          0019 N  2166#
UDASAVE            0029 V  3130#
UDBLK              000E V  2161#
UDCNT              000C V  2160#
UDERRCCS           005E V  2199#
UDLBUGAP           005C V  2198#
UDLEN              0066 V  2204#
UDPASSWORD         0012 V  2164#
UDS                0028 V  2177#
UDSNSYS            0060 N  2146#  2691
UDSROFST           0002 V  2154#  2166
UDSSEG             0004 V  2155#
UDPARAM            0000 V  2151#
UDSSAV             0032 V  2182#
UDX                0026 V  2176#
UERRORMODE         0010 V  2162#
UESSAV             004C V  2189#
UFLAGSAV           004E V  2190#
UFUNC              0006 V  2156#
UINIAT             001B V  2170#
UINTICS            0050 V  2192#
UINTIDS            0052 V  2193#
UINTIES            0054 V  2194#
UINTISS            0056 V  2195#
UINSYS             0060 V  2200#
UIVECTORS          0038 V  2186#
UIVECTORS2         0044 V  2198#
ULEN               0100 N    57#   58   949  2689
ULSTCCB            0064 V  2203#
UMULTCNT           0011 V  2163#
UNKNOWN            0000 N    48#  2354  2667
UOSCS              005A V  2197#
UOSIP              0058 V  2196#
UPDCNT             001A V  2165#
URETSEG            0030 V  2181#
USEARCHA           0008 V  2158#
USEARCHBASE        000A V  2159#
USEARCHL           0007 V  2157#
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  3121#
USI                002A V  2178#
USP                001C V  2171#
USRCODE            0880 V  2956#
USS                001E V  2172#
USTACKSP           0034 V  2183#
USTACKSS           0036 V  2185#
USTATSAV           0061 V  2201#
UUSUSED            0040 V  2187#
UWRKSEG            002E V   747   752   770   776   792   797   819   845   881  1035
```

			1077	1140	1220	1256	1269	1343	1476	1598	1670	1892
			2180#									
VERSION	0078	V	2354#									
VRECORD	0015	V	3107#									
WFLAG	090E	V	3099#									
XDONE	0815	V	2886#									
XE10	0A10	V	3146	3187#								
XE11	0A29	V	3146	3192#								
XE3	0995	V	3146	3162#								
XE5	09B3	V	3146	3169#								
XE6	09C7	V	3146	3173#								
XE7	09DC	V	3146	3177#								
XE8	09EB	V	3146	3180#								
XE9	09FF	V	3146	3184#								
XERRLIST	0946	V	3144#									
XFCBREADONLY	0A9F	V	2994#									
XIOS	0006	N	92#									
XIOSIF	0032	L	709#									
XIOSMOD	0028	N	711	2284#								
XIOSMODBIT	0020	N	102#									
YESMEM	0355	L	1164	1219#								
ZEROLENGTH	0015	N	2893#									