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


                ;*******************************************************
                ;*                                                     *
                ;*      RTM - MP/M-86 or CCP/M Real Time Monitor       *
                ;*                                                     *
                ;*******************************************************
```

CCP/M-86 2.0 V.1
COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CC-104

```
                        eject 1 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
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

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

                        ; Bits that represent present modules
                        ; in module_map
0001            supmod_bit      equ     001h
0002            rtmmod_bit      equ     002h
0004            memmod_bit      equ     004h
0008            bdosmod_bit     equ     008h
0010            ciomod_bit      equ     010h
0020            xiosmod_bit     equ     020h
0040            netmod_bit      equ     040h

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

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        ;f_fuelete      equ     (user * 0100h) + 19
                        f_freadseq      equ     (user * 0100h) + 20
                        ;f_fwriteseq    equ     (user * 0100h) + 21
                        ;f_fmake        equ     (user * 0100h) + 22
                        ;f_frename      equ     (user * 0100h) + 23
                        ;f_loginvector  equ     (user * 0100h) + 24
                        f_getdefdisk    equ     (user * 0100h) + 25
                        f_setdma        equ     (user * 0100h) + 26
                        ;f_getallocvec  equ     (user * 0100h) + 27
                        ;f_writeprotect equ     (user * 0100h) + 28
                        ;f_getrovector  equ     (user * 0100h) + 29
                        ;f_setfileattr  equ     (user * 0100h) + 30
                        ;f_getdpb       equ     (user * 0100h) + 31
                        f_usercode      equ     (user * 0100h) + 32
                        f_freadrdm      equ     (user * 0100h) + 33
                        ;f_fwriterdm    equ     (user * 0100h) + 34
                        ;f_filesize     equ     (user * 0100h) + 35
                        f_setrndrec     equ     (user * 0100h) + 36
                        ;f_resetdrive   equ     (user * 0100h) + 37
                        ;f_accessdrive  equ     (user * 0100h) + 38
                        ;f_freedrive    equ     (user * 0100h) + 39
                        ;f_writerndzero equ     (user * 0100h) + 40
                        ;f_chain        equ     (user * 0100h) + 47
                        f_callbios      equ     (user * 0100h) + 50
                        f_setdmab       equ     (user * 0100h) + 51
                        ;f_getdma       equ     (user * 0100h) + 52
                        ;f_getmaxmem    equ     (user * 0100h) + 53
                        ;f_getabsmaxmem equ     (user * 0100h) + 54
                        ;f_allocmem     equ     (user * 0100h) + 55
                        ;f_allocabsmem  equ     (user * 0100h) + 56
                        f_freemem       equ     (user * 0100h) + 57
                        f_freeall       equ     (user * 0100h) + 58
                        ;f_userload     equ     (user * 0100h) + 59
                        f_delim         equ     (user * 0100h) + 110
                        f_malloc        equ     (user * 0100h) + 128
                        f_memfree       equ     (user * 0100h) + 130
                        ;f_polldev      equ     (user * 0100h) + 131
                        f_flagwait      equ     (user * 0100h) + 132
                        f_flagset       equ     (user * 0100h) + 133
                        f_qmake         equ     (user * 0100h) + 134
                        f_qopen         equ     (user * 0100h) + 135
                        ;f_qdelete      equ     (user * 0100h) + 136
                        f_qread         equ     (user * 0100h) + 137
                        f_cqread        equ     (user * 0100h) + 138
                        f_qwrite        equ     (user * 0100h) + 139
                        f_cqwrite       equ     (user * 0100h) + 140
                        ;f_delay        equ     (user * 0100h) + 141
                        f_dispatch      equ     (user * 0100h) + 142
                        f_terminate     equ     (user * 0100h) + 143
                        f_createproc    equ     (user * 0100h) + 144
                        f_setprior      equ     (user * 0100h) + 145
                        f_conattach     equ     (user * 0100h) + 146
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                                f_condetach     equ     (user * 0100h) + 147
                                ;f_setdefcon    equ     (user * 0100h) + 148
                                f_conassign     equ     (user * 0100h) + 149
                                ;f_clicmd       equ     (user * 0100h) + 150
                                ;f_callrsp      equ     (user * 0100h) + 151
                                f_parsefilename equ     (user * 0100h) + 152
                                ;f_getdefcon    equ     (user * 0100h) + 153
                                ;f_sdataddr     equ     (user * 0100h) + 154
                                ;f_timeofday    equ     (user * 0100h) + 155
                                ;f_pdaddress    equ     (user * 0100h) + 156
                                ;f_abortprocess equ     (user * 0100h) + 157
                                f_lstattach     equ     (user * 0100h) + 158
                                f_lstdetach     equ     (user * 0100h) + 159
                                ;f_setdeflst    equ     (user * 0100h) + 160
                                ;f_clstattch    equ     (user * 0100h) + 161
                                f_cconattcn     equ     (user * 0100h) + 162
                                ;f_osvernum     equ     (user * 0100h) + 163
                                ;f_getdeflst    equ     (user * 0100h) + 164


                                        ; Internal SUP functions
                                f_load          equ     (sup * 0100h) + 10
                                f_cload         equ     (sup * 0100h) + 14


                                        ; Internal RIM functions
                                f_inflagset     equ     (rtm * 0100h) + 5
                                f_sleep         equ     (rtm * 0100h) + 18
                                f_wakeup        equ     (rtm * 0100h) + 19
                                f_findpdname    equ     (rtm * 0100h) + 20
                                f_sync          equ     (rtm * 0100h) + 21
                                f_unsync        equ     (rtm * 0100h) + 22
                                f_no_abort      equ     (rtm * 0100h) + 23
                                f_ok_abort      equ     (rtm * 0100h) + 24
                                f_no_abort_spec equ     (rtm * 0100h) + 25


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


                                        ; Internal CIO functions
                                f_conprint      equ     (cio * 0100h) + 14
                                f_cioterm       equ     (cio * 0100h) + 23
                                f_ciostat       equ     (cio * 0100h) + 24
                                f_rconin        equ     (cio * 0100h) + 2
                                f_rconout       equ     (cio * 0100h) + 3
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

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

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        eject ! include modtab.def
                ;*******************************************************
                ;*
                ;*      Definition of Module Table Entry
                ;*
                ;*******************************************************

0000            mod_entry       equ     0
0004            mod_init        equ     mod_entry + dword
0008            modlen          equ     mod_init + dword
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

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

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

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

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        eject | include flg.def
                    ;*******************************************************
                    ;*
                    ;*      Format of Flag Table Entry
                    ;*
                    ;*******************************************************
                    
0000                flg_pd          equ     word ptr 0
0002                flg_ignore      equ     byte ptr (flg_pd + word)
0003                flglen          equ     flg_ignore + byte
                    
FFFF                flag_off        equ     0ffffh          ;flag not set
FFFE                flag_on         equ     0fffeh          ;flag set
00FF                flag_zig        equ     0ffh            ;normal value
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        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
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        eject ! include mpb.def
                ;*******************************************************
                ;*
                ;*      Memory Parameter Block
                ;*
                ;*******************************************************
                
0000            mpb_start       equ     word ptr 0
0002            mpb_min         equu    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_udaonly      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

                ;*******************************************************
                ;*
                ;*      MAJ rree 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
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        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
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

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

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
523
524  =  0080                pf_ctlc          equ     00080h  ; abort pending
525  =  0100                pf_active        equ     00100h  ; active tty
526  =  0200                pf_tempkeep      equ     00200h  ; don't terminate yet...
527  =  0400                pf_ctld          equ     00400h  ; explicit detach occured
528  =  0800                pf_childabort    equ     00800h  ; child terminated abnormally
529  =  1000                pf_noctls        equ     01000h  ; control S not allowed
530  =  2000                pf_dskld         equ     02000h  ; process was loaded from disk
531  =  4000                pf_nokbd         equ     04000h  ; ignore keyboard
532  =  8000                pf_8087          equ     08000h  ; process uses 8087
533  =                      ;
534  =                      ;       Process descriptor pcm_flag bit values
535  =                      ;               (console mode set by function 109)
536  =                      ;
537  =  0001                pcm_11           equ     00001h  ; control C status for function 11
538  =  0002                pcm_ctls         equ     00002h  ; disable control S-Q
539  =  0004                pcm_rout         equ     00004h  ; raw output, no tabs or control P
540  =  0008                pcm_ctlc         equ     00008h  ; disable control C
541  =                      ;pcm_rsrv        equ     00040h  ; used by CP/M 3.0
542  =  0080                pcm_ctlo         equ     00080h  ; disable control O
543  =  0300                pcm_rsx          equ     00300h  ; 2 bits used by RSX for status
544
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

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

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        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...
                ;*      +----+----+----+----+----+----+----+----+
                ;*   0d              ...name           | msglen  |
                ;*      +----+----+----+----+----+----+----+----+
                ;*   10 |  nmsgs  |   dq    |   nq    | msgcnt  |
                ;*      +----+----+----+----+----+----+----+----+
                ;*   18 | msgout  | buffer  |
                ;*      +----+----+----+----+
                ;*
                ;*      link    - used to link QDs is system lists
                ;*      net     - which machine in the network
                ;*      org     - origin machine in the network
                ;*      flags   - Queue Flags
                ;*      name    - Name of Queue
                ;*      msglen  - # of bytes in one message
                ;*      nmsgs   - maximum # of messages in queue
                ;*      dq      - Root of PDs waiting to read
                ;*      nq      - Root of PDs list waiting to write
                ;*      msgcnt  - # of messages currently in queue
                ;*      msgout  - next message # to read
                ;*      buf     - pointer to queue message buffer
                ;*                (for MX queues, owner of queue)
                ;*
                ;*******************************************************

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

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                qf_rsp          equ     008h    ; rsp queue
                                qf_table        equ     010h    ; from qd table
                                qf_rpl          equ     020h    ; rpl queue
                                qf_dev          equ     040h    ; device queue

                                ;*****************************************************
                                ;*
                                ;*      QPB - Queue Parameter Block
                                ;*
                                ;*      +----+----+----+----+----+----+----+----+
                                ;*  00  |flgs|net |  qaddr  |  nmsgs  | buffptr |
                                ;*      +----+----+----+----+----+----+----+----+
                                ;*  08  |                 name                    |
                                ;*      +----+----+----+----+----+----+----+----+
                                ;*
                                ;*      flgs    - unused
                                ;*      net     - unused (which machine to use)
                                ;*      qaddr   - Queue ID, address of QD
                                ;*      nmsgs   - number of messages to read/write
                                ;*      buffptr - address to read/write into/from
                                ;*      name    - name of queue (for open only)
                                ;*
                                ;*****************************************************

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

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        eject 1 include sync.def
                ;*****************************************************
                ;                                                    *
                ;               Sync Parameter Block                 *
                ;                                                    *
                ;*****************************************************

0000            sy_link         equ     word ptr 0
0002            sy_owner        equu    word ptr sy_link + 2
0004            sy_wait         equ     word ptr sy_owner + 2
0006            sy_next         equ     word ptr sy_wait + 2
0008            synclen         equ     sy_next + 2
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        eject 1 include cmdh.def
                ;*****************************************************
                ;*
                ;*      Command Header Format and Load Fixup Records
                ;*
                ;*****************************************************
                                                        ;group descriptor format
0000            ch_form         equ     byte ptr 0
0001            ch_length       equ     word ptr (ch_form + byte)
0003            ch_base         equ     word ptr (ch_length + word)
0005            ch_min          equ     word ptr (ch_base + word)
0007            ch_max          equ     word ptr (ch_min + word)
0009            chlen           equ     ch_max + word

0008            ch_entmax       equ     8                       ;max number of group
                                                                ;descriptors

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

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

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        eject 1 include apb.def
                ;*****************************************************************
                ;*
                ;*      Abort Parameter Block
                ;*
                ;*****************************************************************

0000            apb_pd          equ     word ptr 0
0002            apb_term        equ     word ptr apb_pd + word
0004            apb_cns         equ     byte ptr apb_term + word
0005            apb_net         equ     byte ptr apb_cns + byte
0006            apb_pdname      equ     byte ptr apb_net + byte
000E            apb_len         equ     apb_pdname + pnamsiz

                ;       Priority processes are set to when aborting

0020            abt_prior       equ     32

                if mpm
                        eject 1 include ccb.def
                endif
                if ccpm
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
747
748  =                      eject ! include vccb.def
749  =                ;***********************************************
750  =                ;*
751  =                ;*      Virtual Console Control Block Definition
752  =                ;*
753  =                ;*      +---------+---------+---------+---------+
754  =                ;*  00  |      attach       |       queue       |
755  =                ;*      +---------+---------+---------+---------+
756  =                ;*  04  |  flag   | startcol| column  |  nchar  |
757  =                ;*      +---------+---------+---------+---------+
758  =                ;*  08  |  mimic  | msource |   pc    |    vc   |
759  =                ;*      +---------+---------+---------+---------+
760  =                ;*  0C  |  btmp   |   rsvd  |       state       |
761  =                ;*      +---------+---------+---------+---------+
762  =                ;*  10  |     maxbufsiz     |       vinq        |
763  =                ;*      +---------+---------+---------+---------+
764  =                ;*  14  |       voutq       |       vcmxq       |
765  =                ;*      +---------+---------+---------+---------+
766  =                ;*  18  | qpbflags| qpbfill |     qpbqaddr      |
767  =                ;*      +---------+---------+---------+---------+
768  =                ;*  1C  |      qpbnmsgs     |     qpbbuffptr    |
769  =                ;*      +---------+---------+---------+---------+
770  =                ;*  20  |       qbuff       |       cosleep     |
771  =                ;*      +---------+---------+---------+---------+
772  =                ;*  24  |       usleep      |       vsleep      |
773  =                ;*      +---------+---------+---------+---------+
774  =                ;*  28  |               Reserved                |
775  =                ;*      +---------+---------+---------+---------+
776  =                ;*
777  =                ;*
778  =                ;*
779  =                ;*      attach  - current owner of device
780  =                ;*                if 0, no owner
781  =                ;*                if 0ffffh, a mimic device
782  =                ;*      queue   - linked list of PDs waiting to attach
783  =                ;*      flag    - run-time flags
784  =                ;*      startcol- used for line editing
785  =                ;*      column  - used for line editing
786  =                ;*      nchar   - 1 character read ahead for CTRL chars.
787  =                ;*      mimic   - cio dev that mimics us.
788  =                ;*                0ffh means no mimic device
789  =                ;*      msource - if attach = 0ffffh, we are a
790  =                ;*                mimic device and msource is the
791  =                ;*                device we are mimicing.
792  =                ;*      pc      - physical console number
793  =                ;*      vc      - virtual console number
794  =                ;*      btmp    - temporary line editing variable
795  =                ;*      rsvd    - unused
796  =                ;*      state   - current state of virtual console
797  =                ;*      maxbufsiz - maximum file size for buffered mode
798  =                ;*      vinq    - address of QPB for this virtual console's
799  =                ;*                input.  written by PIN, created by the VOUT
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        ;*                process associated with this virtual console
                        ;*      voutq   - address of QPB for this virtual console's
                        ;*                output. written and created by the assoc. VOUT
                        ;*      vcmxq   - MX queue for changing of virtual console state
                        ;*      qpbflags- the 1st 8 bytes of a Queue Parameter Block, for
                        ;*                queue reads and writes, used only by reentrant
                        ;*                intercept code
                        ;*      qbuff   - buffer for queue writes
                        ;*      cosleep - temporary list for processes waiting for XIOS console output
                        ;*      usleep  - user process sleeps here
                        ;*      vsleep  - vout process sleeps here
                        ;*
0000                    c_attach        equ     word ptr 0
0002                    c_queue         equ     word ptr c_attach + word
0004                    c_flag          equ     byte ptr c_queue + word
0005                    c_strtcol       equ     byte ptr c_flag + byte
0006                    c_column        equ     byte ptr c_strtcol + byte
0007                    c_nchar         equ     byte ptr c_column + byte
0008                    c_mimic         equ     byte ptr c_nchar + byte
0009                    c_msource       equ     byte ptr c_mimic + byte
000A                    c_pc            equ     byte ptr c_msource + byte
000B                    c_vc            equ     byte ptr c_pc + byte
000C                    c_btmp          equ     byte ptr c_vc + byte
000D                    c_rsvd          equ     byte ptr c_btmp + byte
000E                    c_state         equ     word ptr c_rsvd + byte
0010                    c_maxbufsiz     equ     word ptr c_state + word
0012                    c_vinq          equ     word ptr c_maxbufsiz + word
0014                    c_voutq         equ     word ptr c_vinq + word
0016                    c_vcmxq         equ     word ptr c_voutq + word
0018                    c_qpbflags      equ     byte ptr c_vcmxq + word
0019                    c_qpbfill       equ     byte ptr c_qpbflags + byte
001A                    c_qpbqaddr      equ     word ptr c_qpbfill + byte
001C                    c_qpbnmsgs      equ     word ptr c_qpbqaddr + word
001E                    c_qpbbuffptr    equ     word ptr c_qpbnmsgs + word
0020                    c_qbuff         equ     word ptr c_qpbbuffptr + word
0022                    c_cosleep       equ     word ptr c_qbuff + word
0024                    c_usleep        equ     word ptr c_cosleep + word
0026                    c_vsleep        equ     word ptr c_usleep + word
0028                    ccblen          equ     c_vsleep + word + 4

                        ; Flags for c_flag

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

                        ;CCB state flags

0001                    csm_buffered    equ     00001h
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
853
854  =  0002                csm_background  equ     00002h
855  =  0004                csm_purging     equ     00004h
856  =  0008                csm_noswitch    equ     00008h
857  =  0010                csm_suspend     equ     00010h
858  =  0020                csm_abort       equ     00020h
859  =  0040                csm_filefull    equ     00040h
860  =  0080                csm_ctrlS       equ     00080h
861  =  0100                csm_ctrlQ       equ     00100h
862  =  0200                csm_ctrlP       equ     00200h
863  =
864  =                      ;LCB - list control block is first ten bytes of VCLB
865  =  000A                lcblen          equ     10
866
867                         endif
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                eject 1 include rtmif.rtm
                        ;*******************************************************
                        ;*
                        ;*      RTM Interface Routines
                        ;*
                        ;*******************************************************
                        
                                cseg
                                org 0
                        
=0000 E93F00    0042            jmp init                ;RTM initialization
=0003 E98500    008C            jmp entry               ;RTM entry point
                        
=0006 0000              sysdat          dw      0       ;SYSDAT segment
=  0008                 supervisor      equ     offset $
=0008                                   dw      2       ;SUP entry point
                        
                                org     0ch
=000C 06                dev_ver         db      6       ;development system data version
                                                        ;set in sysdat.fmt
                        
=000D 434F50595249                      db      'COPYRIGHT (C) 1982,'
      474854202843
      292031393832
      2C
=0020 204449474954                      db      ' DIGITAL RESEARCH '
      414C20524553
      454152434820
=0032 585858582D30                      db      'XXXX-0000-'
      30303020
=003C 363534333231      serial          db      '654321'
                        
                        ;====           ============================
                        init:           ; RTM module Initialization
                        ;====           ============================
                        
=0042 BB3900                    mov bx,(offset dispatcher)      ;init interrupt pdisp
=0045 C707CC03                  mov word ptr [bx],offset fdisp
=0049 8C4F02                    mov word ptr 2[bx],cs
=004C BB3C00                    mov bx,(offset rtm_pdisp)       ;init intermodule pdisp
=004F C7074C04                  mov word ptr [bx],offset farpdisp
=0053 8C4F02                    mov word ptr 2[bx],cs
=0056 CB                        retf
                        
                        ;*******************************************************
                        ;*
                        ;*      RTM Function Table
                        ;*
                        ;*******************************************************
                        
                                org     ((offset $)+1) AND 0fffeh       ;Word Boundary
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                
=0058 030C              function        dw      sysreset_entry  ;0
=005A A500                              dw      poll_entry      ;1
=005C A607                              dw      flag_wait_entry ;2
=005E 2907                              dw      flag_set_entry  ;3
=0060 0108                              dw      makeq_entry     ;4
=0062 57D3                              dw      openq_entry     ;5
=0064 A408                              dw      deleteq_entry   ;6
=0066 1009                              dw      readq_entry     ;7
=0068 1409                              dw      creadq_entry    ;8-conditional readq
=006A 9F09                              dw      writeq_entry    ;9
=006C A309                              dw      cwriteq_entry   ;10-conditional writeq
=006E 8500                              dw      delay_entry     ;11
=0070 C700                              dw      dispatch_entry  ;12
=0072 0B0C                              dw      terminate_entry ;13
=0074 E200                              dw      creat_proc_entry;14
=0076 CC00                              dw      set_prior_entry ;15
=0078 D800                              dw      pd_entry        ;16-get PD address
=007A 640C                              dw      abort_spec_entry;17-abort process
=007C E800                              dw      sleep_entry     ;18
=007E FF00                              dw      wakeup_entry    ;19
=0080 CD08                              dw      findpdname_entry;20
=0082 2701                              dw      sync_entry      ;21
=0084 4601                              dw      unsync_entry    ;22
=0086 9E01                              dw      no_abort_entry  ;23
=0088 C101                              dw      ok_abort_entry  ;24
=008A 6D01                              dw      no_abort_spec_entry     ;25
=                       ;               dw      flagalloc       ;26
=
=                       ;=====          ================
=                       entry:          ; RTM Entry Point
=                       ;=====          ================
=008C 32ED01E18BF1              xor ch,ch ! shl cx,1 ! mov si,cx
=0092 2EFF945800                call cs:function[si]
=0097 CB                rtm_ret:retf
=
=                       ;====           ================
=                       osif:           ; O.S. Interface
=                       ;====           ================
=0098 2EFF1E0800C3              callf cs:dword ptr .supervisor ! ret
=
=                       ;======         ================
=                       xiosif:         ; XIOS Interface
=                       ;======         ================
=009E BE0000                    mov si,mod_entry
=00A1 FF5C28C3                  callf dword ptr xiosmod[si] ! ret

```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 969
 969  =                              eject 1 include sysent.rtm
 970  =                        ;*************************************************************
 971  =                        ;*                                                           *
 972  =                        ;*                  SYSTEM ENTRY FUNCTIONS                   *
 973  =                        ;*                                                           *
 974  =                        ;*************************************************************
 975  =
 976  =                        ;==========             ==========================
 977  =                        poll_entry:             ; Poll device - DL=device
 978  =                        ;==========             ==========================
 979  =
 980  =0045 A16800                     mov ax,rlr
 981  =0048 8BD8                       mov bx,ax
 982  =004A 89571A                     mov p_wait[bx],dx
 983  =004D BA6E00                     mov dx,offset plr
 984  =0050 B301                       mov bl,ps_poll
 985  =0052 E93300          00E8       jmp sleep_entry
 986  =
 987  =                        ;===========            ====================
 988  =                        delay_entry:            ;Delay - DX = ticks
 989  =                        ;===========            ====================
 990  =
 991  =0055 881E6800                   mov bx,rlr
 992  =0059 C6470402                   mov p_stat[bx],ps_delay
 993  =005D 2689160000                 mov u_dparam,dx
 994  =0062 33DB                       xor bx,bx       ;return success after coming back into
 995  =0064 E993C3          045A       jmp dsptch      ;context from the dispatcher
 996  =
 997  =                        ;================       ==============
 998  =                        dispatch_entry:         ;Call dispatch
 999  =                        ;================       ==============
1000  =
1001  =0067 33DB                       xor bx,bx       ;return success ala DELAY_ENTRY:
1002  =0069 E98403          0450       jmp pdisp
1003  =
1004  =                        ;================       ============================
1005  =                        set_prior_entry:        ;Set Priority - DX=priority
1006  =                        ;================       ============================
1007  =
1008  =006C 881E6800                   mov bx,rlr
1009  =0070 885705                     mov p_prior[bx],dl
1010  =0073 33DB                       xor bx,bx       ;return success ala DELAY_ENTRY:
1011  =0075 E97803          0450       jmp pdisp
1012  =
1013  =                        ;========               ==================
1014  =                        pd_entry:               ;Return addr of PD
1015  =                        ;========               ==================
1016  =
1017  =0078 268C1E3000                 mov u_retseg,ds
1018  =007D 881E6800                   mov bx,rlr
1019  =0081 C3                         ret
1020  =
```

ccp/m-86 2.0 V.1
COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CC-104

```
1021
1022  =                         ;================       ============================
1023  =                         creat_proc_entry:       ;Create Process - DX->new PD
1024  =                         ;================       ============================
1025  =
1026  =00E2 E83501E96803 021A           call proc_creat ! jmp pdisp
1027  =
1028  =                         ;===========            ============================
1029  =                         sleep_entry:            ;Put Calling PD on System List
1030  =                         ;===========            ============================
1031  =                         ;       entry:  DX = list root to sleep on
1032  =                         ;               BL = sleep status passed to the dispatcher in
1033  =                         ;               the PD.P_WAIT field and becomes the PD.STATUS
1034  =                         ;               when sleeping
1035  =                         ;               interrupts are typically off to ensure
1036  =                         ;               the PD sleeps on the specified list
1037  =                         ;               before another process runs.
1038  =                         ;       exit:   AX = 0 to indicate success after return from dispatcher
1039  =                         ;               if entered with interrupts off the process will return
1040  =                         ;               from the dispatcher with them still off
1041  =
1042  =00E8 A1630088F0                 mov ax,rlr ! mov si,ax
1043  =00EB 8BC226A30000               mov ax,dx ! mov u_dparam,ax
1044  =00F3 885C2E                     mov p_scratch[si],bl
1045  =00F6 C6440405                   mov p_stat[si],ps_sleep
1046  =00FA 33DB                       xor bx,bx
1047  =00FC E95B03          045A       jmp dsptch
1048  =
1049  =                         ;============           ============================
1050  =                         wakeup_entry:           ;wakeup top PD in System List
1051  =                         ;============           ============================
1052  =                         ;       entry:  DX = List Root Address
1053  =                         ;       exit:   first PD on list is the last entry on the DRL
1054  =
1055  =                         ;       Puting the process on the end of the DRL allows the
1056  =                         ;       process to run before equal priority processes waking up
1057  =                         ;       from flag sets, i.e., interrupts.
1058  =                         ;       To work, the dispatcher must disable interrupts
1059  =                         ;       from when the last process on the DRL is placed on the
1060  =                         ;       RLR to when the process is back in context and
1061  =                         ;       turns on the interrupt flag.
1062  =
1063  =00FF 9CFA                       pushf ! cli
1064  =0101 8BDA8B37                   mov bx,dx ! mov si,[bx]         ;SI=PD to wake
1065  =0105 85F6741C        0125       test si,si ! jz wke_out         ;check for a process to wake up
1066  =0109 8B04                           mov ax,p_link[si]           ;take PD off list
1067  =010B 8907                           mov [bx],ax                 ;set list root to next PD
1068  =010D BF5C00                         mov di,offset drl - p_link  ;go to the end of DRL
1069  =0110 33C0                           xor ax,ax                   ;AX=0
1070  =                         wu_drl:
1071  =0112 3905                           cmp p_link[di],ax
1072  =0114 7404            011A           je wu_drlend
1073  =0116 8B3DEBF8        0112             mov di,p_link[di] ! jmps wu_drl
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1074
1075  =                         wu_drlend:
1076  =011A 8935                        mov p_link[di],si       ;make waking PD last on DRL
1077  =011C 8904                        mov p_link[si],ax       ;0 the end of the DRL
1078  =011E C6440400                    mov p_stat[si],ps_run   ;new status
1079  =0122 E82603        0450          call pdisp
1080  =                         wke_out:
1081  =0125 9DC3                        popf ! ret
1082  =
1083  =                         ;==========             ========================
1084  =                         sync_entry:             ;Obtain mutual exclusion
1085  =                         ;==========             ========================
1086  =                         ;       entry:  interrupts on or off
1087  =                         ;               BX = Sync Parameter Block
1088  =                         ;       exit:   ownership of Sync Parameter Block,
1089  =                         ;               interrupt state unchanged
1090  =
1091  =                         ;       Obtain ownership of mutually exclusive section of code
1092  =                         ;       without using MXqueues.  A process only sleeps temporarily
1093  =                         ;       on a SYNC structure.  A process that obtains the
1094  =                         ;       SYNC must call UNSYNC when finished with the
1095  =                         ;       protected data structure and before sleeping or
1096  =                         ;       calling SYNC with on a different SYNC structure.
1097  =
1098  =0127 A16300                      mov ax,rlr
1099  =012A 88F0                        mov si,ax               ;AX=SI calling process
1100  =012C 9CFA                        pushf ! cli
1101  =012E 874702                      xchg ax,sy_owner[bx]    ;AX=owner
1102  =0131 85C0                        test ax,ax              ;0 nobody owns it
1103  =0133 740F          0144          jz s_got_it
1104  =0135 3BC6                          cmp ax,si             ;do we already own it ?
1105  =0137 7403          0144            je s_got_it
1106  =0139 894702                          mov sy_owner[bx],ax ;restore owner
1107  =013C 8D5704                          lea dx,sy_wait[bx]  ;list to sleep on
1108  =013F B304                            mov bl,ps_sync      ;sleep with sync status
1109  =0141 E8A4FF        00E8              call sleep_entry
1110  =                                                        ;awaken with sync ownership
1111  =                         s_got_it:
1112  =0144 9DC3                        popf ! ret              ;we own the SPB
1113  =
1114  =
1115  =                         ;============           ==========================
1116  =                         unsync_entry:           ;release mutual exlclusion
1117  =                         ;============           ==========================
1118  =                         ;
1119  =                         ;       entry:  BX = Sync Parameter Block
1120  =                         ;       exit:   SYNC_OWNER changed to the PD in the SY_NEXT
1121  =                         ;               field if SY_NEXT is non 0.  The PD in the SY_NEXT
1122  =                         ;               must have a 0 P_LINK.  If SY_NEXT=0, the SY_OWNER
1123  =                         ;               field is changed to the first PD on the SY_WAIT
1124  =                         ;               list.  If SY_WAIT is 0, SY_OWNER becomes 0.
1125  =
1126  =0146 A16800                      mov ax,rlr
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1127
1128  =0149 3B4702                  cmp ax,sy_owner[bx]     ;do we own it ?
1129  =014C 7530            017E    jne us_ret
1130  =014E 33C0                      xor ax,ax
1131  =0150 9CFA                      pushf ! cli           ;no other process,can change SYNC or DRL
1132  =0152 874706                    xchg ax,sy_next[bx]   ;0 SY_NEXT field
1133  =0155 85C0                      test ax,ax            ;assigned next process ?
1134  =0157 7404            015D      jz us_wait
1135  =0159 8BF0                        mov si,ax
1136  =015B EB10            016D        jmps us_own
1137  =                         us_wait:
1138  =015D 8B7704                      mov si,sy_wait[bx]  ;give the sync to first waiting process
1139  =0160 85F6                      test si,si            ;no waiting PD ?
1140  =0162 7416            017A      jz us_zero
1141  =0164 8B04                        mov ax,p_link[si]
1142  =0166 894704                      mov sy_wait[bx],ax
1143  =0169 C7040000                    mov p_link[si],0
1144  =                         us_own:
1145  =016D C6440400                    mov p_stat[si],ps_run
1146  =0171 A15C00                      mov ax,drl
1147  =0174 8904                        mov p_link[si],ax   ;put process on DRL
1148  =0176 89365C00                    mov drl,si
1149  =                         us_zero:
1150  =017A 897702                      mov sy_owner[bx],si ;SI=0 or process to now own sync
1151  =017D 9D                          popf                ;allow interrupts
1152  =                         us_ret:                     ;wait for next dispatch
1153  =017E C3                  ret
1154  =
1155  =                         ;=================          ============================
1156  =                         assign_sync_entry:          ;give away mutual exlclusion
1157  =                         ;=================          ============================
1158  =                         ;       interrupts on
1159  =                         ;       entry:
1160  =                         ;               BX = Sync Parameter Block
1161  =                         ;               DX = address of list root, assign
1162  =                         ;                    SP8 to first PD on list
1163  =                         ;       exit:   none
1164  =                         ;
1165  =017F A16800              mov ax,rlr
1166  =0182 394702              cmp sy_owner[bx],ax     ;check that we own the resource
1167  =0185 7516            0190 jne as_ret
1168  =0187 8BF2                  mov si,dx             ;SI=list root
1169  =0189 9CFA                  pushf ! cli           ;no other process can run
1170  =018B 8B3C                  mov di,[si]           ;get first PD from list
1171  =018D 85FF                  test di,di            ;test for 0 list
1172  =018F 7408            019C  jz as_done
1173  =0191 8B05                    mov ax,p_link[di]   ;take PD off list
1174  =0193 8904                    mov [si],ax
1175  =0195 C7050000                mov p_link[di],0    ;0 link of NEXT PD
1176  =                                                 ;status is still PS_SLEEP.
1177  =                                                 ;if TEMPKEEP is on ABORT_SPEC
1178  =                                                 ;will just turn on CTLC flag.
1179  =                                                 ;if no TEMPKEEP, ABORT_SPEC
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1180
1181  =                                                 ;will not find PD on the U_DPARAM
1182  =                                                 ;list and will fail.
1183  =
1184  =0199 897F06                        mov sy_next[bx],di  ;see UN_SYNC_ENTRY
1185  =                                                 ;for use of SYNC_NEXT field
1186  =                           as_done:
1187  =019C 9D                              popf
1188  =                           as_ret:
1189  =019D C3                            ret
1190  =
1191  =                           ;===============
1192  =                           no_abort_entry:
1193  =                           ;===============
1194  =                           ;       Keep the calling process from being aborted by
1195  =                           ;       using the PF_TEMPKEEP flag.
1196  =                           ;       Turn on the PF_TEMPKEEP flag and
1197  =                           ;       increment the P_TKEEPCNT.
1198  =
1199  =                           ;       entry:  none
1200  =                           ;       exit:   PD fields altered
1201  =                           ;               interrupt state unaltered
1202  =
1203  =                           ;       This code must be exclusive of ABORT_SPEC's testing
1204  =                           ;       and acting on the PD flags.
1205  =
1206  =
1207  =019E A16300                          mov ax,rlr
1208  =01A1 8BF0                            mov si,ax
1209  =                           na_spec:
1210  =01A3 807C1900                        cmp p_tkcnt[si],0
1211  =01A7 7508                  01B4      jne na_saved
1212  =01A9 F744060002                        test p_flag[si],pf_tempkeep
1213  =01AE 7404                  01B4        jz na_saved
1214  =01B0 804C1980                            or p_tkcnt[si],80h          ;save orignal TEMPKEEP in MSB
1215  =                           na_saved:                                 ;until rest of O.S. is fixed
1216  =01B4 814C060002                      or p_flag[si],pf_tempkeep
1217  =01B9 FE4419                          inc p_tkcnt[si]
1218  =01BC C3                              ret
1219  =
1220  =                           ;===================
1221  =                           no_abort_spec_entry:
1222  =                           ;===================
1223  =                           ;       Keep the specified process from being aborted by setting
1224  =                           ;       its TEMPKEEP flag.
1225  =
1226  =                           ;       entry:  DX=PD address, interrupts off
1227  =                           ;       exit:   PD TEMPKEEP turned on, TKCNT incremented
1228  =
1229  =01BD 8BF2                            mov si,dx
1230  =01BF EBE2                  01A3      jmps na_spec
1231  =
1232  =                           ;===============
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1233
1234  =                         ok_abort_entry:
1235  =                         ;===============
1236  =                         ;       Allow the calling process to be aborted if
1237  =                         ;       the PF_TEMPKEEP flag is keeping it from being aborted.
1238  =                         ;       P_TKCNT is decremented in the PD, if P_TKCNT
1239  =                         ;       is = 0 then PF_TEMPKEEP is turned off and PF_CTLC
1240  =                         ;       flag is tested.
1241  =
1242  =                         ;       Interrupts should be turned off if an NO_ABORT_SPEC
1243  =                         ;       can be performed on a process running this code.
1244  =
1245  =                         ;       entry:  none
1246  =                         ;       exit:   PD fields altered
1247  =
1248  =01C1 A16800                     mov ax,rlr
1249  =01C4 8BF0                       mov si,ax
1250  =01C6 8A4419                     mov al,p_tkcnt[si]
1251  =01C9 247F                       and al,07FH                    ;high bit is original TEMPKEEP
1252  =01CB FEC8                       dec al
1253  =01CD 7404            01D3       jz oa_restore                  ;assume
1254  =01CF FE4C19                       dec p_tkcnt[si]              ;no borrow from MSB
1255  =01D2 C3                           ret
1256  =                         oa_restore:
1257  =01D3 F6441980                     test p_tkcnt[si],80H         ;was TEMPKEEP on originally ?
1258  =01D7 7512            01EB        jnz oa_ret                   ;if it was don't chk CTLC here
1259  =                                                               ;turn off TEMPKEEP
1260  =01D9 816406FFFD                     and p_flag[si],not pf_tempkeep
1261  =01DE F744068000                     test p_flag[si],pf_ctlc
1262  =01E3 7406            01EB         jz oa_ret
1263  =01E5 BAFFFF                        mov dx,0ffffh              ;if PF_CTLC is set ok to
1264  =01E8 E8200A          0C03          call terminate_entry       ;terminate even if SYS PD,
1265  =                                                               ;see ABT_CHK in ABORT.RTM
1266  =                                                               ;terminate may fail if
1267  =                                                               ;KEEP was set while
1268  =                         oa_ret:                               ;TEMPKEEP was also set.
1269  =01EB C6441900                   mov p_tkcnt[si],0
1270  =01EF C3                         ret
1271
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1272
1273  =                         eject ! include proc.rtm
1274  =                     ;*****************************************************
1275  =                     ;*
1276  =                     ;*      Process Routines
1277  =                     ;*
1278  =                     ;*****************************************************
1279  =
1280  =                     ;======         =========================
1281  =                     freepd:         ; Free Process Descriptor
1282  =                     ;======         =========================
1283  =                     ;       entry:  SI = PD address
1284  =
1285  =                     ; Assumes no memory and on no list except thread.
1286  =                     ; Called only from TERM_ACT in the dispatcher
1287  =
1288  =01F0 BF7000                 mov di,(offset thrdrt)-p_thread
1289  =                     freepd_npd:
1290  =01F3 8B5D02                 mov bx,p_thread[di]     ;find pd on thread
1291  =01F6 3B0E7404      01FE     cmp bx,si ! je freepd_trd
1292  =01FA 8BFBEBF5      01F3         mov di,bx ! jmps freepd_npd
1293  =                     freepd_trd:
1294  =01FE 8B4402894502           mov ax,p_thread[si] ! mov p_thread[di],ax
1295  =0204 8B4406251000           mov ax,p_flag[si] ! and ax,pf_table
1296  =020A 3D0000740A    0219     cmp ax,0 ! jz freepd_exit
1297  =020F 8B1E5C00891C               mov bx,pul ! mov p_link[si],bx
1298  =0215 89355C00               mov pul,si
1299  =                     freepd_exit:
1300  =0219 C3                     ret
1301  =
1302  =                     ;==========             ================
1303  =                     proc_creat:             ; Create Process
1304  =                     ;==========             ================
1305  =                     ;               DX = pd address in u_wrkseg
1306  =
1307  =021A 83FA007503    0222     cmp dx,0 ! jne cp_doit
1308  =021F 2BDBC3                 sub bx,bx ! ret
1309  =                     cp_doit:
1310  =0222 E85D01        0382     call getpdadr           ;si->pdaddr in rtm
1311  =0225 7304          022b     jcxz cp_gpd
1312  =0227 B8FFFFC3               mov bx,0ffffh ! ret
1313  =                     cp_gpd:
1314  =022B 8B14                   mov dx,p_link[si]
1315  =022D 5256                   push dx ! push si
1316  =                                ; init uda fields
1317  =                     cp_uda:
1318  =                                ;set Parent
1319  =
1320  =022F 8B1E6800               mov bx,rlr
1321  =0233 895C1E                 mov p_parent[si],bx
1322  =0236 C744220000             mov p_conmode[si],0         ;console mode - not inherited
1323  =                                                        ;unless RSX which aren't yet
1324  =                                                        ;implemented
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1325
1326  =023B C6441900                    mov p_tkcnt[si],0               ;temp keep count
1327  =023F 8165060FF7                  and p_flag[bx],not (pf_childabort + pf_resource)
1328  =0244 8A4424                      mov al,p_lst[si]
1329  =0247 3A0684007202 024F           cmp al,nlstdev ! jb cp_slst
1330  =024D B000                            mov al,0                    ;make sure its a legal list device
1331  =                         cp_slst:
1332  =                         if mpm
1333  =                                 add al,ncondev
1334  =                         endif
1335  =024F 884424                      mov p_lst[si],al                ;list device is relative to 0 in
1336  =                                                                 ;CCP/M
1337  =
1338  =                                     ;share memory
1339  =
1340  =0252 837F16007424 0282           cmp p_mem[bx],0 ! je cp_nm
1341  =0258 837C16007524 0282               cmp p_mem[si],0 ! jne cp_nm
1342  =025E 8D7F16                          lea di,p_mem[bx]
1343  =0261 8B3D                cp_mnxt:    mov di,ms_link[di]
1344  =0263 83FF00741A    0282              cmp di,0 ! je cp_nm
1345  =0268 57                              push di
1346  =0269 FF75025653                      push ms_start[di] ! push si ! push bx
1347  =026E B90038BD4                       mov cx,f_share ! mov dx,sp
1348  =0273 1E8CD08ED8                      push ds ! mov ax,ss ! mov ds,ax
1349  =0278 E81DFE1F      0098              call osif ! pop ds
1350  =027C 565E585F                        pop bx ! pop si ! pop ax ! pop di
1351  =0280 EBDF          0261              jmps cp_mnxt
1352  =                         cp_nm:
1353  =                                     ;set physical UDA segment
1354  =
1355  =0282 26A12E00                    mov ax,u_wrkseg
1356  =0286 8B54003D0                   mov dx,p_uda[si] ! add dx,ax
1357  =028B 895410                      mov p_uda[si],dx
1358  =
1359  =                                     ;inherit password
1360  =
1361  =028E 061E                        push es ! push ds
1362  =0290 8CC18EC28ED9                mov cx,es ! mov es,dx ! mov ds,cx
1363  =0296 BE12008BFE                  mov si,offset u_df_password ! mov di,si
1364  =029B B90400F3A5                  mov cx,4 ! rep movsw
1365  =
1366  =                                     ;set initial segment values
1367  =
1368  =02A0 8ED233D2                    mov ds,dx ! xor dx,dx
1369  =02A4 391650007503 02AD           cmp ds:u_initcs,dx ! jne cp_uda1
1370  =02AA A35000                          mov ds:u_initcs,ax
1371  =02AD 391652007503 02B6 cp_uda1:cmp ds:u_initds,dx ! jne cp_uda2      ;DX=0
1372  =02B3 A35200                          mov ds:u_initds,ax
1373  =02B6 391656007503 02BF cp_uda2:cmp ds:u_initss,dx ! jne cp_uda3      ;DX=0
1374  =02BC A35600                          mov ds:u_initss,ax
1375  =02BF 391654007503 02C8 cp_uda3:cmp ds:u_inites,dx ! jne cp_uda4      ;DX=0
1376  =02C5 A35400                          mov ds:u_initES,ax
1377  =                         cp_uda4:
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1378
1379  =
1380  =                                     ;Interrupt vector save areas:
1381  =                                     ;Get parent's interrupt vectors 0,1,3,4,224,225
1382  =                                     ;into child's UDA, if the
1383  =                                     ;interrupt vector is not initialized in the
1384  =                                     ;child's UDA.
1385  =
1386  =02C8 8ED8                    mov ds,dx                       ;DX,DS=0,ES=child's UDA
1387  =02CA 8BF2                    mov si,dx                       ;divide error vector, #0
1388  =02CC BF3800                  mov di,offset u_ivectors
1389  =02CF E89800          036A    call cp_icopy
1390  =02D2 E89500          036A    call cp_icopy                   ;single step, #1
1391  =02D5 83C60483C704            add si,4 ! add di,4             ;skip NMI, #2
1392  =02DB E88C00          036A    call cp_icopy                   ;1-byte, #3
1393  =02DE E88900          036A    call cp_icopy                   ;overflow, #4
1394  =02E1 BE8003                  mov si,offset i_os_ip
1395  =02E4 BF5800                  mov di,offset u_os_ip
1396  =02E7 E88000          036A    call cp_icopy                   ;O.S. entry, #224
1397  =02EA E87D00          036A    call cp_icopy                   ;debugger entry, #225
1398  =
1399  =                                     ; set user's stack
1400  =
1401  =02ED 061F                    push es ! pop ds
1402  =02EF A15600                  mov ax,ds:u_initss
1403  =02F2 A31E00                  mov ds:u_ss,ax
1404  =02F5 A13400                  mov ax,ds:u_stack_sp
1405  =02F8 A31C00                  mov ds:u_sp,ax
1406  =
1407  =                                     ; set other uda values
1408  =
1409  =02FB 33D2                    xor dx,dx
1410  =02FD C6056600024             mov ds:u_delim,'$'                      ;console mode init
1411  =0302 88161000                mov ds:u_error_mode,dl
1412  =0306 88161A00                mov ds:u_pd_cnt,dl
1413  =030A C606110001              mov ds:u_mult_cnt,1
1414  =030F C6061B00FF              mov ds:u_in_int,true
1415  =                             ;mov ds:u_insys,dl                      ;don't set insys
1416  =0314 8B1E5200                mov bx,ds:u_initds
1417  =0318 891E3200                mov ds:u_ds_sav,bx
1418  =031C 891E0400                mov ds:u_dma_seg,bx
1419  =0320 A15400                  mov ax,ds:u_initcs
1420  =0323 A34C00                  mov ds:u_cs_sav,ax
1421  =0326 89160000                mov ds:u_dparam,dx
1422  =032A 89164E00                mov ds:u_flag_sav,dx
1423  =032E 89165200                mov ds:u_conccb,dx
1424  =0332 89166400                mov ds:u_lstccb,dx
1425  =
1426  =                                     ; setup user stack for iret
1427  =
1428  =0336 8E1E1E00                mov ds,ds:u_ss
1429  =033A 268B3E1C00              mov di,u_sp                     ;sp->offset
1430  =033F 26A15000                mov ax,u_initcs
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1431
1432  =0343 894502                  mov 2[di],ax                    ;  ->segment
1433  =0346 C745040002              mov word ptr 4[di],0200h        ;  ->flags, interrupts on
1434  =0346 1F07                    pop ds ! pop es                 ; DS=SYSDAT, ES=UDA
1435  =
1436  =                                     ; put on thread list
1437  =
1438  =034D 5F9CFA                 pop si ! pushf ! cli            ; don't need SYNC here
1439  =0350 88167200                mov dx,thrdrt                   ; but this code cannot
1440  =0354 895402                  mov p_thread[si],dx             ; be reentrant
1441  =0357 89367200                mov thrdrt,si
1442  =
1443  =                                     ; put on dispatcher ready list
1444  =
1445  =035B 8B166C003914            mov dx,drl ! mov p_link[si],dx
1446  =0361 89366C009D              mov drl,si ! popf
1447  =
1448  =                                     ; do the next process
1449  =
1450  =0366 5AE930FE      021A      pop dx ! jmp proc_creat
1451  =
1452  =
1453  =
1454  =                         cp_icopy:
1455  =                         ;--------
1456  =                         ;       Entry:  ES:DI = UDA ivector location to store
1457  =                         ;               DS:SI = ivector to copy
1458  =                         ;               DX = 0
1459  =                         ;       Exit:
1460  =                         ;               DI,SI = incremented by 4
1461  =
1462  =036A 263915                         cmp es:word ptr 0[di],dx        ;DX=0, DI=UDA ivector to copy
1463  =036D 750C          037B              jne cp_nocopy                   ;don't copy interrupt vector
1464  =036F 26395502                          cmp es:word ptr 2[di],dx      ;if it is <> 0 in the UDA
1465  =0373 7506          037B                jne cp_nocopy
1466  =0375 B90200                              mov cx,2                    ;move two words
1467  =0378 F3A5                                rep movsw                   ;copy to UDA
1468  =037A C3                                  ret
1469  =                         cp_nocopy:
1470  =037B 83C60483C704                    add si,4 ! add di,4
1471  =0381 C3                              ret
1472  =
1473  =
1474  =                         getpdadr:
1475  =                         ;--------
1476  =                         ;       entry:  DX = PD address in U_WRKSEG
1477  =                         ;       exit:   CX = 0 successful
1478  =                         ;               SI = PD address in SYSDAT if success
1479  =                         ;               CX =
1480  =                         ;       Make sure PD address is in SYSDAT.  If not, copy into
1481  =                         ;       one in PD table.  Make address relative to SYSDAT
1482  =
1483  =0382 52                              push dx                 ;save PD in U_WRKSEG
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1484
1485  =0383 B83000                      mov bx,pdlen
1486  =0386 E8AE07        0B37          call sysdat_chk
1487  =0389 E311          039C          jcxz gpd_bad
1488  =038B 26A12E00                      mov ax,u_wrkseg       ;PD is within SYSDAT
1489  =038F 8CD3                          mov bx,ds
1490  =0391 2BC3                          sub ax,bx             ;U_WRKSEG-SYSDAT
1491  =0393 B104D3E0                      mov cl,4 | shl ax,cl  ;make into bytes
1492  =0397 5E                            pop si                ;SI relative to U_WRKSEG
1493  =0398 03F0                          add si,ax             ;SI->PD relative to SYSDAT
1494  =039A EB2D          03C9            jmps gpd_done
1495  =
1496  =                           gpd_bad:
1497  =039C 5E                            pop si                      ;U_WRKSEG:SI->PD to copy
1498  =039D 9CFA                          pushf | cli                 ;turn off interrupts while
1499  =039F 893E5C00                      mov di,pul                  ;changing the PUL
1500  =03A3 85FF                          test di,di
1501  =03A5 7505          03AC            jnz gpd_have
1502  =03A7 B90C00                        mov cx,e_no_pd              ;no free PDs left
1503  =03AA 9DC3                          popf | ret
1504  =                           gpd_have:                           ;DI->PD in SYSDAT
1505  =03AC 8B05                          mov ax,p_link[di]
1506  =03AE A35C00                        mov pul,ax                  ;update PUL
1507  =03B1 9D                            popf                        ;allow interrupts
1508  =03B2 57                            push di                     ;save address of PD in SYSDAT
1509  =03B3 06                            push es                     ;save UDA
1510  =03B4 1E                            push ds                     ;SYSDAT
1511  =03B5 268E1E2E00                    mov ds,u_wrkseg             ;DS=U_WRKSEG
1512  =03BA 07                            pop es                      ;ES=SYSDAT
1513  =03BB B91800                        mov cx,pdlen/2
1514  =03BE F3A5                          rep movsw                   ;copy PD into SYSDAT PD
1515  =03C0 061F                          push es | pop ds            ;DS=SYSDAT
1516  =03C2 07                            pop es                      ;ES=UDA
1517  =03C3 5E                            pop si                      ;SI=PD in SYSDAT
1518  =03C4 814C061000                    or p_flag[si],pf_table      ;got it from PD table
1519  =
1520  =                           gpd_done:
1521  =03C9 33C9                          xor cx,cx
1522  =03CB C3                            ret
1523
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1524
1525  =                         eject ! include dsptch.rtm
1526  =                     ;*****************************************************
1527  =                     ;*
1528  =                     ;*      Dispatch Routines
1529  =                     ;*
1530  =                     ;*****************************************************
1531  =
1532  =                     ;=====
1533  =                     fdisp:
1534  =                     ;=====
1535  =                     ; This entry point used by interrupt routines in XIOS
1536  =                     ; Note: if the XIOS is performing memory protection interrupt
1537  =                     ; handlers must enable O.S. memory before calling the O.S.
1538  =
1539  =03CC FA                        cli
1540  =03CD 1E2E8E1E0600              push ds ! mov ds,sysdat
1541  =03D3 803E2206FF74 044A         cmp indisp,true ! je nodisp             ;if indisp=true then we are
1542        70           044A
1543  =03DA A32806                        mov ax_sav,ax                       ;in the dispatcher and
1544  =03DD B0FFA22206                    mov al,true ! mov indisp,al         ;this code is skipped
1545  =03E2 8CC0A32406                    mov ax,es ! mov es_sav,ax
1546  =03E7 A1680093A326                  mov ax,rlr ! xchg ax,bx ! mov bx_sav,ax
1547        06
1548  =03EE 8E4710                        mov es,p_uda[bx]
1549  =03F1 B0FF26A21B00                  mov al,true ! mov u_in_int,al
1550  =03F7 5825A33200                    pop ax ! mov u_ds_sav,ax
1551  =03FC 8A4704                        mov al,p_stat[bx]
1552  =03FF 26A26100                      mov u_stat_sav,al
1553  =0403 C6470400                      mov p_stat[bx],ps_run
1554  =0407 A12406                        mov ax,es_sav
1555  =040A 26A34C00                      mov u_es_sav,ax
1556  =
1557  =                                               ; AX in AX_SAV
1558  =                                               ; BX in BX_SAV
1559  =                                               ; ES in U_ES_SAV
1560  =                                               ; DS in U_DS_SAV
1561  =                                               ; p_stat in U_STAT_SAV
1562  =040E E96700          0478      jmp intdisp
1563  =                                   ;dispatcher will jump to here if
1564  =                                   ; u_in_int = true.
1565  =
1566  =                     int_disp_exit:                                    ;interrupts are off
1567  =
1568  =                                               ; AX in AX
1569  =                                               ; BX in BX
1570  =                                               ; ES in U_ES_SAV
1571  =                                               ; DS in U_DS_SAV
1572  =                                               ; p_stat in U_STAT_SAV
1573  =
1574  =0411 A32306                        mov ax_sav,ax
1575  =0414 8BC3A32606                    mov ax,bx ! mov bx_sav,ax
1576  =                                                                       ;check for
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1577
1578  =0419 833E6C0000075 0478          cmp dr1,0 ! jnz intdisp             ;interrupt occurence
1579        58            0478
1580  =                                                                     ;on dispatcher exit
1581  =
1582  =0420 B00026A21800                    mov al,false ! mov u_in_int,al
1583  =0426 A1630088D8                      mov ax,rlr ! mov bx,ax
1584  =042A 268A06100                       mov al,u_stat_sav
1585  =042F 884704                          mov p_stat[bx],al
1586  =0432 A1250688D8                      mov ax,bx_sav ! mov bx,ax
1587  =0437 A12806                          mov ax,ax_sav
1588  =043A C606220600                      mov indisp,false
1589  =043F 268E1E3200                      mov ds,u_ds_sav
1590  =0444 268E064C00                      mov es,u_es_sav
1591  =0449 CF                              iret
1592  =
1593  =044A 1F                  nodisp: pop ds
1594  =044B CF                          iret
1595  =
1596  =
1597  =                         ;========
1598  =                         farpdisp:
1599  =                         ;========
1600  =                         ; Intermodule pdisp (non-interrupt)
1601  =
1602  =044C E80100C8      0450          call pdisp ! retf
1603  =
1604  =                         ;=====
1605  =                         pdisp:
1606  =                         ;=====
1607  =                         ; Call dispatcher with no special action
1608  =
1609  =0450 53d81E6800                  push bx ! mov bx,rlr
1610  =0455 C647040058                  mov p_stat[bx],ps_run ! pop bx
1611  =                                 ;jmp dsptch
1612  =
1613  =                         ;======
1614  =                         dsptch:
1615  =                         ;======
1616  =                         ;       The dispatch function looks like a noop to the
1617  =                         ;       caller. All flags and registers are maintained.
1618  =                         ;       No levels of user stack is used.
1619  =                         ;       (jmp dispatch = ret)
1620  =                         ;       interrupt routines enter through fdisp.
1621  =                         ;
1622  =                         ;       Dispatch has two (2) arguments:
1623  =                         ;       1. the p_stat field of the  process  descriptor
1624  =                         ;           determines the type of  action  to  perform
1625  =                         ;           for this process.
1626  =                         ;       2. the dparam field of the uda is  an  argument
1627  =                         ;           to the action.
1628  =                         ;       The main part of the dispatch routine takes the
1629  =                         ;       currently running process off  the  Ready  list
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1630
1631  =                       ;       and jmps to a routine which will put it on some
1632  =                       ;       other list depending on the p_stat argument.
1633  =                       ;       The subsequent routine will then jump to the
1634  =                       ;       scheduler which will do polling of devices and
1635  =                       ;       move processes off the dispatch ready list onto
1636  =                       ;       the Ready list.  The Ready List is maintained
1637  =                       ;       in priority order with round-robin scheduling
1638  =                       ;       of processes with equivalent priorities.  The
1639  =                       ;       first process on the ready list will then be
1640  =                       ;       switched in.
1641  =
1642  =                               ; set indisp flag
1643  =045A 9CFA              pushf ! cli
1644  =045C 803E2206FF75 0465 cmp indisp,true ! jne dispin
1645        02           0465
1646  =0463 9DC3                  popf ! ret
1647  =0465 C6062206FF    dispin: mov indisp,true
1648  =046A 268F064E00            pop u_flag_sav
1649  =
1650  =                           ; assumming bx=RLR:
1651  =                           ; if PLR=0 and DRL=0 then
1652  =                           ;    if p_stat[bx]=PS_RUN then
1653  =                           ;      if p_link[bx]=0 or
1654  =                           ;         p_prior[p_link[bx]]<>p_prior[bx] then
1655  =                           ;           don't do dispatch
1656  =
1657  =046F A32806                mov ax_sav,ax
1658  =0472 8BC3891E2606          mov ax,bx ! mov bx_sav,bx
1659  =                   intdisp:
1660  =0478 833E6E000075 04AB     cmp plr,0 ! jne dcont                   ;if Poll list = 0 and
1661        2C           04AB
1662  =047F 833E6C000075 04AB       cmp drl,0 ! jne dcont                 ;Dsptch Ready list = 0 and
1663        25           04AB
1664  =0486 A168008BD8              mov ax,rlr ! mov bx,ax                ;(RLR can never be 0 here)
1665  =048E 807F0400751A 04AB       cmp p_stat[bx],ps_run ! jne dcont     ;our status is run and
1666  =0491 833F00740A   04A0         cmp p_link[bx],0 ! je no_disp2      ;other PD to ready to run
1667  =0496 8A4705                     mov al,p_prior[bx]                  ;with an equal priority
1668  =0499 8B1F                       mov bx,p_link[bx]                   ;THEN skip the dispatch
1669  =049B 3A4705740B   04AB          cmp al,p_prior[bx] ! je dcont
1670  =                   no_disp2:
1671  =04A0 A126068BD8                    mov ax,bx_sav ! mov bx,ax
1672  =04A5 A12806                        mov ax,ax_sav
1673  =04A8 E95C02       0707             jmp dext
1674  =
1675  =                   dcont:
1676  =04AB 268C161E0026          mov u_ss,ss ! mov u_sp,sp
1677        89261C00
1678  =04B5 2E8E160600BC          mov ss,sysdat ! mov sp,offset dsptchtos
1679        2206
1680  =04BD A126068BD8            mov ax,bx_sav ! mov bx,ax
1681  =04C2 A12806                mov ax,ax_sav
1682  =04C5 FB                    sti
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1683
1684  =04C6 FC                        cld
1685  =                                       ; save registers
1686  =                                       ; NOTE: We will use DS instead of ES
1687  =                                       ;  No segment overrides...
1688  =
1689  =04C7 061F                      push es ! pop ds
1690  =04C9 A32000                    mov ds:u_ax,ax
1691  =04CC 8BC3A32200                mov ax,bx ! mov ds:u_bx,ax
1692  =04D1 8BC1A32400                mov ax,cx ! mov ds:u_cx,ax
1693  =04D6 8BC2A32600                mov ax,dx ! mov ds:u_dx,ax
1694  =04DB 8BC7A32800                mov ax,di ! mov ds:u_di,ax
1695  =04E0 8BC6A32A00                mov ax,si ! mov ds:u_si,ax
1696  =04E5 8BC5A32C00                mov ax,bp ! mov ds:u_bp,ax
1697  =
1698  =                       ;       Save interrupt vectors 0,1,3,4 not INT 2 which is NMI
1699  =                       ;       MP/M-86, CCP/M-86 1.0 on the IBM PC saved NMI
1700  =                       ;       Block move first 2
1701  =
1702  =04EA 33DB8EDB                  xor bx,bx ! mov ds,bx
1703  =04EE BEF3BF3800                mov si,bx ! mov di,offset u_ivectors
1704  =04F3 BA0400                    mov dx,4
1705  =04F6 8BCAF3A5                  mov cx,dx ! rep movsw
1706  =04FA 8BCA03F2                  mov cx,dx ! add si,dx           ;get next 2
1707  =04FE 03FA                      add di,dx                       ;skip INT 2 location documented
1708  =0500 F3A5                      rep movsw                       ;now as reserved word in UDA
1709  =                                       ; block move osint,debugint
1710  =
1711  =0502 BEB003BF5800              mov si,offset i_os_ip ! mov di,offset u_os_ip
1712  =0506 8BCAF3A5                  mov cx,dx ! rep movsw
1713  =
1714  =                                       ; take current process off RLR.
1715  =050C 2E8E1E0600                mov ds,sysdat
1716  =                       ;                                       ;disable memory for the process
1717  =                       ;                                       ;we are taking out of context
1718  =                       ;                                       ;turn off interrupts ?
1719  =                       ;       mov bx,rlr !; lea si,p_mem-ms_link[bx]
1720  =                       ;dsp_dnxt:
1721  =                       ;       mov si,ms_link[si]
1722  =                       ;       test si,si !; jz dsp_disabled
1723  =                       ;         mov cx,ms_start[si] !; mov dx,ms_length[si]
1724  =                       ;         push si
1725  =                       ;         mov ax,io_disable !; call xiosif
1726  =                       ;         pop si
1727  =                       ;         jmps dsp_dnxt
1728  =                       ;
1729  =                       ;dsp_disabled:
1730  =
1731  =0511 90                        nop
1732  =0512 A1680088F0                mov ax,rlr ! mov si,ax
1733  =0517 8B04A36800                mov ax,p_link[si] ! mov rlr,ax
1734  =051C C7040000                  mov p_link[si],0
1735  =                                       ; We are now in NO-MAN's land
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                                        ; From now until the end of the
                                        ; switch routine, There is no
                                        ; process in context.
                                        ; SI -> PD just taken out of context.
                                        ; jump to routine for given status
=0520 32FF8A5C04D1              xor bh,bh ! mov bl,p_stat[si] ! shl bx,1
      E3
=0527 2EFFA72C05                jmp cs:dsp_table[bx]

                                        org ((offset $)+1) AND 0fffeh

=052C 0D05              dsp_table       dw      disp_act        ;00 - run
=052E 0D06                              dw      disp_act        ;01 - (nop)-poll device
=0530 5B05                              dw      delay_act       ;02 - delay
=0532 0D06                              dw      disp_act        ;03 - (nop)-swap
=0534 8205                              dw      term_act        ;04 - terminate
=0536 4205                              dw      sleep_act       ;05 - sleep
=0538 0D06                              dw      disp_act        ;06 - (nop)-dq
=053A 0D06                              dw      disp_act        ;07 - (nop)-nq
=053C 9605                              dw      flag_act        ;08 - flag wait
=053E 0D06                              dw      disp_act        ;09 - (nop)-ciowait
=0540 0D06                              dw      disp_act        ;10 - (nop)-sync

                        sleep_act:
                        ;---------
                        ; insert running process into list specified by
                        ; u_dparam and set p_stat from p_scratch
                        ; Note: we cannot sleep on the DLR since interrupts are on
                        ; here, and flag_set can change the DLR

=0542 26A10000                  mov ax,u_dparam
=0546 8BD8                      mov bx,ax
=0548 56E8CE00      061A        push si ! call insert_process
=054C 5E814C062000              pop si ! or p_flag[si],pf_resource
=0552 8A442E884404              mov al,p_scratch[si] ! mov p_stat[si],al
=0558 E9E800        0646        jmp schedule

                        delay_act:
                        ;---------
                        ; Put the running process on the Delay List.  The
                        ; delay list is built such that any process's
                        ; remaining delay time is the additive of the delay
                        ; times of all processes ahead of it plus the # of
                        ; ticks in it's own p_wait field.  At each clock tick
                        ; the p_wait field of the top process in the list
                        ; is decremented.  If it reaches zero (0), all
                        ; processes with a zero in the p_wait field are
                        ; placed on the dispatch ready list.
                        ;       input:  SI=pd address

=055B FA                        cli                             ;keep flag set from changing
                        if mpm                                  ;TICK, and changing DLR
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1789
1790  =                             push si ! mov al,io_strtclk
1791  =                             call xiosit ! pop si
1792  =                       endif
1793  =
1794  =                       if ccpm
1795  =0550 C6050C0CFF              mov tick,true
1796  =                       endif
1797  =
1798  =0561 BB6A00                 mov bx,(offset dlr)-p_link
1799  =0564 268B0E000041            mov cx,u_dparam ! inc cx
1800  =0568 83F9007501    0570      cmp cx,0 ! jne del_lp
1801  =056F 49                      dec cx
1802  =0570 8B3F            del_lp: mov di,p_link[bx]
1803  =0572 83FF00740D    0584      cmp di,0 ! je del_o
1804  =0577 8B451A                  mov ax,p_wait[di]
1805  =057A 3BC17706      0584      cmp ax,cx ! ja del_o
1806  =057E 2BC88BDFEBEC  0570          sub cx,ax ! mov bx,di ! jmps del_lp
1807  =0584 893C8937        del_o:  mov p_link[si],di ! mov p_link[bx],si
1809  =0588 894C1A                  mov p_wait[si],cx
1809  =0588 83FF007403    0593      cmp di,0 ! je del_e
1810  =0590 29491A                  sub p_wait[di],cx
1811  =0593 E9B000        0646 del_e: jmp schedule
1812  =
1813  =                       flag_act:
1814  =                       ;--------
1815  =                       ; place running process in flag table to wait
1816  =                       ; for next flag. Note, flag may have been set...
1817  =                       ;       input:  ST=pd address
1818  =                       ;               U_DPARAM=address of Flag entry
1819  =
1820  =0596 26A100008BD8            mov ax,u_dparam ! mov bx,ax
1821  =059C FA                      cli                             ;protect from flag set
1822  =059D 833FFE7409    0548      cmp flg_pd[bx],flag_on ! je gflagon
1823  =05A2 8937C7040000            mov flg_pd[bx],si ! mov p_link[si],0
1824  =05A8 E99800        0646      jmp schedule
1825  =                       gflagon:                                ; Flag set since wait check
1826  =05AB C707FFFF                mov flg_pd[bx],flag_off
1827  =05AF FB                      sti
1828  =05B0 EB5B          0600      jmps disp_act
1829  =
1830  =                       term_act:
1831  =                       ;--------
1832  =                       ; Terminate the running process, free memory, free pd, free sync
1833  =                       ; structures.  Can only be called by TERMINATE_ENTRY.
1834  =
1835  =                       ;       input:  ST=pd address
1836  =                       ;               MEM_SYNC owned by calling process
1837  =
1838  =                                       ; place PD on rlr for now.
1839  =05B2 A16800                  mov ax,rlr
1840  =05B5 8904                    mov p_link[si],ax
1841  =05B7 89366800                mov rlr,si
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1842
1843  =
1844  =                                     ; clean up consoles
1845  =05B6 B91704E8D7FF 0098               mov cx,f_cioterm ! call osif
1846  =
1847  =                                     ; clean up memory
1848  =                                     ; (we own the MXmemory queue)
1849  =                           free_nxt:
1850  =05C1 A1E90088F0                      mov ax,rlr ! mov si,ax
1851  =05C6 8E7416                          mov si,p_mem[si]
1852  =05C9 85F67418      05E5              test si,si ! jz end_free
1853  =05CD 1E33C951                            push ds ! xor cx,cx ! push cx
1854  =05D1 FF7402                              push ms_start[si]
1855  =05D4 8CD08ED88ED4                        mov ax,ss ! mov ds,ax ! mov dx,sp
1856  =05DA B9E200E8E8FF 0098                   mov cx,f_memfree ! call osif
1857  =05E0 5B591F                              pop bx ! pop cx ! pop ds
1858  =05E3 EBDC          05C1                  jmps free_nxt
1859  =                           end_free:
1860  =
1861  =                                                                 ;release any sync structures
1862  =                                                                 ;after releasing memory
1863  =05E5 C606400600                      mov mem_cnt,0               ;and MEM_CNT=1
1864  =                                                                 ;on entry, also own THRD_SYNC
1865  =                                                                 ;so it is safe to call
1866  =                                                                 ;FREEPD below,
1867  =                                                                 ;ASSIGN_SYNC cannot be called
1868  =                                                                 ;with the THRD_SPB
1869  =                                                                 ;SI=terminating process
1870  =05EA B89600                          mov bx,offset slr - sy_link ;release any sync structures
1871  =                           t_nsync:                              ;owned by terminating PD
1872  =05ED 8B1F                            mov bx,sy_link[bx]
1873  =05EF 8503                            test bx,bx                  ;end of syncs?
1874  =05F1 7407          05FA              jz t_sync_done
1875  =05F3 53                                push bx                   ;PD cannot be allowed to
1876  =05F4 E84FFB        0146                call unsync_entry       ;abort if in SY.NEXT
1877  =05F7 5B                                pop bx
1878  =05F8 EBF3          05ED                jmps t_nsync
1879  =
1880  =                           t_sync_done:
1881  =
1882  =                                     ; take off RLR
1883  =05FA 8B356800                        mov si,rlr
1884  =05FE 8B04                            mov ax,p_link[si]
1885  =0600 A36800                          mov rlr,ax
1886  =0603 C7040000                        mov p_link[si],0
1887  =
1888  =                           ;         cmp si,owner_8087           ;release 8087 if we owned it
1889  =                           ;         jne t_end ! mov owner_8087,0
1890  =                           ;t_end:
1891  =                                     ; free up PD
1892  =0607 E8E6FBE93900 01F0               call freepd ! jmp schedule
1893  =
1894  =                           disp_act:
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1895
1896  =                         ;--------
1897  =                         ;       place process on RLR
1898  =                         ;       input:  SI=pd address
1899  =
1900  =0600 C6440400                    mov p_stat[si],ps_run
1901  =0611 BB5800                      mov bx,(offset rlr)-p_link
1902  =0614 E80300E92C00 061A           call insert_process ! jmp schedule
1903  =
1904  =                         ;===============
1905  =                         insert_process:
1906  =                         ;===============
1907  =                         ;
1908  =                         ; put PD# in list ordered by priority
1909  =                         ;
1910  =                         ;       entry:  BX = list root
1911  =                         ;               SI = pd number
1912  =                         ;       exit:   SI   is preserved
1913  =                         ;               interrupt state as on entry
1914  =
1915  =061A 8B4C0681E120                 mov cx,pflag[si] ! and cx,pf_resource
1916         00
1917  =                                                                  ;if a process was waiting
1918  =0621 8B3F                ins_npd: mov di,p_link[bx]               ;on a resource, insert
1919  =0623 85FF7413      063A           test di,di ! jz ins_out         ;it ahead of equal priority
1920  =0627 8A4505                          mov al,p_prior[di]          ;process
1921  =062A 3A4405                          cmp al,p_prior[si]
1922  =062D 7708          063A              ja ins_out                  ;lowest priority first
1923  =062F 7204          0635                  jb ins_nxt              ;higher - keep going down list
1924  =0631 E302          0635                      jcxz ins_nxt        ;equal and not resource
1925  =0633 EB05          063A                          jmps ins_out    ;equal & resource
1926  =0635 8BDFE9E7FF    0621 ins_nxt:         mov bx,di ! jmp ins_npd
1927  =063A E305          0641 ins_out: jcxz ins_exit
1928  =063C 816406DFFF                    and p_flag[si],not pf_resource
1929  =                         ins_exit:
1930  =0641 893C893703                  mov p_link[si],di ! mov p_link[bx],si ! ret
1931  =
1932  =                         ;=========
1933  =                         schedule:
1934  =                         ;=========
1935  =                         ; poll all required devices and place any ready
1936  =                         ; processes on the Ready List
1937  =
1938  =                                 ;we can enable interrupts now.
1939  =                                 ;there MUST be a process on the RLR
1940  =                                 ;at this point, ie. IDLE...
1941  =0646 FB                          sti
1942  =                                 ;go through the Poll List
1943  =0647 8F6E00                      mov di,(offset plr)-p_link
1944  =                                 ;get the next PD on list.
1945  =                                 ;DI is the last one which
1946  =                                 ;has already been checked.
1947  =                         polld_another:
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1948
1949  =064A 8B35                        mov si,p_link[di]
1950  =                                       ;SI is the next PD to check
1951  =064C 85F57436 r   0686           test si,si ! jz drltorlr
1952  =                                       ;SI is valid PD, poll it.
1953  =
1954  =                                       ;If top PD on the PLR has a worse
1955  =                                       ;priority compared to top PD on the RLR,
1956  =                                       ;there is no reason to call the XIOS
1957  =                                       ;and poll the device, this time through
1958  =                                       ;the dispatcher.  We must poll on equal
1959  =                                       ;priority to keep a compute bound
1960  =                                       ;process and the CLOCK from locking
1961  =                                       ;out a polling process.
1962  =                                       ;Note, we stop polling after
1963  =                                       ;the first process that has polled
1964  =                                       ;successfully, or we get to the end of
1965  =                                       ;the PLR.  The process is placed on
1966  =                                       ;the RLR.
1967  =
1968  =0650 881E68085D8                 mov bx,rlr ! test bx,bx             ;if RLR=0: poll
1969  =0656 7404           0662         jz poll_it
1970  =0658 8A4405                       mov al,p_prior[si]                ;priority of 1st poll PD
1971  =065B 3A47057602     0662           cmp al,p_prior[bx] ! jbe poll_it  ;poll if equal or better
1972  =0660 EB24           0686           jmps drltorlr                     ;priority than head of RLR
1973  =                          poll_it:
1974  =0662 57                          push di
1975  =                          if mpm
1976  =                                 mov cx,p_wait[si]
1977  =                          endif
1978  =                          if ccpm
1979  =0663 8B541A                      mov dx,p_wait[si]
1980  =                          endif
1981  =0666 B000E833FA     009E          mov al,io_polldev ! call xiosif
1982  =066B 5F8B35                      pop di ! mov si,p_link[di]
1983  =                                       ;if AL=0, device not ready.
1984  =066E 3C007410       0682          cmp al,0 ! je polld_next
1985  =                                            ;device ready,
1986  =                                            ;move SI from PLR to RLR
1987  =0672 89048905                     mov ax,p_link[si] ! mov p_link[di],ax
1988  =0676 BB6300                       mov bx,(offset rlr)-p_link
1989  =0679 C6440400                     mov p_stat[si],ps_run
1990  =067D E89AFF         061A          call insert_process               ;got one ready to run:
1991  =0680 EB04           0686          jmps drltorlr                     ;stop polling
1992  =                                       ;p_link[SI]=next PD to check
1993  =                          polld_next:  ;SI has been checked
1994  =0682 89FEEBC4       0648          mov di,si ! jmps polld_another
1995  =
1996  =                          drltorlr:
1997  =                          ;--------
1998  =                          ; Pull all processes on the dispatch ready list and
1999  =                          ; place them on the Ready List.
2000  =
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2001
2002  =                                     ;We must disable interrupts while
2003  =                                     ;playing with DRL since interrupts
2004  =                                     ;doing a Flag_set may also.
2005  =                                     ;We must competely drain the DRL since
2006  =                                     ;it is in no particular order.
2007  =
2008  =0686 FA8B366C00                  cli ! mov si,drl                ;protect DRL from flag set
2009  =068B 85F67412        06A1        test si,si ! jz switch
2010  =068F 8B04A36C00                      mov ax,p_link[si] ! mov drl,ax
2011  =                           ;           test ax,ax                  ;is this the last PD on DRL?
2012  =                           ;             jnz drl_noi               ;yes - don't turn on interrupts
2013  =0694 FB                              sti                         ;interrupts off guarentees
2014  =                           ;drl_noi:                               ;the last DRL PD with the
2015  =0695 C6440400                        mov p_stat[si],ps_run       ;best priority will run
2016  =0699 BB5300                          mov bx,(offset rlr)-p_link  ;next and at least until
2017  =069C E879FF          061A            call insert_process         ;it turns on interrupts
2018  =069F EBE5            0686            jmps drltorlr
2019  =
2020  =                                 switch:
2021  =                                 ;------
2022  =                                 ; switch to the first process on the Ready list
2023  =
2024  =06A1 FB                              sti
2025  =06A2 8B1E6800                        mov bx,rlr
2026  =                                         ; if no next process, go back           ;
2027  =                                         ; to schedule.  Gives more immediate    ;
2028  =                                         ; response to polled and interrupt      ;
2029  =                                         ; driven devices                        ;
2030  =06A6 85DB7503        06AD            test bx,bx ! jnz switch1                    ;
2031  =06AA E999FF          0646                jmp schedule                        ;
2032  =                                 switch1:                                    ;
2033  =
2034  =                                 ;                                       ;enable memory for this process
2035  =                                 ;                                       ;turn on interrupts ?
2036  =                                 ;       mov bx,rlr !; lea si,p_mem-ms_link[bx]
2037  =                                 ;dsp_enxt:
2038  =                                 ;       mov si,ms_link[si]
2039  =                                 ;       test si,si !; jz dsp_enabled
2040  =                                 ;         mov cx,ms_start[si] !; mov dx,ms_length[si]
2041  =                                 ;         push si
2042  =                                 ;         mov ax,io_enable !; call xiosif
2043  =                                 ;         pop si
2044  =                                 ;         jmps dsp_enxt
2045  =                                 ;
2046  =                                 ;dsp_enabled:
2047  =                                 ;
2048  =                                 ;       ;Save and restore the 8087 environment if process to run
2049  =                                 ;       ;uses the 8087 and is not the owner.  Interrupts
2050  =                                 ;       ;must be on.  Code from Intel Ap. Note. 113 page 29
2051  =                                 ;
2052  =                                 ;       ;This code shouldn't be added unless interrupt windows in
2053  =                                 ;       ;switch are allowed or the 8087 restore is separated
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2054
2055  =                         ;       ;from the 8086/8088 restore.  The switch code without this
2056  =                         ;       ;commented out 8087 code,
2057  =                         ;       ;creates an interrupt window approx. 100 to 200 micro
2058  =                         ;       ;secs on 5 to 4 meg CPU.
2059  =
2060  =                         ;       ;Allowing interrupt windows in switch
2061  =                         ;       ;means we must check on leaving the dispatcher
2062  =                         ;       ;for an interrupt awakened process (DRL again <> 0)
2063  =                         ;       ;and call the
2064  =                         ;       ;dispatcher again to prevent a 16 milli second wakeup time
2065  =                         ;       ;for a PD doing a flagwait after the interrupt service routine.
2066  =                         ;       ;Calling the dispatcher at the end of the dispatch
2067  =                         ;       ;(see commented out code at end of dispatcher and at
2068  =                         ;       ;INT_DISP_EXIT:)
2069  =                         ;       ;creates contention problems between PDs waiting for a resource
2070  =                         ;       ;and PDs waking up from interrupts.  It cannot be guarenteed
2071  =                         ;       ;who will run next.  An interrupt awakened process can
2072  =                         ;       ;get a just freed resource is should have waited for.
2073  =                         ;       ;The RTM ASSIGN_SYNC_ENTRY is an untested solution
2074  =                         ;       ;for allowing interrupts in the dispatcher switch code.
2075  =
2076  =                         ;                                       ;BX=PD to run next
2077  =                         ;       test p_flag[bx],pf_8087         ;does this process use the NPX ?
2078  =                         ;       jz done8087
2079  =                         ;         cmp bx,owner_8087             ;do we already own it
2080  =                         ;         je done8087
2081  =                         ;           xchg bx,owner_8087          ;new owner, also set by terminate
2082  =                         ;           test bx,bx ! jz get8087     ;no one owns it if BX=0
2083  =                         ;             mov dx,ds                 ;save sysdat in DX
2084  =                         ;             mov ds,p_uda[bx]          ;old owner's UDA
2085  =                         ;             fnstcw ds:u_8087          ;save IEM bit status
2086  =                         ;             nop                       ;delay while 8087 busy saves control reg
2087  =                         ;             fndisi                    ;disable 8087 busy signal
2088  =                         ;             mov ax,ds:u_8087          ;get original control word
2089  =                         ;
2090  =                         ;             fsave ds:u_8087           ;save NPX context
2091  =                         ;             fwait                     ;IEM=1.wait for save to finish
2092  =                         ;             mov ds:u_8087,ax          ;save original control word
2093  =                         ;get8087:
2094  =                         ;           mov ds,dx                   ;DS=sysdat
2095  =                         ;           mov bx,rlr
2096  =                         ;           mov ds,p_uda[bx]            ;PD to run next
2097  =                         ;           frstor ds:u_8087            ;get its 8087 environment
2098  =                         ;           push ds                     ;DS=UDA
2099  =                         ;           jmps restore                ;UDA on stack
2100  =                         ;
2101  =                         ;done8087:                              ;BX=PD to run next
2102  =
2103  =06AD 8B57108EDA                   mov dx,p_uda[bx] ! mov ds,dx    ;DS=UDA
2104  =06B2 52                          push dx                         ;UDA on stack
2105  =
2106  =                                 ;Restore interrupt vectors
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2107
2108  =
2109  =                         restore:
2110  =0683 33C08EC0JBF8                xor ax,ax ! mov es,ax ! mov di,ax
2111  =0689 BE3800                      mov si,offset u_ivectors
2112  =068C BA0400                      mov dx,4
2113  =068F 8BCAF3A5                    mov cx,dx ! rep movsw           ;restore interrupt vectors 0,1
2114  =0693 8BCA03FA                    mov cx,dx ! add di,dx           ;don't touch NMI
2115  =0697 03F2                        add si,dx                       ;skip what was NMI
2116  =0699 F3A5                        rep movsw                       ;restore interupt vectors 3,4
2117  =
2118  =069B BE5800                      mov si,offset u_os_ip           ;DS=UDA
2119  =069E BF8003                      mov di,offset i_os_ip
2120  =06A1 8BCAF3A5                    mov cx,dx ! rep movsw
2121  =
2122  =                                     ; restore registers
2123  =06A5 A122008BD8                  mov ax,ds:u_bx ! mov bx,ax
2124  =06AA A1240089C8                  mov ax,ds:u_cx ! mov cx,ax
2125  =06AF A126008BD0                  mov ax,ds:u_dx ! mov dx,ax
2126  =06B4 A12A0086F0                  mov ax,ds:u_si ! mov si,ax
2127  =06B9 A1280088F8                  mov ax,ds:u_di ! mov di,ax
2128  =06BE A12C008BE8                  mov ax,ds:u_bp ! mov bp,ax
2129  =06F3 A12000                      mov ax,ds:u_ax
2130  =                                     ; restore DS and ES and stack
2131  =06F6 07                          pop es                          ;ES=UDA
2132  =06F7 FA                          cli                             ;turn interrupts off for rest
2133  =06F8 268E161E00                  mov ss,u_ss                     ;of exit
2134  =06FD 268B261C00                  mov sp,u_sp
2135  =0702 2E8E1E0600                  mov ds,sysdat
2136  =                         dext:
2137  =0707 26803E1B00FF 0712           cmp u_in_int,true ! jne dret
2138        7503         0712
2139  =070F E9FFFC       0411               jmp int_disp_exit
2140  =                         dret:
2141  =0712 26FF364E00                  push u_flag_sav
2142  =0717 C606220600                  mov indisp,false
2143  =071C 833E6C000074 0727           cmp drl,0 ! je dd_ret
2144        04           0727
2145  =0723 9D                              popf                ; someone is on DRL from interrupt during
2146  =0724 E929FD       0450               jmp pdisp           ; switch, dispatch now, no 16ms wait
2147  =                         dd_ret:
2148  =0727 9D                          popf
2149  =0728 C3                          ret
2150
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2151
2152  =                     eject 1 include flag.rtm
2153  =                ;*****************************************************
2154  =                ;*
2155  =                ;*              Flag Management
2156  =                ;*
2157  =                ;* FLAGS-flag table offset      NFLAGS-number of flags
2158  =                ;*
2159  =                ;* Format of Flag Table Entry:
2160  =                ;*      +-------------+------+
2161  =                ;*      |  flag       |ignore|
2162  =                ;*      +-------------+------+
2163  =                ;*      flag -  00000h, flag can be allocated
2164  =                ;*      flag -  0ffffh, flag is off but is allocated to some
2165  =                ;*              function.
2166  =                ;*              0fffeh, flag is set
2167  =                ;*              0xxxxh, PD that is waiting
2168  =                ;*      ignore- 0ffh, normal case
2169  =                ;*              0xxh, number of flags to ignore-1
2170  =                ;*
2171  =                ;*      GENSYS initializes the flags reserved by the system
2172  =                ;*      and the XIOS header to 0ffh's.  The rest of the
2173  =                ;*      flags are initialized to 0 by GENSYS.
2174  =                ;*
2175  =                ;*****************************************************
2176  =
2177  =                ;=============
2178  =                ;getfree_flag:
2179  =                ;=============
2180  =
2181  =                ;       input:  DX = 0 then allocate a flag
2182  =                ;               else
2183  =                ;               if DH = 0FFH then release flag number
2184  =                ;               DL (0 relative)
2185  =                ;       output:
2186  =                ;               BX = flag allocated if getting a flag
2187  =                ;                       or 0FFFFH if no flag is available
2188  =                ;               BX = 0FFFFH if attempting to release a
2189  =                ;                       no existent flag
2190  =                ;               CX = 0 no error
2191  =                ;               CX = 4 if illegal flag number
2192  =                ;               CX = 27H if no more flags to allocate
2193  =                ;
2194  =                ;               Flags reserved by CCP/M or MP/M and the XIOS
2195  =                ;               header, cannot be released.
2196  =                ;
2197  =                ;turn off interrupts
2198  =                ;
2199  =                ;       test dx,dx ! jz rf_alloc
2200  =                ;         inc dh ! jnz rf_enum
2201  =                ;           cmp dl,nrsv_flags ! jbe rf_enum
2202  =                ;             cmp dl,nflags ! jae rf_enum
2203  =                ;               xor ax,ax ! mov bx,ax ! mov cx,ax
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2204
2205  =                         ;               mov al,dl
2206  =                         ;               add al,dl ! add al,dl
2207  =                         ;               mov si,ax ! mov word ptr flags[si],0
2208  =                         ;               mov byte ptr flags 2[si],0
2209  =                         ;               ret
2210  =
2211  =                         ;rf_alloc:
2212  =                         ;       mov si,flags ! xor ax,ax
2213  =                         ;       mov bx,ax ! mov cx,nflags
2214  =                         ;rf_nxt:
2215  =                         ;       cmp ax,[si] ! je rf_foundone            ;got one
2216  =                         ;         add si,3 ! inc bx
2217  =                         ;         loop rf_nxt
2218  =                         ;         mov cx,e_noflags ! jmps rf_err
2219  =                         ;rf_foundone:
2220  =                         ;       mov cx,ax                       ;0 CX
2221  =                         ;       dec ax
2222  =                         ;       mov [si],ax                     ;set the 3 bytes to 0ffh
2223  =                         ;       mov 2[si],al
2224  =                         ;       ret
2225  =                         ;rf_enum:
2226  =                         ;       mov cx,e_illflag
2227  =                         ;rf_err:
2228  =                         ;       mov bx,0ffffh
2229  =                         ;       ret
2230  =
2231  =                         ;==============   ========================
2232  =                         flag_set_entry: ; Set Logical Interrupt Flag
2233  =                         ;==============   ========================
2234  =                         ; NOTE: the flagset routine can (must?) be called from outside
2235  =                         ;       the operating system through an interrupt
2236  =                         ;       routine.  UDA variables cannot be used.  This
2237  =                         ;       is the only function an interrupt routine can
2238  =                         ;       call.
2239  =                         ;
2240  =                         ;       input:  DL = flag number
2241  =                         ;       output: BX = 0 if okay,0ffffh if error
2242  =                         ;               CX = if error: e_flag_ovrrun
2243  =
2244  =0729 E8A700        0703         call flag_valid
2245  =072C 80F9017539    076A         cmp cl,flag_tick ! jne notick
2246  =0731 8B1E6A00                   mov bx,dlr
2247  =0735 85DB7429      0762         test bx,bx ! jz dlr_null                ;no process waiting
2248  =0739 FF4F1A7521    075F           dec p_wait[bx] ! jnz nxt_tick
2249  =                           nxt_tpd:
2250  =073C 8B3789366A00                    mov si,p_link[bx] ! mov dlr,si      ;SI,DLR=next waiting PD
2251  =0744 C6470400                        mov p_stat[bx],ps_run               ;put PD done waiting
2252  =0748 A15C008907                      mov ax,drl ! mov p_link[bx],ax      ;on DRL
2253  =074D 891E6C00                        mov drl,bx
2254  =0751 85F6740D      0762              test si,si ! jz dlr_null            ;SI=next waiting PD
2255  =0755 837C1A007504  075F                  cmp p_wait[si],0 ! jnz nxt_tick
2256  =0759 8BDEEBDF      073E                      mov bx,si ! jmps nxt_tpd;another process was waiting
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2257
2258  =                                                              ;the same number of ticks
2259  =075F E99600        07F8 nxt_tick:          jmp flag_exit      ;wait for next tick
2260  =
2261  =                        dlr_null:                            ;DLR is empty turn off
2262  =0762 C6060C0C00E9 07F8          mov tick,false ! jmp flag_exit ;XIOS tick flag
2263        8E09          07F8
2264  =
2265  =                        no_tick:
2266  =076A 807C02FF7406 0776          cmp flg_ignore[si],flag_zig ! je fs_set
2267  =0770 FE4C02E98200 07F8          dec flg_ignore[si] ! jmp flag_exit
2268  =
2269  =0776 83FBFE7506    0781 fs_set: cmp bx,flag_on ! jne fs_non
2270  =077B B90500E97B00 07FC          mov cx,e_flag_ovrrun ! jmp flag_bexit
2271  =
2272  =0781 83FBFF7507    078D fs_non: cmp bx,flag_off ! jne fs_noff
2273  =0786 C704FFFF                   mov flg_pd[si],flag_on
2274  =078A E96B00        07F8         jmp flag_exit
2275  =
2276  =078D A16C008907         fs_noff:mov ax,drl ! mov p_link[bx],ax
2277  =0792 891E6C00C647               mov drl,bx ! mov p_stat[bx],ps_run
2278        0400
2279  =079A 814F062000                 or p_flag[bx],pf_resource      ;12/4/83
2280  =079F C704FFFF                   mov flg_pd[si],flag_off
2281  =07A3 E95200        07F8         jmp flag_exit
2282  =
2283  =                        ;===============   ==============================
2284  =                        flag_wait_entry: ; Wait for Logical Interrupt Flag
2285  =                        ;===============   ==============================
2286  =                        ;       input:  DL = flag number
2287  =                        ;       output: BX = 0 if everything okay
2288  =                        ;               BX = 0ffffh if Error
2289  =                        ;               CX = Error Code:0,e_flag_underrun
2290  =
2291  =07A6 E82A00        07D3         call flag_valid
2292  =07A9 83FBFE7507    07B5         cmp bx,flag_on ! jne fw_non
2293  =07AE C704FFFF                   mov flg_pd[si],flag_off
2294  =07B2 E94300        07F8         jmp flag_exit
2295  =07B5 83FBFF7513    07CD fw_non: cmp bx,flag_off ! jne fw_noff
2296  =07BA 8B1E6800                   mov bx,rlr
2297  =07BE C6470408                   mov p_stat[bx],ps_flagwait
2298  =07C2 2689360000                 mov u_dparam,si
2299  =07C7 E890FCE92B00 045A          call dsptch ! jmp flag_exit
2300  =07CD B90600E92900 07FC fw_noff:mov cx,e_flag_underrun ! jmp flag_bexit
2301  =
2302  =                        flag_valid:     ; Check validity of flag number
2303  =                        ;----------       ----------------------------
2304  =                        ;       entry:  DL = flag number
2305  =                        ;       output: SI = ptr to flag entry
2306  =                        ;               BX = contents of flag entry
2307  =                        ;               CL = flag number
2308  =                        ;               clear interrupt flag - Flags on stack
2309  =
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2310
2311  =07D3 589C                        pop ax ! pushf                  ;AX=return address
2312  =07D5 3A1664A007206 07E1          cmp dl,nflags ! jb flag_good
2313  =                           flag_bad:
2314  =07DB B90400E91B00 07FC                mov cx,e_ill_flag ! jmp flag_bexit  ;flags and next return
2315  =                           flag_good:                                      ;address on stack
2316  =07E1 8AC8                        mov cl,dl                               ;save flag number
2317  =07E3 32F650FA                    xor dh,dh ! push ax ! cli               ;return to fset/fwait on stack
2318  =07E7 8BC2                        mov ax,dx                               ;multiply flag number
2319  =07E9 8BD8                        mov bx,ax                               ;times 3
2320  =07EB 03C0                        add ax,ax                               ;*2
2321  =07ED 03C3                        add ax,bx                               ;*3
2322  =07EF 8B36560003F0                mov si,flags ! add si,ax
2323  =07F5 881C                        mov bx,[si]
2324  =                                 ;test bx,bx     ;FLAG field cannot be 0
2325  =                                 ;jz flag_bad
2326  =07F7 C3                            ret
2327  =
2328  =                           flag_exit:      ; Successful Exit
2329  =                           ;---------        ---------------
2330  =                           ;       entry:  flags and return from flagset or flagwait on stack
2331  =
2332  =07F8 33DB9DC3                    xor bx,bx ! popf ! ret
2333  =
2334  =
2335  =                           flag_bexit:     ; Exit with Error
2336  =                           ;----------       ---------------
2337  =                           ;       entry:  flags and return from flagset or flagwait on stack
2338  =
2339  =07FC 33DB4B                      xor bx,bx ! dec bx
2340  =07FF 9DC3                        popf ! ret
2341
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2342
2343  =                       eject 1 include quel.rtm
2344  =                     ;*****************************************************
2345  =                     ;*
2346  =                     ;*      Queue Routines
2347  =                     ;*
2348  =                     ;*****************************************************
2349  =
2350  =                     ; Each queue system entry point is mutually exclusive
2351  =                     ; of the other queue system entry points.
2352  =                     ; All the routines called in QUE2.RTM are
2353  =                     ; subroutines of the entry points in QUE1.RTM and
2354  =                     ; thus are called only when the process has ownership
2355  =                     ; of the q system.
2356  =
2357  =                     ; If a process is already in the queue system another process
2358  =                     ; will wait until the first process calls Q_UNSYNC.
2359  =                     ; This allows us to keep interrupts on while using shared variables.
2360  =
2361  =                     ; To protect against terminating and potentially losing a QD,
2362  =                     ; a message or neglecting to wake up a DQing or NQing
2363  =                     ; process; a "no abort region" is entered whenever
2364  =                     ; we obtain the queue system.  In addition if we assign
2365  =                     ; the queue system to a DQing or NQing process, the new owner
2366  =                     ; of the q system is forced into a "no abort region".  This
2367  =                     ; makes the environment of the waking NQing or DQing process appear
2368  =                     ; as if it never went to sleep.
2369  =
2370  =                     ; A note for the uninitiated: DQing is short for "Decoding from a Queue"
2371  =                     ; or reading from a queue.  NQing is short for "eNcoding to a Queue"
2372  =                     ; or writing to a queue.
2373  =
2374  =
2375  =                     ;===========           ========================
2376  =                     makeq_entry:          ; Create a System Queue
2377  =                     ;===========           ========================
2378  =                     ;       input : U_WRKSEG:DX = address of QD to create
2379  =                     ;       output: BX = 0 if okay , 0ffffh if error
2380  =                     ;               CX = error Code
2381  =
2382  =0801 E83F02    0A43         call q_sync                    ;obtain q system, BX,DX preserved
2383  =
2384  =0804 E89202    0A99         call getqdaddr                 ;get QD and q buffer
2385  =0807 E303      080C         jcxz qm_gotqd
2386  =0809 E94500    0851           jmp qm_err
2387  =                         qm_gotqd:                         ;DI->QD in SYSDAT
2388  =                                                           ;make sure this queue doesn't
2389  =                                                           ;already exist
2390  =                         qm_chk:
2391  =080C 06                        push es                    ;save UDA
2392  =080D 2E8E060600                mov es,sysdat
2393  =0812 BE7400                    mov si,(offset qlr)-q_link
2394  =                         qm_nxt:
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2395
2396  =0815 8B34                    mov si,q_link[si]           ;go down QLR to see if QD already
2397  =0817 85F6741A      0835      test si,si ! jz qm_go       ;exists
2398  =081B 5756                      push di ! push si         ;save new QD and QLR ptr
2399  =081D B90400                    mov cx,qnamsiz/2
2400  =0820 83C706                    add di,q_name             ;ES:DI->QPB.NAME
2401  =0823 83C606                    add si,q_name             ;DS:SI->QPB.NAME
2402  =0826 F3A7                      repe cmpsw
2403  =0828 5E5F                      pop si ! pop di           ;restore QLR ptr and new QD
2404  =082A 75E9          0815        jne qm_nxt                ;no match try next QD
2405  =
2406  =082C 07                         pop es                 ;names match, restore UDA
2407  =082D E86B03        0B9B         call remqd             ;QD pointed to by DS:DI
2408  =0830 B90A00                     mov cx,e_q_inuse
2409  =0833 EB1C          0851         jmps qm_err
2410  =
2411  =                         qm_go:                        ;no match - alright to make this q
2412  =0835 07                        pop es                 ;ES=UDA
2413  =0836 33D2                      xor dx,dx              ;initialize the QD
2414  =0838 895512                    mov q_dq[di],dx
2415  =083B 895514                    mov q_nq[di],dx
2416  =083E 895516                    mov q_msgcnt[di],dx
2417  =0841 895518                    mov q_msgout[di],dx
2418  =0844 A17400                    mov ax,qlr             ;put QD on QLR
2419  =0847 8905                      mov q_link[di],ax
2420  =0849 893E7400                  mov qlr,di
2421  =084D 33DB                      xor bx,bx              ;return success
2422  =084F EB03          0854        jmps qm_ret
2423  =                         qm_err:
2424  =0851 BBFFFF                    mov bx,0ffffh
2425  =                         qm_ret:
2426  =0854 E9FA01        0A51        jmp q_unsync           ;release q system; BX,CX preserved
2427  =
2428  =                         ;===========            ======================
2429  =                         openq_entry:            ; Find an active Queue
2430  =                         ;===========            ======================
2431  =                         ;       input:  U_WRKSEG:DX = address of QPB
2432  =                         ;
2433  =                         ;       output: sets QPB.QADDR to QD offset in SYSDAT
2434  =                         ;               BX = 0 if okay, 0ffffh if not
2435  =                         ;               CX = error code
2436  =
2437  =0857 E8E901        0A43        call q_sync            ;obtain q system; BX,DX preserved
2438  =085A 06                        push es                ;save UDA
2439  =085B 268E062E00                mov es,u_wrkseg
2440  =0860 BE7400                    mov si,(offset qlr)-q_link
2441  =0863 8BFA                      mov di,dx              ;ES:DI->QPB
2442  =                         qo_nqd:
2443  =0865 8B34                      mov si,q_link[si]      ;go down QLR until QD name
2444  =0867 85F67506      0871        test si,si ! jnz qo_cmp ;matches QPB name or end of QLR
2445  =086B 07                         pop es                ;ES=UDA
2446  =086C B90900                     mov cx,e_no_queue     ;found end of QLR - can't open
2447  =086F EB32          0BA3         jmps qo_err
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2448
2449  =                       qo_cmp:
2450  =0871 5756                      push di ! push si               ;compare names
2451  =0873 B90400                    mov cx,qnamsiz/2
2452  =0876 83C506                    add si,q_name                   ;DS:SI->QD.NAME
2453  =0879 83C708                    add di,qpb_name                 ;ES:DI->QPB.NAME
2454  =087C F3A7                      repe cmpsw
2455  =087E 5E5F                      pop si ! pop di                 ;restore QD,QPB
2456  =0880 75E3          0865        jne qo_nqd                      ;try next QD if no match
2457  =
2458  =0882 F744040400                test q_flags[si],qf_hide        ;names match
2459  =0887 7411          089A        jz noprot                       ;check for protection:
2460  =0889 891E6800                    mov bx,rlr          ;must be system process to
2461  =088D F747060100                  test p_flag[bx],pf_sys          ;open q with QF_HIDE set
2462  =0892 7506          089A          jnz noprot
2463  =0894 07                          pop es                          ;ES=UDA
2464  =0895 B90D00                      mov cx,e_q_protected            ;QF_HIDE and not SYS PD
2465  =0898 EB09          08A3          jmps qo_err
2466  =                       noprot:
2467  =089A 26897502                    mov es:qpb_qaddr[di],si         ;write the QD offset into
2468  =089E 07                          pop es                          ;ES=UDA
2469  =089F 33DB                        xor bx,bx                       ;the QPB_QADDR field
2470  =08A1 EB03          08A6          jmps qo_ret                     ;to open the q
2471  =
2472  =                       qo_err:                                   ;CX=error code
2473  =08A3 BBFFFF                  mov bx,0ffffh
2474  =                       qo_ret:
2475  =08A6 E8A801        0A51      call q_unsync                       ;release q system; BX,CX preserved
2476  =08A9 C3                      ret
2477  =
2478  =
2479  =                       ;==============          ========================
2480  =                       deleteq_entry:          ; Delete a System Queue
2481  =                       ;==============          ========================
2482  =                       ;       Takes QD off the QLR and place it in the QUL
2483  =                       ;       input:  U_WRKSEG:DX = offset of QPB
2484  =                       ;       output: BX = 0 if ok, 0ffffh if error
2485  =                       ;               CX = error code
2486  =
2487  =08AA 8CC0                    mov ax,es                       ;save UDA
2488  =08AC 268E062E00              mov es,u_wrkseg                 ;get QD address from user's
2489  =08B1 8BFA                    mov di,dx                       ;QPB
2490  =08B3 268B7D02                mov di,es:qpb_qaddr[di]         ;DS:DI->QD (unverified)
2491  =08B7 8EC0                    mov es,ax                       ;ES=UDA
2492  =
2493  =                                                             ;Check for KEEP, SYS Flags
2494  =                                                             ;DS:DI->QD
2495  =08B9 F745040200              test q_flags[di],qf_keep
2496  =08BE 7512          08D2      jnz qd_err1
2497  =08C0 F745040400                test q_flags[di],qf_hide
2498  =08C5 7412          08D9        jz qd_ok1
2499  =08C7 8B1E6900                    mov bx,rlr                    ;if hide flag then
2500  =08CB F747060100                  test p_flag[bx],pf_sys        ;SYS flag must be on in PD
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2501
2502  =08D0 7507          08D9              jnz qd_ok1
2503  =
2504  =08D2 B90D00              qd_err1:      mov cx,e_q_protected
2505  =08D5 BBFFFF                            mov bx,0ffffh
2506  =08D8 C3                                ret
2507  =                         qd_ok1:
2508  =08D9 57                          push di                         ;DI->QD
2509  =08DA E86601        0A43          call q_sync                     ;obtain q system; BX,DX preserved
2510  =08DD 5F                          pop di
2511  =08DE 8B4512                      mov ax,q_dq[di]                 ;if any process is NQing or
2512  =08E1 8B5514                      mov dx,q_nq[di]                 ;DQing we can't delete
2513  =08E4 0BD0B90A00                  or dx,ax ! mov cx,e_q_inuse
2514  =08E9 751F          090A          jnz qd_err2
2515  =
2516  =08EB BB7400                        mov bx,(offset qlr)-q_link
2517  =                         qd_nqd:                                 ;look for QD on QLR
2518  =08EE 8B37                            mov si,q_link[bx]           ;DI=QD to delete
2519  =08F0 85F67413      0907              test si,si ! jz qd_noq
2520  =08F4 3BF77404      08FC                cmp si,di ! je qd_found
2521  =08F8 8BDEEBF2      08EE                mov bx,si ! jmps qd_nqd   ;try next queue
2522  =                         qd_found:
2523  =08FC 8B05                            mov ax,q_link[di]           ;found the queue, remove
2524  =08FE 8907                            mov q_link[bx],ax           ;it from QLR
2525  =0900 E89802        0B9B              call remqd
2526  =0903 33DB                            xor bx,bx                   ;return success
2527  =0905 EB06          090D              jmps qd_ret
2528  =
2529  =                         qd_noq:
2530  =0907 B90900                          mov cx,e_no_queue
2531  =                         qd_err2:
2532  =090A BBFFFF                          mov bx,0ffffh
2533  =                         qd_ret:
2534  =090D E94101        0A51          jmp q_unsync                    ;release q system
2535  =                                 ;ret                            ;BX,CX preserved
2536  =
2537  =                         ;============           ============
2538  =                         readq_entry:            ; Read Queue
2539  =                         ;============           ============
2540  =0910 32C0EB02      0916          xor al,al ! jmps readq
2541  =
2542  =                         ;=============
2543  =                         creadq_entry:           ; Conditional Read Queue
2544  =                         ;=============
2545  =0914 B0FF                        mov al,0ffh
2546  =                                 ; jmps readq
2547  =
2548  =                         readq:          ; Read message from queue
2549  =                         ;-----          -------------------------
2550  =                         ; If no buffer is available the process
2551  =                         ; making an unconditional READQ is placed into the DQ list.
2552  =                         ;
2553  =                         ;       Input:  U_WRKSEG:DX = QPB
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2554
2555  =                       ;               AL = 0 if unconditional
2556  =                       ;                  <> 0 if not
2557  =                       ;       output: dX = 0 if okay
2558  =                       ;                  0ffffh if error
2559  =                       ;               CX = Error Code
2560  =
2561  =0916 50                               push ax                         ;save cond code
2562  =0917 E82901      0A43                 call q_sync                     ;get q system; BX,DX preserved
2563  =091A E85A01      0A77                 call queverify                  ;is the QPB valid ?
2564  =091D 58                               pop ax                          ;AL=cond code
2565  =091E 8BF2                             mov si,dx                       ;U_WRKSEG:SI->QPB
2566  =0920 E303        0925                 jcxz qr_ver                     ;CX=0 QPB is ok
2567  =0922 E97300      0998                   jmp qr_err                    ;CX=error code from
2568  =                                qr_ver:                               ;queverify
2569  =0925 06                               push es                         ;save UDA
2570  =0926 268E062E00                       mov es,u_wrkseg                 ;ES:SI->QPB
2571  =092B 26883C02                         mov bx,es:qpb_qaddr[si]         ;BX->QD
2572  =092F 837F1600                         cmp q_msgcnt[bx],0              ;is there a msg to read ?
2573  =0933 751C        0951                 jne qr_readit
2574  =
2575  =                                qr_wait:
2576  =0935 07                                 pop es                        ;ES=UDA
2577  =0936 84C07405    093F                   test al,al ! jz qr_wait1      ;conditional read if AL <> 0
2578  =093A B90E00                               mov cx,e_q_empty
2579  =093D EB59        099d                     jmps qr_err
2580  =                                qr_wait1:                             ;ES=UDA
2581  =093F 5356                               push bx ! push si             ;QD, QPB
2582  =0941 8D5712                             lea dx,q_dq[bx]               ;DX=addr of DQ List
2583  =0944 B306                               mov bl,ps_dq                  ;sleep status=DQ
2584  =0946 E8EC00      0435                   call q_wait                   ;wait for a DQ
2585  =                                                                      ;we now own the q system
2586  =                                                                      ;and cannot be aborted
2587  =0949 5E5B                               pop si ! pop bx               ;QPB, QD
2588  =094B 06                                 push es                       ;save UDA
2589  =094C 268E062E00                         mov es,u_wrkseg
2590  =                                qr_readit:
2591  =0951 268B7C06                         mov di,es:qpb_buffptr[si]       ;ES:[DI]->user's queue buffer
2592  =0955 8B4F0E                           mov cx,q_msglen[bx]             ;ES=U_WRKSEG
2593  =0958 85C97513    096F                 test cx,cx ! jnz qr_lmsg        ;check message length
2594  =095C 33C0                               xor ax,ax
2595  =095E F747040100                         test q_flags[bx],qf_mx                 ;msglen=0, check for MX q
2596  =0963 7420        0985                   jz qr_end
2597  =0965 A16800                             mov ax,rlr                    ;its a MX queue
2598  =0968 89471A                             mov q_buf[bx],ax              ;BUF = PD addr of owner
2599  =096B 33C0                               xor ax,ax
2600  =096D EB16        0985                   jmps qr_end
2601  =                                qr_lmsg:                              ;msglen > 0
2602  =096F 8B471850                         mov ax,q_msgout[bx] ! push ax   ;MSGOUT is # of message to read
2603  =0973 F7E103471A                       mul cx ! add ax,q_buf[bx]       ;compute its start in buffer
2604  =0978 8BF0                             mov si,ax
2605  =097A F3A4                             rep movsb                       ;move to QPB buffer
2606  =097C 5840                             pop ax ! inc ax                 ;set MSGOUT to next message
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2607
2608  =097E 394710              0985          cmp ax,q_nmsgs[bx]              ;circular buffers, so
2609  =0981 7502                              jne qr_end                      ;check for wrap around
2610  =0983 33C0                                xor ax,ax
2611  =                                   qr_end:
2612  =0985 07                                pop es                          ;ES=UDA
2613  =0986 894718                            mov q_msgout[bx],ax
2614  =0989 FF4F16                            dec q_msgcnt[bx]
2615  =098C 8D5714                            lea dx,q_hq[bx]
2616  =098F E8CD00              0A5F          call q_assign_sync              ;give the queue system to
2617  =                                                                       ;first NQing process if any,
2618  =0992 E8CC00              0A51          call q_unsync                   ;release q system if we still
2619  =                                                                       ;own it, exit no abort region
2620  =0995 33DB                              xor bx,bx                       ;indicate success
2621  =0997 C3                                ret
2622  =
2623  =                                   qr_err:
2624  =0998 E8B500              0A51          call q_unsync                   ;release q system, exit no abort
2625  =                                                                       ;region, BX,CX preserved
2626  =099B BBFFFF                            mov bx,0ffffh                   ;indicate error
2627  =099E C3                                ret
2628  =
2629  =                                   ;============            =============
2630  =                                   writeq_entry:           ; Write Queue
2631  =                                   ;============            =============
2632  =099F 32C0E802            09A5          xor al,al ! jmps writeq
2633  =
2634  =                                   ;============            =========================
2635  =                                   cwriteq_entry:          ; conditional Write Queue
2636  =                                   ;============            =========================
2637  =09A3 B0FF                              mov al,0ffh
2638  =                                       ;jmps writeq
2639  =
2640  =                                   writeq:         ; Write message to queue
2641  =                                   ;------         ------------------------
2642  =                                   ; If no buffer is available when making an unconditional
2643  =                                   ; WRITEQ call, the calling process is placed into the NQ list.
2644  =                                   ;
2645  =                                   ;       input:  U_WRKSEG:DX = QPB
2646  =                                   ;               AL = 0 if unconditional
2647  =                                   ;                  <> 0 if not
2648  =                                   ;       output: BX = 0 if okay
2649  =                                   ;                  0ffffh if error
2650  =                                   ;               CX = Error Code
2651  =
2652  =09A5 50                                push ax                         ;save cond code
2653  =09A6 E89A00              0A43          call q_sync                     ;get queue system
2654  =                                                                       ;BX,DX preserved
2655  =09A9 E8CB00              0A77          call queverify                  ;is the QPB valid ?
2656  =09AC 58                                pop ax                          ;AL = cond code
2657  =09AD 8BFA                              mov di,dx                       ;U_WRKSEG:DI->QPB
2658  =09AF E303                09B4          jcxz qw_ver                     ;CX=0 QPB is ok
2659  =09B1 E97A00              0A2E            jmp qw_err                    ;CX=error code from
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2660
2661  =                       qw_ver:                         ;queverify
2662  =0984 06                        push es                 ;save UDA
2663  =0985 268E062E00                mov es,u_wrkseg         ;ES:DI->QPB
2664  =098A 268B5D02                  mov bx,es:qpb_qaddr[di] ;BX->QD
2665  =098E 8B4F16                    mov cx,q_msgcnt[bx]
2666  =0991 3B4F10                    cmp cx,q_nmsgs[bx]      ;is there a buffer to write ?
2667  =0994 751C              09E2    jne qw_writeit
2668  =
2669  =                       qw_wait:
2670  =0996 07                          pop es                ;ES=UDA
2671  =0997 84C07405          09A0      test al,al ! jz qw_wait1  ;conditional read if AL <> 0
2672  =099B B90F00                        mov cx,e_q_full
2673  =099E E85E              0A2E        jmps qw_err
2674  =                       qw_wait1:                       ;ES=UDA
2675  =09A0 53b7                        push bx ! push di     ;save QD, QPB
2676  =09A2 8D5714                      lea dx,q_nq[bx]       ;DX=addr of NQ List
2677  =09A5 B307                        mov bl,ps_nq          ;sleep status=NQ
2678  =09A7 E8B800            0A35      call q_wait           ;sleep on NQ List
2679  =                                                       ;q system is assigned to us,
2680  =                                                       ;we are in no abort region
2681  =                                                       ;by DQing process that woke us
2682  =09AA 5F5B                        pop di ! pop bx       ;QPB,QD
2683  =09AC 06                          push es               ;save UDA
2684  =09AD 268E062E00                  mov es,u_wrkseg
2685  =                       qw_writeit:
2686  =09B2 26837506                    mov si,es:qpb_buffptr[di] ;U_WRKSEG:[SI]->user's queue buffer
2687  =09B6 8B4F0E                      mov cx,q_msglen[bx]   ;ES=U_WRKSEG
2688  =09B9 85C9750E          09FB      test cx,cx ! jnz qw_lmsg  ;check message length
2689  =09BD F747040100                    test q_flags[bx],qf_mx  ;msglen=0, check for MX q
2690  =09C2 742A              0A1E        jz qw_end
2691  =09C4 33C0                            xor ax,ax
2692  =09C6 894714                          mov q_buf[bx],ax  ;its a MX queue
2693  =09C9 EB23              0A1E          jmps qw_end       ;BUF = 0
2694  =
2695  =                       qw_lmsg:                        ;msglen > 0
2696  =09CB 8B4718                      mov ax,q_msgout[bx]
2697  =09CE 034716                      add ax,q_msgcnt[bx]   ;AX = # of message to write
2698  =0A01 3B4710                      cmp ax,q_nmsgs[bx]    ;check for wrap around
2699  =0A04 7203              0A09      jb qw_move
2700  =0A06 2B4710                        sub ax,q_nmsgs[bx]
2701  =                       qw_move:
2702  =0A09 F7E103471A                  mul cx ! add ax,q_buf[bx] ;compute its start in buffer
2703  =0A0E 8BF8                        mov di,ax             ;DI offset of new msg in buffer
2704  =0A10 8CD88CC2                    mov ax,ds ! mov dx,es
2705  =0A14 8EC08EDA                    mov es,ax ! mov ds,dx ;ES=SYSDAT, DS=U_WRKSEG
2706  =0A18 F3A4                        rep movsb             ;move to QPB buffer
2707  =0A1A 8CC08ED8                    mov ax,es ! mov ds,ax ;DS=SYSDAT
2708  =                       qw_end:
2709  =0A1E 07                          pop es                ;ES=UDA
2710  =0A1F FF4716                      inc q_msgcnt[bx]      ;one more message in the queue
2711  =0A22 8D5712                      lea dx,q_dq[bx]       ;wake DQing process if any
2712  =0A25 E83700            0A5F      call q_assign_sync
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2713
2714  =0A28 E82600        0A51        call q_unsync                ;release q system if we still
2715  =0A2B 33DB                      xor bx,bx                    ;own it, exit no abort region
2716  =0A2D C3                        ret                          ;BX=0: success
2717  =
2718  =                           qw_err:
2719  =0A2E E82000        0A51        call q_unsync                ;release q system; exit no abort
2720  =                                                            ;region BX,CX preserved
2721  =0A31 BBFFFF                    mov bx,0ffffh                ;indicate error
2722  =0A34 C3                        ret
2723  =
2724  =
2725  =                       q_wait:                              ;wait on DQ or NQ list
2726  =                       ;------
2727  =                       ;       entry:  DX = list to wait on
2728  =                       ;               BL = sleep status
2729  =                       ;               calling process owns queue system through
2730  =                       ;               a call to q_sync
2731  =
2732  =                       ;       Queue message or buffer space is not available.
2733  =                       ;       Give up queue system and sleep on DQ or NQ list, but
2734  =                       ;       do not allow any other process to read or write to a queue
2735  =                       ;       until we get on the sleep list.
2736  =
2737  =
2738  =0A35 9CFA                      pushf ! cli                  ;keep abort spec from running ...
2739  =0A37 5352                      push bx ! push dx            ;save sleep status, list address
2740  =0A39 E81500        0A51        call q_unsync                ;does not go to dispatcher, we can be
2741  =0A3C 5A5B                      pop dx ! pop bx              ;aborted once on sleep list
2742  =0A3E E8A7F6        00E8        call sleep_entry             ;go to dispatcher
2743  =0A41 9D                        popf                         ;come back after q_assign call
2744  =0A42 C3                        ret                          ;Q_ASSIGN: wakes us up when resource
2745  =                                                            ;is ready
2746  =
2747  =                       q_sync:                              ;obtain ownership of q system
2748  =                       ;------
2749  =                       ;       entry:  interrupts should be on
2750  =                       ;               ES=UDA
2751  =                       ;       exit:   we own the queue system for make,open,read,write,delete
2752  =                       ;               queue operations, interrupts unchanged
2753  =                       ;               BX,DX preserved - usually entry parameters
2754  =
2755  =0A43 5352                      push bx ! push dx
2756  =0A45 E856F7        019E        call no_abort_entry          ;we cannot abort while in the queue system
2757  =0A48 BB4906                    mov bx,offset q_spb
2758  =0A4B E8D9F6        0127        call sync_entry
2759  =0A4E 5A5B                      pop dx ! pop bx
2760  =0A50 C3                        ret
2761  =
2762  =                       q_unsync:                            ;release the queue system
2763  =                       ;--------                            ;to other processes
2764  =                       ;       entry:  interrupts can be or off
2765  =                       ;       exit:   interrupts unchanged,
```

CCP/M-86 2.0 V. 1
COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CC-104

```
2766
2767  =                       ;               have not gone to dispatcher,
2768  =                       ;               bX,CX preserved - usually return codes
2769  =
2770  =0A51 5351                      push bx ! push cx
2771  =0A53 BB4906                    mov bx,offset q_spb
2772  =0A56 E8EDF6      0146          call unsync_entry       ;wait for queue system
2773  =0A59 E865F7      01C1          call ok_abort_entry     ;allow calling PD to be terminated
2774  =0A5C 5B58                      pop cx ! pop bx
2775  =0A5E C3                        ret
2776  =
2777  =                   q_assign_sync:
2778  =                   ;-------------
2779  =                   ;       entry:  DX = address of DQ or NQ list to give the queue
2780  =                   ;       exit:   none
2781  =
2782  =                   ;       Assign is used so a process is forced to read or write
2783  =                   ;       after DQing or NQing.  Otherwise an awakening DQing or NQing
2784  =                   ;       process could be deleted, leaving a message or buffer space
2785  =                   ;       available with other processes still DQing or NQing.
2786  =                   ;       This message or buffer space would never be reclaimed.
2787  =                   ;       To prevent this situation, the waking DQing or NQing process
2788  =                   ;       is kept from aborting and assigned the QSPB.  Note
2789  =                   ;       it is ok for a process to be aborted while it is on the NQ or
2790  =                   ;       DQ list before this routine is called.
2791  =
2792  =0A5F 9CFA                      pushf ! cli                     ;keep abort spec from terminating
2793  =0A61 8BDA                      mov bx,dx                       ;waking process
2794  =0A63 8B17                      mov dx,[bx]                     ;first process on list
2795  =0A65 85D2                      test dx,dx                      ;is there a process to wake?
2796  =0A67 740C        0A75          jz qa_ret
2797  =0A69 53                        push bx                         ;save list root
2798  =0A6A E850F7      01BD          call no_abort_spec_entry        ;can not abort while in queue
2799  =0A6D 5A                        pop dx                          ;system
2800  =0A6E 9D                        popf                            ;interrupts are ok now -
2801  =0A6F BB4906                    mov bx,offset q_spb             ;present owner and next owner
2802  =0A72 E90AF7      017F          jmp assign_sync_entry           ;have TEMPKEEP on
2803  =
2804  =                   qa_ret:
2805  =0A75 9D                        popf
2806  =0A76 C3                        ret
2807
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2808
2809  =                         eject 1 include que2.rtm
2810  =                     ;*****************************************************************
2811  =                     ;*                                                               *
2812  =                     ;*      Queue system subroutines: Q System must be owned         *
2813  =                     ;*      before calling any of the following routines             *
2814  =                     ;*                                                               *
2815  =                     ;*****************************************************************
2816  =
2817  =                     queverify:              ; check QLR for existence of QD address
2818  =                     ;---------              ----------------------------------------
2819  =                     ;       entry:  U_WRKSEG:DX = offset of QPB
2820  =                     ;       exit:   CX = 0 if queue is found
2821  =                     ;               CX = E_NO_QUEUE error code, if not
2822  =                     ;               BX=SI=QD offset
2823  =                     ;               ES,DX preserved
2824  =
2825  =0A77 06                          push es                         ;save UDA
2826  =0A78 268E062E00                  mov es,u_wrkseg
2827  =0A7D 8BDA                        mov bx,dx                       ;ES:BX->QPB
2828  =0A7F 268B5F02                    mov bx,es:qpb_qaddr[bx]         ;get QD from user's QPB
2829  =0A83 07                          pop es                          ;restore UDA
2830  =0A84 8D367400                    lea si,qlr-q_link
2831  =                     qv_nxt:
2832  =0A88 8B34                        mov si,q_link[si]               ;SI addresses of valid QDs
2833  =0A8A 85F67407      0A95          test si,si ! jz qv_nqf
2834  =0A8E 3BDE75F6      0A88              cmp bx,si ! jne qv_nxt
2835  =0A92 33C9C3                          xor cx,cx ! ret             ;BX=SI=QD offset, return success
2836  =                     qv_nqf:
2837  =0A95 B90900                      mov cx,e_no_queue               ;couldn't find the queue
2838  =0A98 C3                          ret
2839  =
2840  =                     getqdaddr:
2841  =                     ;----------
2842  =                     ;       entry:  DX = offset of QD in U_WRKSEG
2843  =                     ;       exit:   DI = offset of QD in SYSDAT
2844  =                     ;
2845  =                     ; If QD address is within SYSDAT use it.
2846  =                     ; Else get a QD from QDUL and set QF_TABLE flag.
2847  =                     ; If Q_NMSGS=0 in QD then return error.
2848  =                     ; If 0 length buffer needed, zero the Q_BUF field.
2849  =                     ; If buffer space is within SYSDAT use it.  Else get buffer
2850  =                     ; from QMAU.  Return the QD address within SYSDAT.
2851  =
2852  =0A99 52                          push dx                         ;save QD offset
2853  =0A9A 8B1C00                      mov bx,qdlen
2854  =0A9D E89700        0B37          call sysdat_chk
2855  =0AA0 E311          0AB3          jcxz g_qul                      ;CX=0 if not within SYSDAT
2856  =0AA2 26A12E00                      mov ax,u_wrkseg               ;# of paragraphs to U_WRKSEG
2857  =0AA6 8CDB                          mov bx,ds
2858  =0AA8 2BC3                          sub ax,bx                     ;from SYSDAT
2859  =0AAA D3E0                          mov cl,4 ! shl ax,cl          ;make into # of bytes
2860  =0AAC 5F                            pop di                        ;DI=QD in U_WRKSEG
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2861
2862  =0AAF 03F8                        add di,ax               ;make QD relative to SYSDAT
2863  =0AB1 EB2E          0AE1          jmps g_qd
2864  =                           g_qul:
2865  =0AB3 8B3E5E00                    mov di,qul              ;get QD from QUL
2866  =0AB7 85FF                        test di,di              ;DI=unused QD
2867  =0AB9 7505          0AC0          jnz g_gotqd
2868  =0ABB 5A                            pop dx
2869  =0ABC B90700                        mov cx,e_no_qd        ;no QD available
2870  =0ABF C3                            ret
2871  =                           g_gotqd:
2872  =0AC0 8B05                        mov ax,q_link[di]
2873  =0AC2 A35E00                      mov qul,ax
2874  =0AC5 5E                          pop si                  ;SI=QD in U_WRKSEG
2875  =0AC6 57                          push di                 ;save SYSDAT QD
2876  =0AC7 B90E00                      mov cx,qdlen/2
2877  =                                                         ; ES=UDA, DS=SYSDAT
2878  =0ACA 8CD8                        mov ax,ds
2879  =0ACC 268E1E2E00                  mov ds,u_wrkseg         ; DS:SI = QD in U_WRKSEG
2880  =0AD1 068EC0                      push es ! mov es,ax     ; ES:DI = QD in SYSDAT
2881  =0AD4 F3A5                        rep movsw               ; copy QD into QD in SYSDAT
2882  =0AD6 8CC08ED8                    mov ax,es ! mov ds,ax
2883  =0ADA 07                          pop es                  ; ES=UDA, DS=SYSDAT
2884  =0ADB 5F                          pop di                  ; SYSDAT:DI -> New QD
2885  =0ADC 814D041000                  or q_flags[di],qf_table ; Set "from table" flag
2886  =
2887  =                           g_qd:                         ;DI=QD in SYSDAT
2888  =0AE1 33C9                      xor cx,cx                 ;CX=0
2889  =0AE3 394D10                    cmp q_nmsgs[di],cx        ;must have one or more msgs
2890  =0AE6 7505          0AED        jne g_buflen              ;or illegal queue
2891  =0AE8 B90800                      mov cx,e_no_qbuf
2892  =0AEB EB40          0B2D          jmps g_qdfree
2893  =                           g_buflen:                     ;if MX or buffer length is zero
2894  =0AED F745040100                test q_flags[di],qf_mx    ;0 Q_BUF field and return
2895  =0AF2 7505          0AF9        jnz g_mx                  ;CX=0 to indicate success
2896  =0AF4 394D0E                      cmp q_msglen[di],cx
2897  =0AF7 7504          0AFD          jne g_getbuf
2898  =                           g_mx:
2899  =0AF9 894D1A                        mov q_buf[di],cx
2900  =0AFC C3                            ret
2901  =                           g_getbuf:                     ;non 0 length buffer needed
2902  =0AFD 837D1A00                  cmp q_buf[di],0           ;is there a buffer specified?
2903  =0B01 7422          0B25        je g_buf
2904  =0B03 8B450E                      mov ax,q_msglen[di]     ;yes make sure it fits in SYSDAT
2905  =0B06 F76510                      mul q_nmsgs[di]         ;AX=needed buffer space
2906  =0B09 8BD8                        mov bx,ax
2907  =0B0B 8B551A                      mov dx,q_buf[di]
2908  =0B0E E82600        0B37          call sysdat_chk
2909  =0B11 E312          0B25          jcxz g_buf              ;CX=0 not within SYSDAT
2910  =0B13 26A12E00                      mov ax,u_wrkseg       ;# of paragraphs to U_WRKSEG
2911  =0B17 8CD8                          mov bx,ds
2912  =0B19 2BC3                          sub ax,bx             ;from SYSDAT
2913  =0B1B B104D3E0                      mov cl,4 ! shl ax,cl  ;make into # of bytes
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2914
2915  =0B1F 01451A                      add q_buf[di],ax          ;make buffer relative to SYSDAT
2916  =0B22 33C9                        xor cx,cx                 ;indicate success
2917  =0B24 C3                          ret
2918  =
2919  =                           q_buf:                          ;allocate buffer space to queue
2920  =0B25 E83F00        0B67            call qspace
2921  =0B28 E30C          0B36            jcxz q_ret
2922  =0B2A B90800                          mov cx,e_no_qbuf      ;no buffer space
2923  =                           q_qdfree:
2924  =0B2D A15E00                          mov ax,qul            ;error return QD to QUL
2925  =0B30 8905                            mov q_link[di],ax
2926  =0B32 893E5E00                        mov qul,di            ;CX = error code
2927  =                           q_ret:
2928  =0B36 C3                          ret
2929  =
2930  =
2931  =                           sysdat_chk:
2932  =                           ;----------
2933  =                           ;       entry:  DX = offset of data structure to check
2934  =                           ;               relative to U_WRKSEG
2935  =                           ;               BX = length of data structure
2936  =                           ;       exit:   CX <> 0 if within SYSDAT
2937  =                           ;               or wrap around
2938  =                           ;               CX = 0 if not within SYSDAT
2939  =                           ;               SI,DI preserved
2940  =
2941  =                           ;       Check data structure for being within SYSDAT
2942  =                           ;       Also check that data structure doesn't wrap
2943  =                           ;       around at 64K, or 1 megabyte.
2944  =
2945  =0B37 0303                          add dx,bx               ;find end of data structure
2946  =0B39 7229          0B64            jc sc_no                ;check for 64K wrap around
2947  =0B3B 83C20F                          add dx,0fh
2948  =0B3E B104                            mov cl,4              ;round up to next paragraph
2949  =0B40 D3EA                            shr dx,cl             ;1st paragraph after struct relative
2950  =0B42 2603162E00                      add dx,u_wrkseg       ;to U_WRKSEG
2951  =0B47 721B          0B64              jc sc_no              ;next paragraph after struct
2952  =                                                           ;check for 1 MB wrap around
2953  =0B49 2EA10600                          mov ax,sysdat
2954  =0B4D 3BD0                              cmp dx,ax           ;check for below SYSDAT
2955  =0B4F 7213          0B64                jb sc_no
2956  =0B51 050010                              add ax,1000h      ;check for above SYSDAT
2957  =0B54 3B064400                            cmp ax,endseg     ;must be within 64k and ENDSEG
2958  =0B58 7203          0B50                  jb sc_end         ;(use the smaller of the 2)
2959  =0B5A A14400                                mov ax,endseg
2960  =                           sc_end:
2961  =0B5D 3BD0                              cmp dx,ax           ;DX=next paragraph,
2962  =0B5F 7703          0B64                ja sc_no
2963  =0B61 B101                                mov cl,1          ;indicate within SYSDAT
2964  =0B63 C3                                  ret
2965  =                           sc_no:
2966  =0B64 33C9                          xor cx,cx
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2967
2968  =0866 C3                          ret
2969  =
2970  =                           qspace:
2971  =                           ;------
2972  =                           ;       entry:  DI = QD address
2973  =                           ;       exit:   CX = 0 if ok else error code
2974  =                           ;               DI preserved
2975  =                           ;       Allocate buffer space for QD
2976  =                           ;       The QMAU describes the Memory Allocation Unit
2977  =                           ;       for the queues.  QMAU is set up by GENCCPM or GENSYS.
2978  =
2979  =0867 8B450E                        mov ax,q_msglen[di]             ;compute size of buffer
2980  =086A F76510                        mul q_nmsgs[di]                 ;AX=size of request
2981  =086D 33C9                          xor cx,cx
2982  =086F 515150                        push cx ! push cx ! push ax      ;call MALLOC
2983  =0872 5051                          push ax ! push cx               ;with MPB on stack
2984  =0874 8BD48CD0                      mov dx,sp ! mov ax,ss
2985  =0878 8ED8                          mov ds,ax
2986  =087A BB6000                        mov bx,offset qmau
2987  =087D 57                            push di                         ;save QD
2988  =087E B90903                        mov cx,f_maualloc               ;go through OSIF to
2989  =0881 E814F5        0098            call osif                       ;get U_WRKSEG=SS
2990  =0884 5F                            pop di                          ;DI=QD
2991  =0885 2E8E1E0600                    mov ds,sysdat
2992  =088A 83F9007508    0897            cmp cx,0 ! jne qspace_ret
2993  =088F 8BEC                            mov bp,sp
2994  =0891 8B4600                          mov ax,mpb_start[bp]
2995  =0894 89451A                          mov q_buf[di],ax              ;address of buffer
2996  =
2997  =                           qspace_ret:
2998  =0897 83C40A                        add sp,10                       ;pop MPB from stack
2999  =089A C3                            ret
3000  =
3001  =                           remqd:
3002  =                           ;-----
3003  =                           ; Place QD on queue unused list
3004  =                           ;       entry: DS:DI = QD address
3005  =                           ;       exit: none
3006  =
3007  =089B 8B4504                        mov ax,q_flags[di]
3008  =089E 2510007408    08AE            and ax,qf_table ! jz rqd_exit
3009  =08A3 A15E008905                        mov ax,qul ! mov q_link[di],ax
3010  =08A8 893E5E00                          mov qul,di
3011  =08AC EB01          08AF              jmps qrelease
3012  =                           rqd_exit:
3013  =08AE C3                            ret
3014  =
3015  =
3016  =                           qrelease:
3017  =                           ;--------
3018  =                           ;       entry:  DI = QD address
3019  =                           ;       exit :  none
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3020
3021  =                       ;
3022  =                       ;       Release buffer space for QD.
3023  =                       ;       If the released space is ajacent to another
3024  =                       ;       free area, they are joined in the SAT table.
3025  =                       ;       The QMAU describes the Memory Allocation Unit
3026  =                       ;       for the queues.  QMAU is set up by GENCCPM or GENSYS.
3027  =
3028  =
3029  =08AF 8B451A                    mov ax,q_buf[di]
3030  =08B2 B96000                    mov cx,offset qmau
3031  =08B5 505051                    push ax ! push ax ! push cx             ;MFPB on stack
3032  =08B8 B90A03                    mov cx,f_maufree
3033  =08BB 8BD48CD08ED8              mov dx,sp ! mov ax,ss ! mov ds,ax
3034  =08C1 E804F4          0098      call osif
3035  =08C4 83C406                   add sp,6                                ;pop MFPB from stack
3036  =08C7 2E8E1E0600               mov ds,cs:sysdat                        ;DS=SYSDAT
3037  =08CC C3                       ret
3038
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3039
3040  =                         eject 1 include findpd.rtm
3041  =                       ;*******************************************************
3042  =                       ;*
3043  =                       ;*      Find Process Descriptor
3044  =                       ;*
3045  =                       ;*******************************************************
3046  =
3047  =                       ;================       ================================
3048  =                       findpdname_entry:       ; Find Process Descriptor by Name
3049  =                       ;================       ================================
3050  =                       ; Find process by name in thread list
3051  =                       ; Before calling this routine, calling process must
3052  =                       ; own the THRD_SPB (Thread Sync Parameter Block) as
3053  =                       ; interrupts are not turned off.
3054  =                       ;
3055  =                       ;       input:  DX->name in u_wrkseg
3056  =                       ;               BX->thread list root - p_thread
3057  =                       ;       output: BX=pd if found
3058  =                       ;                 =0ffffh if not found
3059  =                       ;               CX=0 if found
3060  =                       ;                 =e_no_pdname if not found
3061  =
3062  =0BCD 06268E062E00              push es 1 mov es,u_wrkseg
3063  =                       fpn_cmpname:
3064  =0BD3 8BF2                      mov si,dx
3065  =0BD5 8B5F02                    mov bx,p_thread[bx]
3066  =0BD8 83FB00741E    0BFB        cmp bx,0 1 je fpn_nomatch
3067  =0BDD B100                        mov cl,0
3068  =0BDF 8D7F08                      lea di,p_name[bx]
3069  =                       fpn_cmplet:
3070  =0BE2 80F908740F    0BF6            cmp cl,8 1 je fpn_found
3071  =0BE7 268A042A05                      mov al,es:[si] 1 sub al,[di]
3072  =0BEC D0E075E3      0BD3              shl al,1 1 jnz fpn_cmpname
3073  =0BF0 FEC14647                          inc cl 1 inc si 1 inc di
3074  =0BF4 EBEC          0BE2                jmps fpn_cmplet
3075  =0BF6 B90000EB06    0C01 fpn_found: mov cx,0 1 jmps fpn_exit
3076  =                       fpn_nomatch:
3077  =0BFB B9140089FFFF              mov cx,e_no_pdname 1 mov bx,0ffffh
3078  =                       fpn_exit:
3079  =0C01 07C3                      pop es 1 ret
3080
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3081
3082  =                       eject 1 include abort.rtm
3083  =                   ;*****************************************************************
3084  =                   ;*
3085  =                   ;*      Terminate and Abort Specify Process Entry Points
3086  =                   ;*
3087  =                   ;*****************************************************************
3088  =
3089  =                   ;===============        ===============
3090  =                   sysreset_entry:         ; System Reset
3091  =                   ;===============        ===============
3092  =0C03 3302                  xor dx,dx
3093  =0C05 B98F00                mov cx,f_terminate
3094  =0C08 E98DF4        0098    jmp osif
3095  =
3096  =                   ;===============        ================================
3097  =                   terminate_entry:        ; Terminate - DX=terminate code
3098  =                   ;===============        ================================
3099  =                   ; This entry point is used for a process to terminate itself.  The
3100  =                   ; code from the label 'TERMINATE:' on, is also used by a process to be
3101  =                   ; terminated when it comes back into context, as set up by
3102  =                   ; abort specified process.
3103  =
3104  =0C0B 891E6800              mov bx,rlr
3105  =0C0F E82702        0E39    call abt_chk
3106  =0C12 E303          0C17    jcxz terminate
3107  =0C14 E98E00        0CA5      jmp term_err
3108  =
3109  =                   terminate:                          ;process terminated by abort
3110  =0C17 8B1E6800              mov bx,rlr                  ;spec resumes execution here
3111  =0C1B FA                    cli                         ;interrupts off during stack switch
3112  =0C1C 8E5710                mov ss,p_uda[bx]            ;reset the stack to top of UDA
3113  =0C1F BC0001                mov sp,ulen
3114  =0C22 FB                    sti
3115  =
3116  =0C23 C6470520             mov p_prior[bx],abt_prior   ;finish termination quickly
3117  =                                                       ;abort spec can't force us
3118  =                                                       ;to reenter terminate when
3119  =                                                       ;we have best priority
3120  =
3121  =0C27 BB5906                mov bx,offset thrd_spb      ;get ownership of thread
3122  =0C2A E8FAF4        0127    call sync_entry             ;list before searching
3123  =
3124  =0C2D 3302                  xor dx,dx
3125  =0C2F BB7000                mov bx,(offset thrdrt)-p_thread
3126  =0C32 A16800                mov ax,rlr
3127  =                   nxtchld:
3128  =0C35 8B5F02                mov bx,p_thread[bx]         ;set all child process's p_parent
3129  =0C38 85DB740A      0C46    test bx,bx | jz nochld      ;field to 0
3130  =0C3C 39471E                  cmp p_parent[bx],ax
3131  =0C3F 75F4          0C35      jne nxtchld
3132  =0C41 89571E                    mov p_parent[bx],dx     ;found a child, 0 its parent field
3133  =0C44 EBEF          0C35        jmps nxtchld
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3134
3135  =                         nochld:
3136  =0C46 BB5906                      mov bx,offset thrd_spb
3137  =0C49 E8FAF4        0146          call unsync_entry
3138  =                                                   ;set parent's child abort flag
3139  =0C4C 8B1E6800                    mov bx,rlr        ;if terminating process's
3140  =0C50 F747068000                  test p_flag[bx],pf_ctlc   ;CTL_C flag is on, ie, we are
3141  =0C55 7411          0C68          jz term_r1        ;terminating from an abort spec
3142  =0C57 8167067FFF                    and p_flag[bx],not pf_ctlc  ;or control C
3143  =0C5C 8B5F1E                        mov bx,p_parent[bx]
3144  =0C5F 85DB7405      0C68          test bx,bx ! jz term_r1
3145  =0C63 814F060008                    or p_flag[bx],pf_childabort
3146  =                         term_r1:
3147  =0C68 26C6061000FE                mov q_error_mode,0feh     ;call BDOS termination
3148  =0C6E B92D05                      mov cx,f_bdosterm
3149  =0C71 E824F4        0098          call osif
3150  =
3151  =0C74 BB4906                      mov bx,offset q_spb
3152  =0C77 E8ADF4        0127          call sync_entry
3153  =0C7A E8E201        0E5F          call rlsmx        ;release all MXqueues
3154  =0C7D BB4906                      mov bx,offset q_spb
3155  =0C80 E8C3F4        0146          call unsync_entry
3156  =
3157  =0C83 BB4106                      mov bx,offset mem_spb     ;get ownership of MEM
3158  =0C86 E89EF4        0127          call sync_entry
3159  =0C89 C606400601                  mov mem_cnt,1             ;keep MEM from freeing
3160  =                                                   ;the MEM_SPB
3161  =
3162  =0C8E BB5906                      mov bx,offset thrd_spb    ;get the MEM sync BEFORE
3163  =0C91 E893F4        0127          call sync_entry           ;the THRD sync to avoid
3164  =                                                   ;deadlock
3165  =
3166  =0C94 B93400                      mov cx,f_freeall          ;free all memory except
3167  =0C97 E8FEF3        0098          call osif                 ;load stack and UDA
3168  =0C9A 8B1E6800                    mov bx,rlr                ;dispatcher does rest of
3169  =0C9E C6470404                    mov p_sta*[bx],ps_term    ;termination
3170  =0CA2 E985F7        045A          jmp dsptch
3171  =
3172  =                         term_err:
3173  =                         ;--------
3174  =                         ;       entry:  CX=1 then can't terminate because KEEP or SYS flag
3175  =                         ;               CX=0ffffh TEMPKEEP on, CTLC turned on
3176  =
3177  =                         ;       called from TERMINATE_ENTRY and ABORT_SPEC_ENTRY
3178  =                         ;       after call to ABT_CHK:
3179  =
3180  =0CA5 497405        0CAD          dec cx ! jz term_err1
3181  =                                                   ;TEMPKEEP and CTLC flag on
3182  =0CA8 33DB8BCB                      xor bx,bx ! mov cx,bx     ;and process will terminate
3183  =0CAC C3                            ret               ;itself, or needs CTLC information
3184  =
3185  =                         term_err1:
3186  =0CAD B92300                      mov cx,e_pd_noterm        ;couldn't term because of
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3187
3188  =0CB0 B8FFFFC3              mov bx,0ffffh ! ret          ;SYS or KEEP flags,
3189  =                                                        ;CTLC is off
3190  =
3191  =                     ;================
3192  =                     abort_spec_entry:              ;Abort the specified process
3193  =                     ;================
3194  =                     ;       entry:  DX = address of APB in the caller's U_WRKSEG
3195  =                     ;       exit:   BX = 0 if success, 0FFFFH if failure
3196  =                     ;               CX = 0 "     "   , error code if failure
3197  =
3198  =                     ; Set up the specified PD for termination when it is next in context.
3199  =                     ; If the running PD is the same as the PD to abort, we can just use
3200  =                     ; the terminate entry point.  Otherwise we use the Abort Parameter Block
3201  =                     ; to find it.  If it cannot be found, name and console must both match,
3202  =                     ; the abort fails.  If the TEMPKEEP flag is on, set the CTLC flag
3203  =                     ; and return.  If the KEEP flag is on and not the TEMPKEEP flag,
3204  =                     ; then fail.
3205  =                     ; If the terminate code in DL on entry, is not 0ffh
3206  =                     ; and the SYS flag is on in the PD to be aborted, then fail.
3207  =                     ; The PD is taken off the list it is attached to via its link field.
3208  =                     ; The terminating PD's priority is set to ABT_PRIOR,
3209  =                     ; and the address of TERMINATE: is put on top of its UDA stack.
3210  =                     ; This forces the terminating process to run next (it has the
3211  =                     ; best priority in the system) and for it to resume execution
3212  =                     ; at TERMINATE: on returning from the dispatcher.
3213  =
3214  =0CB4 52                          push dx
3215  =0CB5 BB5906                      mov bx,offset thrd_spb        ; get ownership of thread
3216  =0CB8 E86CF4          0127        call sync_entry               ; while searching
3217  =0CBB 5E                          pop si                        ; U_WRKSEG:SI->APB
3218  =
3219  =0CBC 1E268E1E2E00                push ds ! mov ds,u_wrkseg
3220  =0CC2 8B1C                        mov bx,apb_pd[si]             ; get PD adr to abort
3221  =0CC4 8B4C02                      mov cx,apb_term[si]           ; get termination/memfree code
3222  =0CC7 8A6404                      mov ah,apb_cns[si]            ; console from APB
3223  =0CCA 1F                          pop ds
3224  =0CCB 51                          push cx                       ; save termination code
3225  =0CCC 85DB7406        0CD6        test bx,bx ! jz abt_findit    ; got a PD ID, but not verified
3226  =0CD0 E81C01          0DEF          call find_pdthrd            ; find it on thread
3227  =0CD3 51                            push cx                     ; save return code
3228  =0CD4 EB0A            0CE0          jmps abt_unsync
3229  =
3230  =                           abt_findit:
3231  =0CD6 83C206                        add dx,offset apb_pdname    ; get adr of named PD
3232  =0CD9 BB7000                        mov bx,offset thrdrt - p_thread ; find it on thread
3233  =0CDC E8FD00          0DDC          call findpdnc               ; look for PD name and console match
3234  =0CDF 51                            push cx                     ; save return code
3235  =                           abt_unsync:
3236  =0CE0 53                          push bx                       ; BX=PD if found
3237  =0CE1 BB5906                      mov bx,offset thrd_spb
3238  =0CE4 E85FF4          0146        call unsync_entry
3239  =0CE7 5B                          pop bx
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3240
3241  =0CE8 59                          pop cx                          ; CX=0 if no PD found
3242  =0CE9 E366          0D51          jcxz abt_err1
3243  =                                                                 ; successfully found PD
3244  =0CEB 5A                          pop dx                          ; DX = terminate code
3245  =0CEC A16800                      mov ax,rlr                      ; we are aborting ourselves
3246  =0CEF 38C3                        cmp ax,bx                       ; jump to TERMINATE_ENTRY
3247  =0CF1 7503          0CF6          jne abt_notus
3248  =0CF3 E915FF        0C08            jmp terminate_entry
3249  =
3250  =                           abt_notus:
3251  =0CF6 C6062206FF                  mov indisp,true                 ; stop dispatching
3252  =                                 ;pushf ! cli                    ; do not let process or another
3253  =                                                                 ; process (NO_ABORT_SPEC) change the
3254  =0CFB 814F068000                  or p_flag[bx],pf_ctlc           ; flags while testing and acting on them.
3255  =0D00 E83601        0E39          call abt_chk                    ; ok to abort this PD ?
3256  =0D03 E305          0D0A          jcxz abt_ok
3257  =                                   ;popf                         ; can't abort it - return
3258  =0D05 E82101        0E29            call ok_disp
3259  =0D08 E898          0CA5            jmps term_err
3260  =
3261  =                                 ; The interrupt window above may be lessened if the TERMINATE_ENTRY
3262  =                                 ; code is moved to the dispatcher and the status is set to PS_TERM.
3263  =                                 ; Turning interrupts back on, the aborting process could come
3264  =                                 ; back into context.  Before turning on TEMPKEEP it would check
3265  =                                 ; its status, if it is PS_TERM, then it goes to the dispatcher
3266  =                                 ; to terminate.
3267  =
3268  =                           abt_ok:                               ; this process will be aborted
3269  =0D0A 8A5704                      mov dl,p_stat[bx]               ; call abort function based on status
3270  =0D0D B60088FA                    mov dh,0 ! mov di,dx
3271  =
3272  =0D11 03FF                        add di,di
3273  =0D13 2EFF95590D                  call cs:abort_tab[di]           ; find via p_link and take PD off its list
3274  =                                 ;popf
3275  =0D18 E80E01        0E29          call ok_disp                    ; process can't come back into context
3276  =                                                                 ; so interrupts are ok
3277  =0D1B E335          0D52          jcxz abt_err2                   ; couldn't find PD on list by indicated
3278  =0D1D C606220600                    mov indisp,false              ; by P_STAT
3279  =0D22 C6470520                      mov p_prior[bx],abt_prior
3280  =                                                                 ; set to low priority
3281  =0D26 06                            push es                       ; save calling process' UDA
3282  =0D27 8E4710                        mov es,p_uda[bx]              ; UDA of PD being aborted
3283  =0D2A BEFE00                        mov si,(ulen-2)               ; reset stack to top of UDA
3284  =0D2D 2689361C00                    mov u_sp,si
3285  =0D32 268C061E00                    mov u_ss,es
3286  =0D37 26C704170C                    mov es:[si],offset terminate
3287  =                                                                 ; TERMINATE: will be executed when
3288  =                                                                 ; terminating process comes back into
3289  =                                                                 ; context
3290  =0D3C 33C0                          xor ax,ax                     ; on exit from dispatcher interrupts
3291  =0D3E 26A34E00                      mov u_flag_sav,ax             ; will be off
3292  =0D42 26C606180000                  mov u_inint,false             ; return into context with a RET and
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3293
3294  =                                                    ; not an IRET
3295  =0048 07                      pop es                 ; ES=calling PD's UDA
3296  =0049 E88200          0DCE    call abt_putdrl
3297  =004C 33D8                    xor bx,bx              ; indicate sucess
3298  =004E E9FFF6          0450    jmp pdisp              ; force abort to happen before we return
3299  =                                                    ; since terminating process has better
3300  =                                                    ; priority
3301  =
3302  =                     abt_err1:
3303  =0051 5A                      pop dx                 ; throw out termination code
3304  =                     abt_err2:
3305  =0052 B91400                  mov cx,e_no_pdname     ; set TERM_ERR for other returns
3306  =0055 BBFFFF                  mov bx,0ffffh
3307  =0058 C3                      ret
3308  =
3309  =                     ; The folowing are the abort handlers for a specific PD status.
3310  =                     ; These labels are entered in the abort_tab(le) in the RTM data area.
3311  =                     ; Interrupts assumed off.
3312  =                     ; For all of the following:
3313  =                     ;       entry:  BX = PD addr to be aborted
3314  =                     ;               AH = cons
3315  =                     ;       exit:   CX=0 if error
3316  =                     ;               BX preserved
3317  =                     ;
3318  =                     ; abort_specified process jump table
3319  =                     ;
3320  =                                                    ; Status
3321  =0059 6F0D            abort_tab       dw      abtrun ; 0  = ready list root
3322  =005B 7E0D                            dw      abtslp ; 1  = poll
3323  =005D 8A0D                            dw      abtdly ; 2  = delay
3324  =005F 7E0D                            dw      abtslp ; 3  = swap
3325  =0061 A30D                            dw      abttrm ; 4  = term
3326  =0063 6F0D                            dw      abtrun ; 5  = sleep
3327  =0065 7E0D                            dw      abtslp ; 6  = dq
3328  =0067 7E0D                            dw      abtslp ; 7  = nq
3329  =0069 4F0D                            dw      abtflg ; 8  = flagwait
3330  =006B 7E0D                            dw      abtslp ; 9  = ciowait
3331  =006D 7E0D                            dw      abtslp ; 10 = sync
3332  =  0016                abt_tablen      equ     offset $ - offset abort_tab
3333  =
3334  =                     abtrun:
3335  =                     ;-------
3336  =                     ; On ready list root or dispatcher ready list
3337  =
3338  =006F BF6800                  mov di,(offset rlr) - p_link
3339  =0072 E86C00          0E01    call find_pd
3340  =0075 E302            0D79    jcxz abtr_drl
3341  =0077 EB4F            0DC8      jmps snip_it
3342  =                     abtr_drl:                      ; wasn't on rlr, try drl
3343  =0079 BF6C00                  mov di,(offset drl) - plink
3344  =007C EB45            0DC3    jmps abt_common
3345  =
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3346
3347  =                       abtslp:
3348  =                       ;------
3349  =                       ; On list indicated by u_dparam
3350  =                       ; Note: a PD in the SY_NEXT field will not be found
3351  =                       ; and cannot be terminated.  Usually the PD has TEMPKEEP on
3352  =                       ; so it never gets this far.
3353  =
3354  =0D7E 068E4710                  push es ! mov es,p_uda[bx]
3355  =0D82 26883E0000                mov di,u_dparam
3356  =0D87 07EB39        0DC3        pop es ! jmps abt_common
3357  =
3358  =                       abtdly:
3359  =                       ;------
3360  =0D8A BF6A00                    mov di,offset dlr - p_link
3361  =0D8D E87100        0E01        call find_pd
3362  =0D90 E33B          0DCD        jcxz abt_tab_err
3363  =0D92 8B34                        mov si,p_link[si]     ; Fix wait field in next PD on dlr
3364  =0D94 85F6                        test si,si
3365  =0D96 7409          0DA1          jz ad_nofix           ; Are there more PDs after the one
3366  =0D98 895418                        mov dx,p_wait[si]   ; being aborted on the DLR ?
3367  =0D9B 035718                        add dx,p_wait[bx]   ; Add wait of aborting PD
3368  =0D9E 895418                        mov p_wait[si],dx   ; New value in next PD on dlr
3369  =                       ad_nofix:
3370  =0DA1 EB25          0DC8          jmps snip_it
3371  =
3372  =                       abttrm:
3373  =                       ;------
3374  =0DA3 C606220600                mov indisp,false
3375  =0DA8 E8A5F6        0450        call pdisp              ; we are terminating already
3376  =0DAB 33C9                      xor cx,cx               ; make sure termination completes
3377  =0DAD EB1E          0DCD        jmps abt_tab_err        ; before returning error
3378  =
3379  =                       abtflg:
3380  =                       ;------
3381  =0DAF 33C9                      xor cx,cx
3382  =0DB1 26893E0000                mov di,u_dparam         ; flag PD field asleep on
3383  =0DB6 3B1D                      cmp bx,[di]             ; if interrupts are allowed
3384  =0DB8 7508          0DC2        jne aflg_err            ; in this code, check DRL
3385  =0DBA FF4502                      inc flg_ignore[di]    ; and RLR if not in flags
3386  =0DBD C705FFFF                    mov flg_pd[di],flag_off
3387  =0DC1 41                          inc cx
3388  =                       aflg_err:
3389  =0DC2 C3                          ret
3390  =
3391  =                       abt_common:
3392  =                       ;----------
3393  =0DC3 E83B00E305    0E01        call find_pd ! jcxz abt_tab_err
3394  =                               ;jmps snip_it
3395  =
3396  =                       snip_it:
3397  =                       ;-------
3398  =                       ; Take PD out of linked list of PDs
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3399
3400  =                       ;       entry:  BX = PD being sniped out
3401  =                       ;               DI = offset of previous PD or offset of root - p_link.
3402  =
3403  =0DC8 8917                     mov dx,p_link[bx]       ; DX = next link
3404  =0DCA 8915                     mov p_link[di],dx
3405  =0DCC C3                       ret                     ; CX<>0
3406  =
3407  =                       abt_tab_err:
3408  =                       ;-----------
3409  =0DCD C3                       ret                     ; CX = 0
3410  =
3411  =                       ;
3412  =                       ; End of abort_tab(le) functions
3413  =                       ;
3414  =
3415  =
3416  =                       abt_putdrl:
3417  =                       ;-------
3418  =                       ; Puts PD on DRL
3419  =                       ;       entry:  BX = PD to insert
3420  =                       ;       exit:   BX preserved
3421  =
3422  =0DCE 9CFA                     pushf ! cli
3423  =0DD0 88166C00                 mov dx,drl
3424  =0DD4 8917                     mov p_link[bx],dx
3425  =0DD6 891E6C00                 mov drl,bx
3426  =0DDA 9DC3                     popf ! ret
3427  =
3428  =                       ; Utility routines for abort entry point
3429  =
3430  =                       findpdnc:
3431  =                       ;--------
3432  =                       ; Find PD by name and console, via thread list.
3433  =                       ;       entry:  BX = offset of thread list root - p_thread
3434  =                       ;               DX = adr of name in u_wrkseg
3435  =                       ;               AH = console number
3436  =                       ;               ownership of thread sync
3437  =                       ;       exit:   BX = PD
3438  =                       ;               CX = 0 failure
3439  =
3440  =0DDC 33C9                     xor cx,cx
3441  =                       nxt_pdname:
3442  =0DDE 50                       push ax                 ; save console number
3443  =0DDF E8E8FD        0BCD       call findpdname_entry
3444  =0DE2 58                       pop ax
3445  =0DE3 E303          0DE8       jcxz fnc_found_one      ; CX = 0 is success from findpdname
3446  =0DE5 33C9                       xor cx,cx
3447  =0DE7 C3                         ret                   ; CX = 0 is failure from this routine
3448  =                       fnc_found_one:                  ; found PD w/ same name
3449  =0DE8 396720                   cmp p_cns[bx],ah        ; chk for same console #
3450  =0DEB 75F1          0DDE         jnz nxt_pdname
3451  =0DED 41C3                     inc cx ! ret            ; success
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3452
3453  =
3454  =                       find_pdthrd:
3455  =                       ;-----------
3456  =                       ; Find PD on thread list
3457  =                       ;       entry:  BX = PD address we want
3458  =                       ;               ownership of thread sync
3459  =                       ;       exit:   CX = 0 if not found
3460  =                       ;               BX prserved
3461  =
3462  =0DFF 33C9                      xor cx,cx
3463  =0DF1 BF7000                    mov di,offset thrdrt - p_thread
3464  =                       fp_next:
3465  =0DF4 8B7D02                    mov di,p_thread[di]
3466  =0DF7 85FF                      test di,di
3467  =0DF9 7405              0E00    jz fp_err
3468  =0DFB 3BDF                        cmp bx,di
3469  =0DFD 75F5              0DF4      jne fp_next
3470  =0DFF 41                            inc cx
3471  =                       fp_err:
3472  =0E00 C3                        ret
3473  =
3474  =
3475  =                       find_pd:
3476  =                       ;-------
3477  =                       ; Find PD through link field, interrupts assumed off
3478  =                       ;       entry:  BX = PD address
3479  =                       ;               DI = offset of list root - p_link
3480  =                       ;       exit:   BX = SI = PD address
3481  =                       ;               DI = Previous PD
3482  =                       ;               CX = 0 if failure
3483  =
3484  =0E01 8935                      mov si,p_link[di]
3485  =0E03 33C9                      xor cx,cx
3486  =                       fpd_nxt_pd:
3487  =0E05 85F6                      test si,si              ; SI could be zero to start
3488  =0E07 7408              0E14        jz  fpd_not_found
3489  =0E09 3BF3                      cmp si,bx               ; Are addresses the same ?
3490  =0E0B 7406              0E13        jz fpd_found
3491  =                       fpd_nxt_lnk:
3492  =0E0D 8BFE                      mov di,si               ; Save previous link
3493  =0E0F 8B34                      mov si,p_link[si]
3494  =0E11 EBF2              0E05    jmps fpd_nxt_pd
3495  =                       fpd_found:
3496  =0E13 41                        inc cx
3497  =                       fpd_not_found:
3498  =0E14 C3                        ret
3499  =
3500  =                       fflgpd:
3501  =                       ;-----
3502  =                       ;       Find offset into flag table of flag waiting PD
3503  =                       ;       entry:  BX = PD
3504  =                       ;       exit:   DI = offset in RTM data of flag
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3505
3506  =                         ;               CX = 0 if failure
3507  =                         ;               BX preserved
3508  =
3509  =0E15 8A0E4A00                    mov cl,nflags
3510  =0E19 32ED                        xor ch,ch
3511  =0E1B 8B3E5600                    mov di,flags
3512  =                         ffp_nxt_flg:
3513  =0E1F 3B1D                        cmp bx,flg_pd[di]       ; assume legal flag
3514  =0E21 7405          0E28          jz ffp_pdfound
3515  =0E23 83C703                        add di,flglen
3516  =0E26 E2F7          0E1F          loop ffp_nxt_flg
3517  =                         ffp_pdfound:
3518  =0E28 C3                          ret                     ; CX is 0 at end of loop instr
3519  =                                                         ; CX <> 0 if found
3520  =
3521  =
3522  =                         ok_disp:
3523  =                         ;-------
3524  =0E29 C606220600                  mov indisp,false
3525  =0E2E 833E6C0000                  cmp drl,0
3526  =0E33 7403          0E38          jz od_ret
3527  =0E35 E818F6        0450            call pdisp
3528  =                         od_ret:
3529  =0E38 C3                          ret
3530  =
3531  =
3532  =                         abt_chk:
3533  =                         ;-------
3534  =                         ;       Check different PD flags for terminate or abort
3535  =                         ;       CTLC flag is on if called from ABORT_SPEC
3536  =                         ;       entry:  BX = PD to possibly abort
3537  =                         ;               DL = termination code
3538  =                         ;               interrupts off if called from ABORT_SPEC
3539  =                         ;       exit:   CX = 00000H if ok to abort
3540  =                         ;                    00001H IF NOT OK TO ABORT
3541  =                         ;                    0FFFFH IF TEMPKEEP
3542  =                         ;               BX = PD as on entry
3543  =                         ;               interrupts unchanged
3544  =
3545  =0E39 33C9                        xor cx,cx
3546  =0E3B 8B4706                      mov ax,p_flag[bx]               ;AX = PD flags
3547  =
3548  =                                                                 ;test TEMPKEEP first
3549  =                                                                 ;for signaling KEEP process
3550  =                                                                 ;to terminate i.e., the TMP
3551  =                                                                 ;while in the CLI
3552  =0E3E A900027406    0E49          test ax,pf_tempkeep ! jz ac_keep
3553  =0E43 0D8000                        or ax,pf_ctlc                 ;signal control C
3554  =0E46 49                            dec cx                        ;temp keep return
3555  =0E47 EB12          0E58            jmps ac_ret
3556  =                         ac_keep:
3557  =0E49 A902007509    0E57          test ax,pf_keep ! jnz ac_no
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3558
3559  =                                                    ;not KEEP, test SYS
3560  =0E4E FEC2740C      0E5E          inc dl ! jz ac_ok  ;if DL = 0ffh -> ok to terminate
3561  =0E52 A901007407    0E5E          test ax,pf_sys ! jz ac_ok  ;DL<>0ffh & not sys-> ok to terminate
3562  =                           ac_no:
3563  =0E57 257FFF                       and ax,not pf_ctlc ;turn off ctlc
3564  =0E5A 41                           inc cx             ;CX=1, can't terminate
3565  =
3566  =                           ac_ret:
3567  =0E5B 894706                       mov p_flag[bx],ax  ;new flags if no termination
3568  =                           ac_ok:
3569  =0E5E C3                           ret
3570  =
3571  =                           rlsmx:
3572  =                           ;------
3573  =                           ;      Release all MX queues owned by the terminating process
3574  =                           ;      called only from TERMINATE_ENTRY
3575  =                           ;      entry:  we own the Queue SYNC
3576  =                           ;      exit:   none
3577  =
3578  =                           ;      To guard against queue deletes, and to make interrupt
3579  =                           ;      windows are smaller, we start over from the beginning of
3580  =                           ;      QLR after we write to an MXqueue that we own.
3581  =
3582  =0E5F 8B1E6800                     mov bx,rlr         ;BX=running process
3583  =                           ;      pushf ! cli
3584  =0E63 BE7400                       mov si,(offset qlr)-q_link
3585  =                           rm_nxt:
3586  =0E66 8B34                         mov si,q_link[si]  ;SI=QD currently checking
3587  =0E68 85F6                         test si,si         ;end of list ?
3588  =0E6A 7420          0E8C           jz rm_done
3589  =0E6C F744040100                     test q_flags[si],qf_mx  ;is it an MX queue ?
3590  =0E71 74F3          0E66             jz rm_nxt
3591  =0E73 3B5C1A                           cmp bx,q_buf[si]  ;do we own it ?
3592  =0E76 75EE          0E66             jne rm_nxt
3593  =                           ;            popf           ;allow interrupts
3594  =0E78 56                             push si            ;put 2-word QPB on stack
3595  =0E79 33C0                           xor ax,ax
3596  =0E7B 50                             push ax
3597  =0E7C 8BD4                           mov dx,sp
3598  =0E7E B98B00                         mov cx,f_qwrite
3599  =0E81 1E                             push ds            ;save DS
3600  =0E82 161F                           push ss ! pop ds   ;DS=SS
3601  =0E84 E811F2        0098             call osif
3602  =0E87 1F                             pop ds             ;restore DS
3603  =0E88 5858                           pop ax ! pop ax    ;throw out 2-word QPB
3604  =0E8A EBD3          0E5F             jmps rlsmx         ;start over from
3605  =                                                       ;top of queue list
3606  =                                                       ;a queue could have been
3607  =                                                       ;deleted while we were writing
3608  =                           rm_done:
3609  =                           ;      popf
3610  =0E8C C3                           ret
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3611
3612
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, ...
SER. #_____

```
3613
3614  =                       eject 1 include patch.cod
3615  =                     ;*********************************************************
3616  =                     ;*
3617  =                     ;*      PATCH AREA  --  128 bytes long
3618  =                     ;*
3619  =                     ;*********************************************************
3620  =
3621  =                     patch:
3622  =0C8D 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;00-0f
3623  =0C93 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3624  =0C99 90909090                nop ! nop ! nop ! nop
3625  =
3626  =0C9D 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;10-1f
3627  =0CA3 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3628  =0CA9 90909090                nop ! nop ! nop ! nop
3629  =
3630  =0CAD 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;20-2f
3631  =0CB3 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3632  =0CB9 90909090                nop ! nop ! nop ! nop
3633  =
3634  =0CBD 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;30-3f
3635  =0CC3 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3636  =0CC9 90909090                nop ! nop ! nop ! nop
3637  =
3638  =0CCD 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;40-4f
3639  =0CD3 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3640  =0CD9 90909090                nop ! nop ! nop ! nop
3641  =
3642  =0CDD 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;50-5f
3643  =0CE3 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3644  =0CE9 90909090                nop ! nop ! nop ! nop
3645  =
3646  =0CED 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;60-6f
3647  =0CF3 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3648  =0CF9 90909090                nop ! nop ! nop ! nop
3649  =
3650  =0CFD 909090909090            nop ! nop ! nop ! nop ! nop ! nop  ;70-7f
3651  =0D03 909090909090            nop ! nop ! nop ! nop ! nop ! nop
3652  =0D09 90909090                nop ! nop ! nop ! nop
3653  =
3654
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3655
3656  =                         eject 1 include vec.fmt
3657  =                       ;*******************************************************
3658  =                       ;*
3659  =                       ;* Interrupt Vectors - to fiddle with the interrupt
3660  =                       ;*       vectors, set the Data Segment Register to 0
3661  =                       ;*       and use the following variables.
3662  =                       ;*
3663  =                       ;*******************************************************
3664  =
3665  =                               JSEG
3666  =
3667  =0000                   i_divide_ip     rw      1       ; int 0
3668  =0002                   i_divide_cs     rw      1
3669  =0004                   i_trace_ip      rw      1       ; int 1
3670  =0006                   i_trace_cs      rw      1
3671  =0008                   i_nomask_ip     rw      1       ; int 2
3672  =000A                   i_nomask_cs     rw      1
3673  =000C                   i_break_ip      rw      1       ; int 3
3674  =000E                   i_break_cs      rw      1
3675  =0010                   i_ovrflw_ip     rw      1       ; int 4
3676  =0012                   i_ovrflw_cs     rw      1
3677  =0014                   i_interrupts    rw      ((osint-5)*2)
3678  =0380                   i_os_ip         rw      1
3679  =0382                   i_os_cs         rw      1
3680  =0384                   i_debug_ip      rw      1
3681  =0386                   i_debug_cs      rw      1
3682
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3683
3684  =                       eject 1 include uda.fmt
3685  =                     ;*****************************************************
3686  =                     ;*
3687  =                     ;*      User Data Area - The User Data Area is an
3688  =                     ;*      extension of the process descriptor but it
3689  =                     ;*      travels with the user.  It contains info
3690  =                     ;*      that is needed only while in context.
3691  =                     ;*
3692  =                     ;*      While in the operating system, The Extra
3693  =                     ;*      Segment register points to the beginning
3694  =                     ;*      of the User Data Area.
3695  =                     ;*
3696  =                     ;*****************************************************
3697  =
3698  =  0060               ud_insys        equ     60h
3699  =
3700  =                       eseg
3701  =                       org     0
3702  =
3703  =0000                 u_dparam        rw      1       ; arg to dispatch
3704  =
3705  =                     ;       this area overlays part of BDOS
3706  =0002                 u_dma_ofst      rw      1       ; BDOS dma offset
3707  =0004                 u_dma_seg       rw      1       ; BDOS dma segment
3708  =0006                 u_func          rb      1       ; actual function number
3709  =0007                 u_searchl       rb      1       ; BDOS search length
3710  =0008                 u_searcha       rw      1       ; BDOS search FCB offset
3711  =000A                 u_searchabase   rw      1       ; BDOS search user's segment
3712  =000C                 u_dcnt          rw      1       ; BDOS directory count
3713  =000E                 u_dblk          rw      1       ; BDOS directory block #
3714  =0010                 u_error_mode    rb      1       ; BDOS error mode
3715  =0011                 u_mult_cnt      rb      1       ; BDOS multi-sector count
3716  =0012                 u_df_password   rb      8       ; BDOS default password
3717  =001A                 u_pd_cnt        rb      1       ; BDOS process count
3718  =  0019               uda_ovl_len     equ     (offset $)-(offset u_dma_ofst)
3719  =                     ;       end of overlay area
3720  =
3721  =
3722  =001B                 u_in_int        rb      1
3723  =001C                 u_sp            rw      1       ; save register area
3724  =001E                 u_ss            rw      1
3725  =0020                 u_ax            rw      1
3726  =0022                 u_bx            rw      1
3727  =0024                 u_cx            rw      1
3728  =0026                 u_dx            rw      1
3729  =0028                 u_di            rw      1
3730  =002A                 u_si            rw      1
3731  =002C                 u_bp            rw      1
3732  =002E                 u_wrkseg        rw      1       ; curr seg addr of buf
3733  =0030                 u_retseg        rw      1       ; usr ES return
3734  =0032                 u_ds_sav        rw      1       ;\
3735  =0034                 u_stack_sp      rw      1       ; usr stack segment
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3736
3737  =0036                 u_stack_ss      rw      1       ; usr stack pointer
3738  =0038                 u_ivectors      rw      4       ; save int 0,1
3739  =0040                 u_unused        rw      2       ;
3740  =0044                 u_ivectors2     rw      4       ; save int 3,4
3741  =0048                 u_es_sav        rw      1       ; > Used during interrupts
3742  =004A                 u_flag_sav      rw      1       ;/
3743  =
3744  =0050                 u_initcs        rw      1
3745  =0052                 u_initds        rw      1
3746  =0054                 u_inites        rw      1
3747  =0056                 u_initss        rw      1
3748  =0058                 u_os_ip         rw      1       ; O.S. vec save
3749  =005A                 u_os_cs         rw      1
3750  =005C                 u_debug_ip      rw      1       ; RTS,Debug Vector Save
3751  =005E                 u_debug_cs      rw      1
3752  =0060                 u_insys         rb      1       ; # times through user_entry
3753  =0061                 u_stat_sav      rb      1       ; used during interrupts
3754  =0062                 u_conccb        rw      1       ; default console's CCB addr
3755  =0064                 u_lstccb        rw      1       ; default list devices CCB addr
3756  =0066                 u_delim         rb      1       ; delimiter for user function 9
3757  =
3758  =                     ;       org     ulen
3759  =                     ;u_8087         rw      47      ; 8087 save area
3760  =                     ;                               ; see dispatcher, loader and terminate
3761
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3762
3763  =                         eject 1 include sysdat.dat
3764  =                       ;*****************************************************
3765  =                       ;*
3766  =                       ;*      System Data Area
3767  =                       ;*
3768  =                       ;*****************************************************
3769  =
3770  =                               CSEG
3771  =                               org     0ch
3772  =                       ;dev_ver
3773  =0000C 06                       db      6       ;development system data version
3774  =                                               ;SYSDAT.CON has 16 byte code segment
3775  =
3776  =                               DSEG
3777  =                               org     0
3778  =
3779  =                               ;
3780  =                               ;This data is initialized by GENCCPM
3781  =                               ;
3782  =
3783  =                               ;Module Table - contains the FAR CALL addresses
3784  =                               ;       of each module for their initialization
3785  =                               ;       and entry routines.
3786  =                               ;
3787  =                               ;       +---+---+---+---+---+---+---+---+
3788  =                               ;       |     entry     |   initialize  |
3789  =                               ;       +---+---+---+---+---+---+---+---+
3790  =                               ;
3791  =                               ;             entry     init
3792  =                               ;             -----     ----
3793  =
3794  =  0000                 module_table    equ     dword ptr (offset $)
3795  =  0000                 supmod          equ     (offset $)
3796  =0000 030000000000              dw      3,0,    0,0     ;SUP
3797        0000
3798  =
3799  =  0008                 rtmmod          equ     (offset $)
3800  =0008 030000000000              dw      3,0,    0,0     ;RTM
3801        0000
3802  =
3803  =  0010                 memmod          equ     (offset $)
3804  =0010 030000000000              dw      3,0,    0,0     ;MEM
3805        0000
3806  =
3807  =  0018                 ciomod          equ     (offset $)
3808  =0018 030000000000              dw      3,0,    0,0     ;CIO
3809        0000
3810  =
3811  =  0020                 bdosmod         equ     (offset $)
3812  =0020 030000000000              dw      3,0,    0,0     ;BDOS
3813        0000
3814  =
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3815
3816  =  0028                  xiosmod         equ     (offset $)
3817  =0028 030C0000000C                       dw      0C03H,0,        0C00H,0 ;XIOS
3818        0000
3819  =
3820  =  0030                  netmod          equ     (offset $)
3821  =0030 030000000000                       dw      3,0,    0,0     ;NET
3822        0000
3823  =
3824  =  0038                  dispatcher      equ     (offset $)
3825  =0038 00000000                           dw      0,0             ;far dispatcher (does IRET)
3826  =
3827  =  003C                  rtm_pdisp       equ     (offset $)
3828  =003C 00000000                           dw      0,0             ;far dispatcher (does RETF)
3829  =
3830  =                                ; location in memory of MP/M-86 or CCP/M-86
3831  =
3832  =0040 0810               osseg           dw      1008h   ;1st parag. of MP/M-86 or CCP/M
3833  =0042 0000               rspseg          dw      0       ;segment of first RSP
3834  =0044 0000               endseg          dw      0       ;1st parag. outside of MP/M or CCP/M
3835  =
3836  =0046 3F                 module_map      db      03fh    ;bit map of modules that exist
3837  =                                                        ; in this system.  low order bit
3838  =                                                        ; corresponds to 1st module in
3839  =                                                        ; module table.  If bit is on,then
3840  =                                                        ; module exists.
3841  =
3842  =                                ; some answers to GENCCPM questions
3843  =
3844  =0047 04                 ncns            db      4       ;# system console devices
3845  =0048 01                 nlst            db      1       ;# system list devices
3846  =0049 05                 nccb            db      5       ;# character control blocks
3847  =004A 20                 nflags          db      20h     ;# flags
3848  =004B 01                 srchdisk        db      1       ;system disk
3849  =004C 0040               mmp             dw      04000h  ;Max Memory per process
3850  =004E 00                 nslaves         db      0       ;Number of network requestors
3851  =004F 00                 dayfile         db      0       ;if 0ffh,display command info
3852  =0050 01                 tempdisk        db      1       ;Temporary disk
3853  =0051 3C                 tickspersec     db      60      ;Number of ticks per second
3854  =
3855  =                                ; data lists created by GENCCPM
3856  =
3857  =0052 0000               free_root       dw      0       ;locked unused list
3858  =0054 0000               ccb             dw      0       ;addr. Console Ctrl Blk Table
3859  =0056 0000               flags           dw      0       ;addr. Flag Table
3860  =0058 2000               mdul            dw      020h    ;Mem descr. Unused List
3861  =005A 0000               mfl             dw      0       ;Memory Free List
3862  =005C 1400               pul             dw      014h    ;Proc. descr. Unused List
3863  =005E 2000               qul             dw      020h    ;QCB Unused List
3864  =                                                        ;MAU for queue buffer info
3865  =0060 0000               qmau            dw      0       ;link
3866  =0062 0000                               dw      0       ;start segment
3867  =0064 0004                               dw      400h    ;length
```

CCP/M-86 2.0 V.1
COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CC-104

```
3868
3869  =0066 0000                        dw      0       ;plist
3870  =
3871  =                           ;
3872  =                           ;This data is initialized at Assembly time
3873  =                           ;
3874  =
3875  =0068 7704           rlr         dw      initpd  ;Ready List Root
3876  =006A 0000           dlr         dw      0       ;Delay List Root
3877  =006C 0000           drl         dw      0       ;Dispatcher Ready List
3878  =006E 0000           plr         dw      0       ;Poll List Root
3879  =0070 0000           scl         dw      0       ;Shared Code List
3880  =0072 7704           thrdrt      dw      initpd  ;Process Thread Root
3881  =0074 9401           qlr         dw      mxloadqd;Queue List Root
3882  =0076 0000           mal         dw      0       ;Memory Alloc List
3883  =
3884  =                           ; Version Information
3885  =
3886  =0078 0000           version     dw      unknown ;addr. version str in SUP code segment
3887  =                                           ;set by GENCCPM if CCP/M
3888  =                    if mpm
3889  =                    bvernum     dw      01130h  ;MPM-86 w/BDOS v3.0
3890  =                    osvernum    dw      01121h  ;MPM-86 V2.1
3891  =                    endif
3892  =
3893  =                    if ccpm
3894  =007A 3114           bvernum     dw      01431h  ;CCP/M w/BDOS 3.1
3895  =007C 2014           osvernum    dw      01420h  ;CCP/M V2.0
3896  =                    endif
3897  =
3898  =                           ; Time of Day Structure
3899  =
3900  =007E                tod         rw      0
3901  =007E 7E06           tod_day     dw      067EH   ;day since 1/1/78 (09 Jul 82)
3902  =0080 12             tod_hr      db      12h     ;hour of day
3903  =0081 00             tod_min     db      00h     ;minute of hour
3904  =0082 00             tod_sec     db      00h     ;second of minute
3905  =
3906  =                           ; info from XIOS
3907  =
3908  =0083 00             ncondev     db      0               ;# console devs in XIOS
3909  =0084 00             nlstdev     db      0               ;# character devs in XIOS
3910  =0085 00             nciodev     db      0               ;# character i/o devices
3911  =                                                        ; supported by XIOS.
3912  =0086 0000           lcb         dw      0               ; list control block address
3913  =0088 0000           openvec     dw      0               ; open file vector
3914  =008A 20             lock_max    db      20h             ; Max Locked Records/process
3915  =008B 20             open_max    db      20h             ; Max Open Files/process
3916  =008C 0000           owner8087   dw      0               ; no one owns it initially
3917  =008E                rw          rw      1               ; RESERVED
3918  =0090 FF             cmod        db      0ffh            ; BDOS Compatibility
3919  =0091 00             ndp8087     db      false           ; true 8087 exits
3920  =                                                        ; (Numeric Data Processor)
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3921
3922  =0092 00000000          err_intercept   dw      0,0             ; BDOS does a callf here
3923  =                                                               ; to print error msgs,
3924  =                                                               ; if second word is <> 0
3925  =0096 4106              slr             dw      offset mem_spb  ; Sync List Root
3926  =0098 000000000000              dw      0,0,0,0         ; RESERVED
3927        0000
3928  =
3929  =
3930  =                               ; SYSENT Table - MP/M-86, CCP/M-86 system function information
3931  =                               ;       The supervisor calls the appropriate module
3932  =                               ;       through this table.
3933  =                               ;
3934  =                               ;        Low Byte  High Byte
3935  =                               ;       +----+----+--------+
3936  =                               ;       |function |flgs|mod|
3937  =                               ;       +----+----+--------+
3938  =                               ;
3939  =                               ;       flgs    - 001h - network intercept
3940  =                               ;                 if on, the network module is called
3941  =                               ;                 first, on return, either the function
3942  =                               ;                 is called or it is considered complete
3943  =                               ;                 depending on the return.
3944  =                               ;       mod     - module number (0-15)
3945  =                               ;       function- function to call within module
3946  =
3947  =                               ;       note: sup function 0 returns not the
3948  =                               ;                 implemented error code to the caller,
3949  =                               ;                 and sup function 1 returns the illegal
3950  =                               ;                 function error code.
3951  =
3952  =                               ; standard CPM-2 functions
3953  =
3954  =                               ;       func,   module
3955  =
3956  =                               org     ((offset $) + 1) and 0fffeh
3957  =
3958  =00A0 0002              sysent  db      0,      rtm     ;  0-system reset
3959  =00A2 0004                      db      0,      cio     ;  1-conin
3960  =00A4 0104                      db      1,      cio     ;  2-conout
3961  =00A6 0001                      db      0,      sup     ;  3-raw conin/aux in
3962  =00A8 0001                      db      0,      sup     ;  4-raw conout/aux out
3963  =00AA 0404                      db      4,      cio     ;  5-list out
3964  =00AC 0504                      db      5,      cio     ;  6-raw conio
3965  =00AE 0001                      db      0,      sup     ;  7-getiobyte
3966  =00B0 0001                      db      0,      sup     ;  8-setiobyte
3967  =00B2 0604                      db      6,      cio     ;  9-conwrite
3968  =00B4 0704                      db      7,      cio     ; 10-conread
3969  =00B6 0804                      db      8,      cio     ; 11-constat
3970  =00B8 0201                      db      2,      sup     ; 12-get version
3971  =00BA 0005                      db      0,      bdos    ; 13-diskreset
3972  =00BC 0105                      db      1,      bdos    ; 14-diskselect
3973  =00BE 0205                      db      2,      bdos    ; 15-file open
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
3974
3975  =00C0 0305                db      3,      bdos    ; 16-file close
3976  =00C2 0405                db      4,      bdos    ; 17-search first
3977  =00C4 0505                db      5,      bdos    ; 18-search next
3978  =00C6 0605                db      6,      bdos    ; 19-file delete
3979  =00C8 0705                db      7,      bdos    ; 20-file read seq
3980  =00CA 0805                db      8,      bdos    ; 21-file write seq
3981  =00CC 0905                db      9,      bdos    ; 22-file make
3982  =00CE 0A05                db      10,     bdos    ; 23-file rename
3983  =00D0 0B05                db      11,     bdos    ; 24-login vector
3984  =00D2 0C05                db      12,     bdos    ; 25-get def disk
3985  =00D4 0D05                db      13,     bdos    ; 26-set dma
3986  =00D6 0E05                db      14,     bdos    ; 27-get alloc vector
3987  =00D8 0F05                db      15,     bdos    ; 28-write protect
3988  =00DA 1005                db      16,     bdos    ; 29-get r/0 vector
3989  =00DC 1105                db      17,     bdos    ; 30-set file attr.
3990  =00DE 1205                db      18,     bdos    ; 31-get disk parm block
3991  =00E0 1305                db      19,     bdos    ; 32-user code
3992  =00E2 1405                db      20,     bdos    ; 33-file read random
3993  =00E4 1505                db      21,     bdos    ; 34-file write random
3994  =00E6 1605                db      22,     bdos    ; 35-file size
3995  =00E8 1705                db      23,     bdos    ; 36-set random record
3996  =00EA 1805                db      24,     bdos    ; 37-reset drive
3997  =00EC 1905                db      25,     bdos    ; 38-access drive
3998  =00EE 1A05                db      26,     bdos    ; 39-free drive
3999  =00F0 1B05                db      27,     bdos    ; 40-file write random w/zero fill
4000  =
4001  =                                       ;CPM-3 extensions
4002  =
4003  =00F2 0001                db      0,      sup     ; 41-Test and Write (NOT IMPLEMENTED)
4004  =                                               ; Would be BDOS func # 28
4005  =00F4 1C05                db      28,     bdos    ; 42-Lock Record
4006  =00F6 1D05                db      29,     bdos    ; 43-Unlock Record
4007  =00F8 1E05                db      30,     bdos    ; 44-Set Multi-sector
4008  =00FA 1F05                db      31,     bdos    ; 45-Set Bdos Error Mode
4009  =00FC 2005                db      32,     bdos    ; 46-Get Disk Free Space
4010  =00FE 0C01                db      12,     sup     ; 47-Chain to Program
4011  =                                               ; In CP/M-86 BDOS func # 34
4012  =0100 2105                db      33,     bdos    ; 48-Flush Buffers
4013  =0102 0101                db      1,      sup     ; 49-
4014  =
4015  =                                       ;CPM-86 extensions
4016  =
4017  =0104 0301                db      3,      sup     ; 50-call xios
4018  =0106 2205                db      34,     bdos    ; 51-set dma base
4019  =0108 2305                db      35,     bdos    ; 52-get dma
4020  =010A 0003                db      0,      mem     ; 53-get max mem
4021  =010C 0103                db      1,      mem     ; 54-get abs max mem
4022  =010E 0203                db      2,      mem     ; 55-alloc mem
4023  =0110 0303                db      3,      mem     ; 56-alloc abs mem
4024  =0112 0403                db      4,      mem     ; 57-free mem
4025  =0114 0503                db      5,      mem     ; 58-free all mem
4026  =0116 0401                db      4,      sup     ; 59-load
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4027
4028  =011d 0101                db      1,      sup     ; 60-
4029  =011A 0101                db      1,      sup     ; 61-
4030  =011C 0101                db      1,      sup     ; 62-
4031  =011E 0101                db      1,      sup     ; 63-
4032  =
4033  =                                 ;CP/NET functions
4034  =
4035  =0120 4007                db      64,     net     ; 64-network login
4036  =0122 4107                db      65,     net     ; 65-network logoff
4037  =0124 4207                db      66,     net     ; 66-network send msg
4038  =0126 4307                db      67,     net     ; 67-network rcv msg
4039  =0128 4407                db      68,     net     ; 68-network status
4040  =012A 4507                db      69,     net     ; 69-get network config addr
4041  =
4042  =                                 ;CP/M-3 extensions
4043  =
4044  =012C 2405                db      36,     bdos    ; 98-Reset Alloc Vector
4045  =012E 2505                db      37,     bdos    ; 99-Truncate File
4046  =0130 2605                db      38,     bdos    ;100-Set Dir Label
4047  =0132 2705                db      39,     bdos    ;101-Return Dir Label
4048  =0134 2805                db      40,     bdos    ;102-Read File XFCB
4049  =0136 2905                db      41,     bdos    ;103-Write File XFCB
4050  =0138 2A05                db      42,     bdos    ;104-Set Date and Time
4051  =013A 2B05                db      43,     bdos    ;105-Get Date and Time
4052  =013C 2C05                db      44,     bdos    ;106-Set Default Password
4053  =013E 0D01                db      13,     sup     ;107-Return Serial Number
4054  =0140 0001                db      0,      sup     ;108-(not implemented)
4055  =0142 1904                db      25,     cio     ;109-Get/Set Console Mode
4056  =0144 1A04                db      26,     cio     ;110-Get/Set Output Delimiter
4057  =0146 1B04                db      27,     cio     ;111-Print Block
4058  =0148 1C04                db      28,     cio     ;112-List Block
4059  =
4060  =                                 ; MP/M functions
4061  =
4062  =014A 0603                db      6,      mem     ;128-mem req
4063  =014C 0603                db      6,      mem     ;129-(same function as 128)
4064  =014E 0703                db      7,      mem     ;130-mem free
4065  =0150 0102                db      1,      rtm     ;131-poll device
4066  =0152 0202                db      2,      rtm     ;132-flag wait
4067  =0154 0302                db      3,      rtm     ;133-flag set
4068  =0156 0402                db      4,      rtm     ;134-queue make
4069  =0158 0502                db      5,      rtm     ;135-queue open
4070  =015A 0602                db      6,      rtm     ;136-queue delete
4071  =015C 0702                db      7,      rtm     ;137-queue read
4072  =015E 0802                db      8,      rtm     ;138-cond. queue read
4073  =0160 0902                db      9,      rtm     ;139-queue write
4074  =0162 0A02                db      10,     rtm     ;140-cond. queue write
4075  =0164 0B02                db      11,     rtm     ;141-delay
4076  =0166 0C02                db      12,     rtm     ;142-dispatch
4077  =0168 0D02                db      13,     rtm     ;143-terminate
4078  =016A 0E02                db      14,     rtm     ;144-create process
4079  =016C 0F02                db      15,     rtm     ;145-set priority
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4080
4081  =016E 0904                    db      9,      cio     ;146-console attach
4082  =0170 0A04                    db      10,     cio     ;147-console detach
4083  =0172 0B04                    db      11,     cio     ;148-set def console
4084  =0174 0C04                    db      12,     cio     ;149-console assign
4085  =0176 0501                    db      5,      sup     ;150-CLI
4086  =0178 0601                    db      6,      sup     ;151-call RPL
4087  =017A 0701                    db      7,      sup     ;152-parse filename
4088  =017C 0D04                    db      13,     cio     ;153-get def console
4089  =017E 0801                    db      8,      sup     ;154-sysdat addr
4090  =0180 0901                    db      9,      sup     ;155-time of day
4091  =0182 1002                    db      16,     rtm     ;156-get PD addr
4092  =0184 1102                    db      17,     rtm     ;157-abort process
4093  =
4094  =                             ; MPM II extensions
4095  =
4096  =0186 0F04                    db      15,     cio     ;158-attach list
4097  =0188 1004                    db      16,     cio     ;159-detach list
4098  =018A 1104                    db      17,     cio     ;160-set list dev
4099  =018C 1204                    db      18,     cio     ;161-Cond. Attach list
4100  =018E 1304                    db      19,     cio     ;162-Cond. Attach Console
4101  =0190 0801                    db      11,     sup     ;163-MP/M Version Number
4102  =0192 1404                    db      20,     cio     ;164-get list dev
4103  =
4104  =
4105  =                             ; Initialized Queues
4106  =
4107  =                             org     ((offset $) + 1) and 0fffeh
4108  =
4109  =0194 7408           mxloadqd        dw      mxdiskqd
4110  =0196 0000                    db      0,0
4111  =0198 0300                    dw      qf_keep+qf_mx
4112  =019A 4D584C6F6164            db      'MXLoad  '
4113        2020
4114  =01A2 000001000000            dw      0,1,0,0,1,0,0
4115        000001000000
4116        0000
4117  =01B0 0000           mxloadqpb       db      0,0
4118  =01B2 940101000000            dw      mxloadqd,1,0
4119  =                    ;                db      'MXLoad  '
4120  =
4121  =                             ; Data Used by Load Program
4122  =
4123  =                             org     ((offset $) + 1) and 0fffeh
4124  =
4125  =01B8 0000           lod_uda         dw      0
4126  =01BA 0000           lod_lstk        dw      0
4127  =01BC 0000           lod_basep       dw      0
4128  =01BE 0000           lod_nldt        dw      0
4129  =01C0 0000           lod_pd          dw      0
4130  =01C2                lod_fcb         rs      36
4131  =01E6 0000           lod_indma       dw      0
4132  =01E8 00             lod_nrels       db      0
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4133
4134  =01E9 00                lod_chain       db      0
4135  =01EA 00                lod_user        db      0
4136  =01EB 00                lod_disk        db      0
4137  =01EC 00                lod_fifty       db      0
4138  =01ED 00                lod_8080        db      0
4139  =01EE 00                lod_1byte       db      0
4140  =01EF 0000              lod_fixrec      dw      0
4141  =01F1 0000              lod_fixrecl     dw      0
4142  =01F3                   lod_dma         rb      dskrecl
4143  =0273                   ldtab           rb      ldtabsiz
4144  =
4145  =0310                   cli_dma_ofst    rw      1
4146  =0312                   cli_dma_seg     rw      1
4147  =0321                   cli_pflag       rw      1
4148  =0323                   cli_chain       rb      1
4149  =0324                   cli_term        rb      1
4150  =0325                   cli_dma         rb      dskrecl         ;dma buffer
4151  =
4152  =                                       ;copy of user's clicb
4153  =03A5                   cli_net         rb      1               ;net
4154  =03A6                   cli_ppd         rw      1               ;parent PD
4155  =03A8                   cli_cmdtail     rb      129             ;command sent
4156  =0429                                   rb      1
4157  =
4158  =042A                   cli_fcb         rb      fcblen+1        ;internal FCB
4159  =
4160  =044B 0000              cli_cuspqpb     db      0,0             ;QPB of command
4161  =044D 00000000                          dw      0,0
4162  =0451 A603                              dw      offset cli_ppd
4163  =0453 242424242424                      db      '$$$$$$$$'
4164        2424
4165  =
4166  =045B 0000              cli_ucb         db      0,0             ;cns,match
4167  =045D 0000                              dw      0               ;pd
4168  =045F 242424242424                      db      '$$$$$$$$'      ;name
4169        2424
4170  =
4171  =0467 A803              cli_pcb         dw      offset cli_cmdtail      ;parse
4172  =0469 2A04                              dw      offset cli_fcb          ;ctl bk
4173  =
4174  =046B 0000              cli_pd          dw      0       ;pd of load prog
4175  =046D 0000              cli_err         dw      0       ;error return
4176  =046F 0000              cli_bpage       dw      0       ;base page
4177  =0471 01                cli_lddsk       db      1       ;load disk
4178  =
4179  =                               ;parent information
4180  =
4181  =0472 00                cli_cns         db      0       ;pd.p_cns save
4182  =0473 00                cli_user        db      0       ;pd.p_dsk save
4183  =0474 00                cli_dsk         ub      0       ;pd.p_user save
4184  =0475 00                cli_err_mode    db      0       ;u_error_mode save
4185  =0476 00                cli_dfil        db      0       ;dayfile flag
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4186
4187  =
4188  =                                ;
4189  =                                ;System Initialization Variables
4190  =                                ;
4191  =
4192  =0477 0000                initpd          dw      0               ;link
4193  =0479 0000                                dw      0               ;thread
4194  =047B 00                                  db      ps_run          ;stat
4195  =047C 01                                  db      1               ;prior
4196  =047D 0500                                dw      pf_sys+pf_kernal;flag
4197  =047F 496E69742020                        db      'Init  '        ;name
4198        2020
4199  =0487 0000                                dw      unknown         ;uda segment
4200  =0489 00                                  db      0               ;disk
4201  =048A 00                                  db      0               ;user
4202  =048B 0000                                db      0,0             ;ldsk,luser
4203  =048D 0000                                dw      0               ;mem
4204  =048F 0000                                dw      0               ;dvract
4205  =0491 0000                                dw      0               ;wait
4206  =0493 0000                                db      0,0             ;org,net
4207  =0495 0000                                dw      0               ;parent
4208  =0497 00                                  db      0               ;cns
4209  =0498 00                                  db      0               ;abort
4210  =0499 0000                                db      0,0             ;cin,cout
4211  =049B 00                                  db      0               ;lst
4212  =049C 000000                              db      0,0,0           ;sf3,4,5
4213  =049F                                     rb      4               ;reserved
4214  =04A3 00000000                            dw      0,0             ;pret,scratch
4215  =
4216  =
4217  =                                ;User Data Area of Init process
4218  =                                ;paragraph aligned
4219  =
4220  =                                org     ((offset $)+0fh) AND 0fff0h
4221  =04B0                     inituda         rb      ulen
4222  =05B0                     init_tos        rw      0
4223  =                                org     offset inituda + ud_insys
4224  =0510 01                                  db      1               ;keep the SUP from doing stack
4225  =                                org     offset init_tos         ;switches
4226  =
4227  =
4228  =                                ; RTM data
4229  =                                ; is word aligned from init uda
4230  =
4231  =05B0 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
4232  =05B6 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
4233  =05BC CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
4234  =
4235  =05C2 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
4236  =05C8 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
4237  =05CE CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
4238  =05D4 CCCCCCCCCCCC                        dw      0cccch,0cccch,0cccch
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4239
4240  =
4241  =05DA CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4242  =05E0 CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4243  =05E6 CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4244  =05EC CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4245  =
4246  =05F2 CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4247  =05F8 CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4248  =05FE CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4249  =0604 CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4250  =
4251  =060A CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4252  =0610 CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4253  =0616 CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4254  =061C CCCCCCCCCCCC                    dw      0ccccch,0ccccch,0ccccch
4255  =
4256  =0622                 dsptchtos       rw      0
4257  =
4258  =0622 00              indisp          db      false   ;?currently in dispatch?
4259  =0623 00              intflag         db      0       ;if 0, interrupts not enabled -
4260  =                                                     ;not implemented
4261  =0624 0000            es_sav          dw      0       ;(staying word aligned)
4262  =0626 0000            bx_sav          dw      0
4263  =0628 0000            ax_sav          dw      0
4264  =
4265  =                             ; MEM Data
4266  =
4267  =062A 0000            beststart       dw      0
4268  =062C 0000            bestlen         dw      0
4269  =062E 0000            bestsi          dw      0
4270  =0630 0000            bestmau         dw      0
4271  =0632 0000            currmau         dw      0
4272  =0634 0000            currsi          dw      0
4273  =0636 000000000000    currmpb         dw      0,0,0,0,0
4274        00000000
4275  =
4276  =
4277  =                     ;       SYNC Parameter Blocks
4278  =
4279  =                     ;       The MEM ENTRY: point uses the following for
4280  =                     ;       mutual exclusion and recursion.
4281  =
4282  =0640 00              mem_cnt         db      0       ;how many times a process has recursivly
4283  =                                                     ;called the memory manager
4284  =
4285  =0641 4906            mem_spb         dw      q_spb   ;link   Mem Sync Parameter Block
4286  =0643 0000                            dw      0       ;owner
4287  =0645 0000                            dw      0       ;wait
4288  =0647 0000                            dw      0       ;next
4289  =
4290  =                     ;       The queue sub-system in the RTM uses the following
4291  =                     ;       structure for mutual exclusion
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4292
4293  =
4294  =0649 5105              q_spb           dw      cli_spb ;link   Queue Sync Parameter Block
4295  =064B 0000                              dw      0       ;owner
4296  =064D 0000                              dw      0       ;wait
4297  =064F 0000                              dw      0       ;next
4298  =
4299  =
4300  =                       ;       The CLI uses the CLI_SPB for mutual exclusion
4301  =
4302  =0651 5906              cli_spb         dw      thrd_spb;link   CLI Sync Parameter Block
4303  =0653 0000                              dw      0       ;owner
4304  =0655 0000                              dw      0       ;wait
4305  =0657 0000                              dw      0       ;next
4306  =
4307  =                       ;       When the thread is accessed, the THRD_SPB must be owned
4308  =                       ;       first.
4309  =
4310  =0659 A00B              thrd_spb        dw      msg_spb ;link   Thread Sync Parameter Block
4311  =065B 0000                              dw      0       ;owner
4312  =065D 0000                              dw      0       ;wait
4313  =065F 0000                              dw      0       ;next
4314  =
4315  =                       ; Currently the order in which the SYNCs must be obtained if
4316  =                       ; more than one is needed is:
4317  =
4318  =                       ;       CLI
4319  =                       ;       QUEUE                   ;called by CLI for RSPs
4320  =                       ;       MEM                     ;called by make queue
4321  =                       ;       THREAD                  ;used from the MEM module
4322  =
4323  =                       ; The SYNCs must be released in reverse order
4324  =                       ; MSG_SPB is used by the BDOS to protect the BDOS error message
4325  =                       ; buffer.  MSG_SPB is in DATA.BDO
4326
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4327
4328  =                         eject 1 include data.bdo
4329  =
4330  =                 ;*********************************************************
4331  =                 ;*
4332  =                 ;*      bdos data area
4333  =                 ;*
4334  =                 ;*********************************************************
4335  =
4336  =  FFFF           CCPMOFF equ     true
4337  =
4338  =                 if 8CPM
4339  =
4340  =                 ;
4341  =                 ;       8086 variables that must reside in code segment
4342  =                 ;
4343  =                 cseg    $
4344  =                 ;
4345  =                 axsave          dw      0       ; register saves
4346  =                 ss_save         dw      0
4347  =                 sp_save         dw      0
4348  =                 stack_begin     dw      endstack
4349  =                 ;
4350  =                 ;       variables in data segment:
4351  =                 ;
4352  =                 dseg    cpmsegment
4353  =                         org     bdosoffset+bdoscodesize
4354  =
4355  =                 header  rs      128
4356  =                         rs      72
4357  =
4358  =                 pag0    dw      0               ;address of user's page zero
4359  =                 ip0     db      0               ;initial page value for ip register
4360  =                 ;
4361  =                 ;     memory control block
4362  =                 ;
4363  =                 umembase        dw      0       ;user'sbase for memory request
4364  =                 umemlg  dw      0               ;length of memory req
4365  =                 contf   db      0               ;flag indicates added memory is avail
4366  =                 ;
4367  =                 ;
4368  =                 hold_info       dw      0       ;save info
4369  =                 hold_spsave     dw      0       ;save user sp during program load
4370  =                 hold_sssave     dw      0       ;save user ss during program load
4371  =
4372  =                 mod8080         db      0
4373  =                 ;
4374  =                 ;       byte i/o variables:
4375  =                 ;
4376  =                 compcol db      0               ;true if computing column position
4377  =                 strtcol db      0               ;starting column position after read
4378  =                 column  db      0               ;column position
4379  =                 listcp  db      0               ;listing toggle
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4380
4381  =                       kbchar  db      0               ;initial key char = 00
4382  =
4383  =                       endif
4384  =
4385  =                               useg
4386  =                       if BNPM
4387  =                               org     0c00h   ;org for MP/M system
4388  =                       endif
4389  =
4390  =                       if BNPM and CCPMOFF
4391  =                               org     0800h   ;org for CCP/M system
4392  =                       endif
4393  =
4394  =                       if BNPM
4395  =0800 0000              rlog            dw      0       ;removeable logged-in disks
4396  =0802 0000              tlog            dw      0       ;removeable disk test login vector
4397  =0804 0000              ntlog           dw      0       ;new tlog vector
4398  =
4399  =                       endif
4400  =
4401  =0806 0000              rodsk           dw      0       ;read only disk vector
4402  =0808 0000              dlog            dw      0       ;logged-in disks
4403  =
4404  =080A 00                open_fnd        db      0               ;open file found in srch_olist
4405  =
4406  =                       ;the following variables are set to zero upon entry to file system
4407  =
4408  =080b 00                fcbdsk          db      0               ;disk named in fcb
4409  =080C 00                parlg           db      0               ;length of parameter block copied
4410  =080D 0000              aret            dw      0               ;adr value to return
4411  =  080D                 lret            equ     byte ptr aret   ;low(aret)
4412  =080F 00                resel           db      0               ;reselection flag
4413  =0810 00                rd_dir_flag     db      0               ;must read/locate directory record
4414  =0811 00                comp_fcb_cks    db      0               ;compute fcb checksum flag
4415  =0812 00                search_user0    db      0               ;search user 0 for file (open)
4416  =0813 00                make_xfcb       db      0               ;make & search xfcb flag
4417  =0814 00                find_xfcb       db      0               ;search find xfcb flag
4418  =0815 0000              xdcnt           dw      0               ;empty directory dcnt
4419  =0817 00                DIR_CNT         DB      0               ;DIRECT I/O COUNT
4420  =0818 00                MULT_NUM        DB      0               ;MULTI-SECTOR COUNT used in bdos
4421  =0819 00                olap_type       db      0               ;record lock overlap type
4422  =081A 00                ff_flag         db      0               ;0ffh xios error return flag
4423  =081b 00                err_type        db      0               ;type of error to print if not zero
4424  =081C                                   rb      4               ;reserved bytes
4425  =  0015                 zerolength      equ     (offset $)-(offset fcbdsk)
4426  =0820 01                mult_sec        db      1               ;multi sector count passed to xios
4427  =0821 00                RELOG           DB      0               ;AUTOMATIC RELOG SWITCH
4428  =
4429  =0822 00                BLK_OFF         DB      0               ;RECORD OFFSET WITHIN BLOCK
4430  =0823 0000              blk_num         dw      0               ;block number
4431  =                       ;
4432  =0825 2708              ua_lroot        dw      offset ua0      ;unallocated block list root
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4433
4434  =                         ;
4435  =0827 20080000            ua0     dw      offset ua1,0
4436  =082B FF00                        db      0ffh,0
4437  =082D 33080000            ua1     dw      offset ua2,0
4438  =0831 FF00                        db      0ffh,0
4439  =0833 39080000            ua2     dw      offset ua3,0
4440  =0837 FF00                        db      0ffh,0
4441  =0839 3F080000            ua3     dw      offset ua4,0
4442  =083D FF00                        db      0ffh,0
4443  =083F 45080000            ua4     dw      offset ua5,0
4444  =0843 FF00                        db      0ffh,0
4445  =0845 00000000            ua5     dw      0,0
4446  =0849 FF00                        db      0ffh,0
4447  =
4448  =084B                     HASH            RB      4               ;HASH CODE WORK AREA
4449  =084F 00                  HASHL           DB      0               ;HASH SEARCH LENGTH
4450  =
4451  =0850 0B                  LOG_FXS         db      11              ;number of log_fxs entrys
4452  =                                         ;       fxi15,fxi17,fxi19,fxi22,fxi23
4453  =0851 020406090A                          db      02h  ,04h  ,06h  ,09h  ,0ah
4454  =                                         ;       fxi30,fxi35,fxi99,fxi100,fxi102,fxi103
4455  =0856 111625262829                        db      11h  ,16h  ,25h  ,26h  ,28h  ,29h
4456  =085C 00000000                            db      0,0,0,0         ;reserved
4457  =0860 07                  rw_fxs          db      7               ;number of rw_fxs entrys
4458  =                                         ;       fxi20,fxi21,fxi33,fxi34,fxi40,fxi42,fxi43
4459  =0861 070814151B1C                        db      07h  ,08h  ,14h  ,15h  ,1bh  ,1ch  ,1dh
4460        1D
4461  =0868 0000                                db      0,0             ;reserved
4462  =086A 02                  sc_fxs          db      2               ;number of sc_fxs entrys
4463  =                                         ;       fxi16,fxi18
4464  =086B 0305                                db      03h  ,05h
4465  =086D 0000                                db      0,0             ;reserved
4466  =
4467  =086F 00                  DEBLOCK_FX      DB      0               ;DEBLOCK FUNCTION #
4468  =0870 00                  PHY_OFF         DB      0               ;RECORD OFFSET WITHIN PHYSICAL RECORD
4469  =0871 0000                CUR_BCBA        DW      0               ;CURRENT BCB OFFSET
4470  =0873 0000                ROOT_BCBA       DW      0               ;ROOT BCB OFFSET
4471  =0875 0000                EMPTY_BCBA      DW      0               ;EMPTY BCB OFFSET
4472  =
4473  =                         if BMPM
4474  =
4475  =0877 0000                P_LAST_BCBA     DW      0               ;PROCESS'S LAST BCB OFFSET
4476  =0879 00                  P_BCB_CNT       DB      0               ;PROCESS BCB COUNT IN BCB LIST
4477  =
4478  =                         endif
4479  =
4480  =087A 00                  fx_intrn        db      0               ;internal BDOS function number
4481  =
4482  =087B 0000                TRACK           DW      0       ;BCB RECORD'S TRACK
4483  =087D 0000                SECTOR          DW      0       ;BCB RECORD'S SECTOR
4484  =
4485  =                         ;       seldsk,usrcode are initialized as a pair
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4486
4487  =087F 00              seldsk          db      0       ;selected disk num
4488  =0880 00              usrcode         db      0       ;curr user num
4489  =
4490  =0881 0000            info            dw      0       ;info adr
4491  =0883 0000            searcha         dw      0       ;search adr
4492  =
4493  =                     ;the following variable order is critical
4494  =
4495  =                     ;variables copied from uda for mp/m                     x
4496  =
4497  =                     ;variables included in fcb checksum for mp/m and cp/m     x
4498  =
4499  =                     ;variables used to access system lock list for mp/m         x
4500  =
4501  =0885 0000            dma_ofst        dw      0       ;dma offset             1
4502  =0887 0000            dma_seg         dw      0       ;dma segment            2
4503  =0889 00              func            db      0       ;bdos function #        3
4504  =088A 00              searchl         db      0       ;search len             4
4505  =
4506  =                     if BMPM
4507  =
4508  =088B 0000            searchaofst     dw      0       ;search adr ofst        5
4509  =088D 0000            searchabase     dw      0       ;search adr base        6
4510  =
4511  =                     endif
4512  =
4513  =088F 0000            dcnt            dw      0       ;directory counter      7
4514  =0891 0000            dblk            dw      0       ;directory block        8 ?? - not used - ??
4515  =0893 00              error_mode      db      0       ;bdos error mode        9
4516  =0894 00              mult_cnt        db      0       ;bdos multi-sector cnt 10
4517  =0895                 df_password     rb      8       ;process default pw    11
4518  =
4519  =                     if BMPM
4520  =
4521  =0890 00              pd_cnt          db      0       ;bdos process cnt      12 1
4522  =
4523  =                     endif
4524  =
4525  =089E 00              high_ext        db      0       ;fcb high extent bits     2
4526  =089F 00              xfcb_read_only  db      0       ;xfcb read only flag      3
4527  =08A0 FF              curdsk          db      0ffh    ;current disk             4 1
4528  =
4529  =                     if BMPM
4530  =
4531  =08A1 00              packed_dcnt     db      0       ;packed dblk+dcnt            2
4532  =08A2 00                              db      0
4533  =08A3 00                              db      0
4534  =08A4 0000            pdaddr          dw      0       ;process descriptor addr     3
4535  =
4536  =                     endif
4537  =
4538  =                             org ((offset $) + 1) and 0fffeh
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4539
4540  =
4541  =                         ;       curtrka - alloca are set upon disk select
4542  =                         ;       (data must be adjacent)
4543  =
4544  =0886 0000               cdrmaxa         dw      0       ;ptr to cur dir max val
4545  =0888 0000               drvlbla         dw      0       ;drive label data byte addr
4546  =088A 0000               buffa           dw      0       ;ptr to dir dma addr
4547  =088C 0000               dpbaddr         dw      0       ;curr disk param block addr
4548  =088E 0000               checka          dw      0       ;curr checksum vector addr
4549  =0890 0000               alloca          dw      0       ;curr alloc vector addr
4550  =0892 0000               DIR_BCBA        DW      0       ;DIRECTORY BUFFER CONTROL BLOCK ADDR
4551  =0894 0000               DAT_BCBA        dw      0       ;DATA BUFFER CONTROL BLOCK ADDR
4552  =0896 0000               HASH_SEG        dw      0       ;HASH TABLE SEGMENT
4553  =  000C                  addlist         equ     12      ;"$-buffa" = addr list size
4554  =
4555  =                         ;       sectpt - offset obtained from disk parm block at dpbaddr
4556  =                         ;       (data must be adjacent)
4557  =
4558  =0898 0000               sectpt          dw      0       ;sectors per track
4559  =089A 00                 blkshf          db      0       ;block shift factor
4560  =089B 00                 blkmsk          db      0       ;block mask
4561  =089C 00                 extmsk          db      0       ;extent mask
4562  =089D 0000               maxall          dw      0       ;max alloc num
4563  =089F 0000               dirmax          dw      0       ;max dir num
4564  =08C1 0000               dirblk          dw      0       ;reserved alloc bits for dir
4565  =08C3 0000               chksiz          dw      0       ;size of checksum vector
4566  =08C5 0000               offsetv         dw      0       ;offset tracks at beginning
4567  =08C7 00                 PHYSHF          DB      0       ;PHYSICAL RECORD SHIFT FACTOR
4568  =08C8 00                 PHYMSK          DB      0       ;PHYSICAL RECORD MASK
4569  =08C9                    endlist         rs      0       ;end of list
4570  =  0011                  dpblist         equ     (offset endlist)-(offset sectpt)
4571  =                                                        ;size
4572  =
4573  =                         ;       local variables
4574  =
4575  =08C9                    common_dma      rb      16      ;copy of user's dma 1st 16 bytes
4576  =08D9 00                 make_flag       db      0       ;make function flag
4577  =08DA 00                 actual_rc       db      0       ;directory ext record count
4578  =08DB 00                 save_xfcb       db      0       ;search xfcb save flag
4579  =08DC 00                 save_mod        db      0       ;open_reel module save field
4580  =08DD 00                 pw_mode         db      0       ;password mode
4581  =08DE 00                 attributes      db      0       ;fcb interface attributes hold byte
4582  =
4583  =                         if MPM
4584  =
4585  =                         ;       number of lock list items required for lock operation
4586  =08DF 010002020101       required_table  db      1,0,2,2,1,1,2,2,1,1,2,2,1,1,2,2
4587          020201010202
4588          01010202
4589  =
4590  =08EF 00                 chk_olist_flag  db      0       ;check | test olist flag
4591  =08F0 0000               lock_sp         dw      0       ;lock stack ptr
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4592
4593  =08F2 00              lock_shell      db      0       ;lock shell flag
4594  =08F3 00              check_fcb_ret   db      0       ;check_fcb return switch
4595  =08F4 00              lock_unlock     db      0       ;lock | unlock function flag
4596  =08F5 00              incr_pdcnt      db      0       ;increment process_cnt flag ??
4597  =08F6 00              free_mode       db      0       ;free lock list entries flag ??
4598  =                                                     ;1=free entries for curdsk
4599  =                                                     ;0=free all entries
4600  =08F7 0000            cur_pos         dw      0       ;current position in lock list
4601  =08F9 0000            prv_pos         dw      0       ;previous position in lock list
4602  =
4603  =08FB 00              dont_close      db      0       ;inhibit actual close flag
4604  =08FC 00              open_cnt        db      0       ;process open file count
4605  =08FD 00              lock_cnt        db      0       ;process locked record count
4606  =08FE 0000            file_id         dw      0       ;address of file' lock list entry
4607  =0900 00              set_ro_flag     db      0       ;set drive r/o flag
4608  =0901 00              check_disk      db      0       ;disk reset open file check flag
4609  =0902 00              flushed         db      0       ;lock list open file flush flag
4610  =
4611  =                     ;free_root, lock_max, open_max initialized by sysgen
4612  =
4613  =0903 5200                            dw      offset free_root
4614  =0905 0000            open_root       dw      0       ;lock list open file list root
4615  =0907 0000            lock_root       dw      0       ;lock list locked record list root
4616  =
4617  =                     endif
4618  =
4619  =0909 0000            sdcnt           dw      0       ;saved dcnt of file's 1st fcb
4620  =090B 0000            sdcnt0          dw      0       ;saved dcnt (user 0 pass)
4621  =
4622  =                     if 6CPM
4623  =
4624  =                     chain_flag      db      0       ;chain flag ??
4625  =                     tod             db      0ffh,0ffh,0ffh,0ffh ??
4626  =
4627  =                     endif
4628  =
4629  =
4630  =090D 00              rmf             db      0       ;read mode flag for open$reel
4631  =090E 00              wflag           db      0       ;xios/bios write flag
4632  =090F 00              dirloc          db      0       ;directory flag in rename, etc.
4633  =0910 00              linfo           db      0       ;low(info)
4634  =0911 00              dminx           db      0       ;local for diskwrite
4635  =0912 00              single          db      0       ;set true if single byte
4636  =                                                     ;alloc map
4637  =0913 00              rcount          db      0       ;record count in curr fcb
4638  =0914 00              extval          db      0       ;extent num and extmsk
4639  =0915 00              vrecord         db      0       ;curr virtual record
4640  =0916 00              ADRIVE          DB      0       ;CURRENT DISK - must preceed arecord
4641  =0917 0000            arecord         dw      0       ;curr actual record
4642  =0919 00                              db      0       ;curr actual record high byte
4643  =091A 0000            ARECORD1        dw      0       ;curr actual block# * blkmsk
4644  =091C 0000            drec            dw      0       ;curr actual directory record
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4645
4646  =091E 0000              CUR_dma_seg     DW      0
4647  =0920 0000              CUR_DMA         DW      0
4648  =
4649  =                       ;       local variables for directory access
4650  =
4651  =0922 00                dptr            db      0       ;directory pointer 0,1,2,3
4652  =  088F                 ldcnt           equ     byte ptr dcnt   ;low(dcnt)
4653  =0923 00                user_zero_pass  db      0       ;search user zero flag
4654  =
4655  =                       ;       shell variables
4656  =
4657  =0924 0000              shell_si        dw      0       ;bdos command offset
4658  =0926 000000            shell_rr        db      0,0,0   ;r0,r1,r2 save area
4659  =
4660  =                       ;       special 8086 variables:
4661  =
4662  =0929 0000              uda_save        dw      0       ;user data area saved value
4663  =092B 0000              parametersegment dw     0       ;user parameter segment
4664  =092D 0000              returnseg       dw      0       ;user return segment
4665  =
4666  =                       ;       error messages
4667  =
4668  =092F 00                err_drv         db      0
4669  =0930 0000              err_pd_addr     dw      0
4670  =
4671  =0932 0D0A43502F4D      dskmsg          db      13,10,'CP/M Error On '
4672        204572726F72
4673        204F6E20
4674  =0942 203A2000          dskerr          db      ': ',0
4675  =
4676  =0946 000060096909      xerr_list       dw      0,permsg,rodmsg,rofmsg,selmsg
4677        78098709
4678  =0950 B309C709DC09                      dw      xe5,xe6,xe7,xe8,xe9,xe10,xe11,xe3
4679        EB09FF09100A
4680        290A9509
4681  =
4682  =0960 446973632049      permsg          db      'Disk I/O',0
4683        2F4F00
4684  =0969 526561642F4F      rodmsg          db      'Read/Only Disk',0
4685        6E6C79204469
4686        736B00
4687  =0978 526561642F4F      rofmsg          db      'Read/Only File',0
4688        6E6C79204669
4689        6C6500
4690  =0987 496E76616C69      selmsg          db      'Invalid Drive',0
4691        642044726976
4692        6500
4693  =
4694  =0995 46696C65204F      xe3             db      'File Opened in Read/Only Mode',0
4695        70656E656420
4696        696E20526561
4697        642F4F6E6C79
```

CCP/M-86 2.0 V.1
COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CC-104

```
4698
4699            20436F6e46500
4700    =
4701    =0953   46696C652043        xe5             db      'File Currently Open',0
4702            757272656E74
4703            6C79204F7065
4704            6E00
4705    =096C   436C6F736520        xe6             db      'Close Checksum Error',0
4706            436865636B73
4707            756020457272
4708            6F7200
4709    =098C   50617373776F        xe7             db      'Password Error',0
4710            726420457272
4711            6F7200
4712    =09E8   46696C652041        xe8             db      'File Already Exists',0
4713            6C7265616479
4714            204578697374
4715            7300
4716    =09FF   496C6C656761        xe9             db      'Illegal ? in FCB',0
4717            6C203F20696E
4718            20463420200
4719    =0A10   4F70656E2046        xe10            db      'Open File Limit Exceeded',0
4720            696C65204C69
4721            6D6974204578
4722            636565646564
4723            00
4724    =0A29   4E6F20526F6F        xe11            db      'No Room in System Lock List',0
4725            6020696E2053
4726            797374656D20
4727            4C6F63683204C
4728            69737400
4729    =
4730    =0A45   0D0A00              crlf_str        db      13,10,0
4731    =0A48   0D0A42646F73        pr_fx           db      13,10,'Bdos Function = '
4732            2046756E6374
4733            696F6E203D20
4734    =0A5A   202020              pr_fx1          db      '   '
4735    =0A5D   2046696C6520        pr_fcb          db      ' File = '
4736            3D20
4737    =0A65                       pr_fcb1         rs      12
4738    =0A71   00                                  db      0
4739    =
4740    =0A72   0D0A44697368        deniedmsg       db      13,10,'Disk reset denied, Drive '
4741            207265736574
4742            20646656L6965
4743            642C20447269
4744            766520
4745    =0A8D   003A                deniedrv        db      0,':'
4746    =0A8F   20436F6E736F                        db      ' Console '
4747            6C6520
4748    =0A98   00                  deniedcns       db      0
4749    =0A99   2050726F6772                        db      ' Program '
4750            616D20
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4751
4752  =0AA2 313233343536        deniedprc       db      '12345678',0
4753        373800
4754  =
4755  =                         if BNPM
4756  =
4757  =                         ;       bdos stack switch variables and stack
4758  =                         ;       used for all bdos disk functions
4759  =
4760  =                                 org     ((offset $) + 1) and 0fffeh
4761  =
4762  =                                 ; 69 word bdos stack
4763  =
4764  =0AAC CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4765  =0AB2 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4766  =0AB8 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4767  =0ABE CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4768  =0AC4 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4769  =0ACA CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4770  =0AD0 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4771  =0AD6 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4772  =0ADC CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4773  =0AE2 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4774  =0AE8 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4775  =0AEE CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4776  =0AF4 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4777  =0AFA CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4778  =0B00 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4779  =0B06 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4780  =0B0C CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4781  =0B12 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4782  =0B18 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4783  =0B1E CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4784  =0B24 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4785  =0B2A CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4786  =0B30 CCCCCCCCCCCC                dw      0cccch,0cccch,0cccch
4787  =0B36                     bdosstack       rw      0
4788  =
4789  =0B36                     save_sp         rw      1
4790  =0B38                     ss_save         rw      1
4791  =0B3A                     sp_save         rw      1
4792  =
4793  =                         ;       local buffer area:
4794  =
4795  =0B3C                     info_fcb        rb      40      ;local user FCB
4796  =0B64                     save_fcb        rb      16      ;fcb save area for xfcb search
4797  =
4798  =0B74 0000                mxdiskqd        dw      0       ;link
4799  =0B76 0000                                db      0,0     ;net,org
4800  =0B78 0300                                dw      qf_keep+qf_mx   ;flags (MX queue)
4801  =0B7A 4D5864697368                        db      'MXdisk  '
4802        2020
4803  =0B82 00000100                            dw      0,1     ;msglen,nmsgs
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4804
4805  =0b88 00000000                    dw      0,0     ;nq,dq
4806  =0b8A 01000000                    dw      1,0     ;msgcnt,out
4807  =0B8E 0000                        dw      0       ;buffer ptr
4808  =
4809  =0b90 00                 mxdiskqpb db      0       ;flgs
4810  =0b91 00                          db      0       ;net
4811  =0b92 7403                        dw      mxdiskqd        ;qaddr
4812  =0b94 0100                        dw      1       ;nmsgs
4813  =0b96 0000                        dw      0       ;buffer
4814  =0b98 4D5864697368                db      'MXdisk  '
4815        2020
4816  =
4817  =                         ;       Error Message sync param block for mutual exclusion
4818  =
4819  =0BA0 0000               msg_spb  dw      0       ;link   Error Message sync parameter block
4820  =0BA2 0000                        dw      0       ;owner
4821  =0BA4 0000                        dw      0       ;wait
4822  =0BA6 0000                        dw      0       ;next
4823  =                         endif
4824  =
4825  =                         if BCPM
4826  =
4827  =                         ;
4828  =                         ;       special 8086 variables:
4829  =                         ;
4830  =                         ioloc           db      0       ;iobyte
4831  =                         user_parm_seg   dw      0       ;holds user parameter seg during load
4832  =                         nallocmem       db      0       ;no. of allocated memory segments
4833  =                         ncrmem          db      0       ;no. of available memory segments
4834  =                         crmem           dw      0,0     ;memory table (16 elements)
4835  =                                         dw      0,0,0,0,0,0
4836  =                                         dw      0,0,0,0,0,0,0,0
4837  =                                         dw      0,0,0,0,0,0,0,0
4838  =                                         dw      0,0,0,0,0,0,0,0
4839  =                         ;
4840  =                         mem_stack_length        equ     40
4841  =                         memstack        rs      mem_stack_length
4842  =                                                 ;8 possible allocations
4843  =                         stbase  equ     word ptr 0
4844  =                         stlen   equu    word ptr 2
4845  =                         ccpflag equ     byte ptr 4
4846  =                         nccpalloc       db      0       ;number of current ccp allocations
4847  =                         mem_stk_ptr     dw      0       ;current memory stack location
4848  =
4849  =                         stackarea       rw      ssize   ;stack size
4850  =                         endstack        rb      0       ;top of stack
4851  =                         ;
4852  =                         endif
4853  =                                         org     0fffh
4854  =                         if CCPMOFF
4855  =                                         org     0bfffh
4856  =                         endif
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4857
4858  =0BFF 00
4859                                    db      0
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4860
4861  =                         eject ! include xiosdat.fmt                    ;SUP 5 - DH - 14APR82
4862  =                       ;*********************************************************
4863  =                       ;*
4864  =                       ;*      Concurrent CP/M XIOS Data Area
4865  =                       ;*
4866  =                       ;*********************************************************
4867  =
4868  =                               DSEG
4869  =
4870  =                       org     0C0CH
4871  =
4872  =0C0C                   tick            rb      1
4873  =                                                       ;see XIOS for format of the rest of the
4874  =                                                       ;XIOS header, most of the variables
4875  =                                                       ;are copied to the System Data Segment
4876  =                                                       ;by GENSYS.
4877
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4878
4879                                    eject 1 end
4880
4881
4882  END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR: 68%
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
ABORTSPECENTRY   0C84 L   939  3192#
ABORTTAB         0D59 V  3273  3321# 3332
ABTCHK           0E39 L  3105  3255  3532#
ABTCOMMON        0DC3 L  3344  3356  3391#
ABTDLY           0D8A L  3323  3358#
ABTERR1          0D51 L  3242  3302#
ABTERR2          0D52 L  3277  3304#
ABTFINDIT        0CD6 L  3225  3230#
ABTFLG           0DAF L  3329  3379#
ABTNOTUS         0CF6 L  3247  3250#
ABTOK            0D0A L  3256  3268#
ABTPRTOR         0020 N   741# 3116  3279
ABTPUTORL        0DCE L  3296  3416#
ABTRDRL          0D79 L  3340  3342#
ABTRUN           0D6F L  3321  3326  3334#
ABTSLP           0D7E L  3322  3324  3327  3328  3330  3331  3347#
ABTTABLERR       0DCD L  3362  3377  3393  3407#
ABTTABLEN        0016 N  3332#
ABTTRM           0DA3 L  3325  3372#
ABTUNSYNC        0CE0 L  3228  3235#
ACKEEP           0E49 L  3552  3556#
ACNO             0E57 L  3557  3562#
ACOK             0E5L L  3560  3561  3568#
ACRET            0E5B L  3555  3566#
ACTUALRC         08DA V  4577#
ADDLIST          000C N  4553#
ADNOFIX          0DA1 L  3365  3369#
ADRIVE           0916 V  4640#
AFLGERR          0DC2 L  3384  3388#
ALLOCA           0880 V  4549#
APBCNS           0004 N   734#  735  3222
APBLEN           000E N   737#
APBNET           0005 N   735#  736
APBPD            0000 N   732#  733  3220
APBPDNAME        0006 N   736#  737  3231
APBTERM          0002 N   733#  734  3221
ARECORD          0917 V  4641#
ARECORD1         091A V  4643#
ARET             080D V  4410# 4411
ASDONE           019C L  1172  1186#
ASRET            019D L  1167  1188#
ASSIGNSYNCENTRY  017F L  1156# 2802
ATTRIBUTES       08DE V  4581#
AXSAV            0628 V  1543  1574  1587  1657  1672  1681  4263#
BCPM             0000 N    68#   69  4338  4622  4825
BDOS             0005 N    90#  233  3971  3972  3973  3975  3976  3977  3978  3979
                        3980  3981  3982  3983  3984  3985  3986  3987  3988  3989
                        3990  3991  3992  3993  3994  3995  3996  3997  3998  3999
                        4005  4006  4007  4008  4009  4012  4018  4019  4044  4045
                        4046  4047  4048  4049  4050  4051  4052
BDOSMOD          0020 N  3811#
BDOSMODBIT       0008 N    99#
BDOSSTACK        0836 V  4797#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
BESTLEN          062C V  4268#
BESTMAU          0630 V  4270#
BESTST           062E V  4269#
BESTSTART        062A V  4267#
BLKMSK           08BB V  4560#
BLKNUM           0823 V  4430#
BLKOFF           0822 V  4429#
BLKSHF           08BA V  4559#
BMPM             FFFF N    69#  4386  4390  4394  4473  4506  4519  4529  4583  4755
BUFFA            08AA V  4546#
BVERNUM          007A V  3889#  3894#
BXSAV            0626 V  1546   1575  1586  1658  1671  1680  4262#
CATTACH          0000 N   813#   814
CBIMF            000C N   823#   824
CCB              0054 V   744   3858#
CCBLEN           002C N   839#
CCOLUMN          0006 N   817#   818
CCUSLEAP         0022 N   836#   837
CCPM             FFFF N    66#   255   746  1794  1978  3893
CCPMOFF          FFFF N  4336#  4390  4854
CDRMAXA          08A6 V  4544#
CFCOFP           0020 N   848#
CFCONPC          0002 N   844#
CFCONOUT         0008 N   846#
CFLAG            0004 N   815#   816
CFLISTCP         0001 N   843#
CFSWITCHS        0004 N   845#
CFVOUT           0010 N   847#
CHBASE           0003 N   706#   707
CHECKA           08AE V  4548#
CHECKDISK        0901 V  4608#
CHECKFCBRET      08F3 V  4594#
CHENTMAX         0008 N   711#
CHFIXREC         0070 N   715#
CHFORM           0000 N   704#   705
CHKOLISTFLAG     03FF V  4590#
CHKSIZ           08C3 V  4565#
CHLBYTE          007F N   714#
CHLEN            0009 N   709#
CHLENGTH         0001 N   705#   706
CHMAX            0007 N   708#   709
CHMIN            0005 N   707#   708
CIO              0004 N    89#   224   225   226   227   228  3959  3960  3963  3964
                          3967  3968  3969  4055  4056  4057  4058  4081  4082  4083
                          4084  4088  4096  4097  4098  4099  4100  4102
CIOMOD           0018 N  3807#
CIOMODBIT        0010 N   100#
CLIACB           0458 V  4166#
CLIBPAGE         046F V  4176#
CLICHAIN         0323 V  4143#
CLICMDTAIL       03A8 V  4155#  4171
CLICNS           0472 V  4181#
CLICUSPOPB       0448 V  4160#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
CLIDFIL          0476 V  4185#
CLIDMA           0325 V  4150#
CLIDMAOFST       031D V  4145#
CLIDMASEG        031F V  4146#
CLIDSK           0474 V  4183#
CLIERR           046D V  4175#
CLIERRMODE       0475 V  4184#
CLIFCB           042A V  4158# 4172
CLILOGSK         0471 V  4177#
CLINET           03A5 V  4153#
CLIPCB           0467 V  4171#
CLIPD            046B V  4174#
CLIPFLAG         0321 V  4147#
CLIPPD           03A6 V  4154# 4162
CLISPB           0651 V  4294  4302#
CLITERM          0324 V  4149#
CLIUSER          0473 V  4182#
CMAXBUFSIZ       0010 N   826#  827
CMIMIC           0008 N   819#  820
CMOD             0090 V  3918#
CMSOURCE         0009 N   820#  821
CNCHAR           0007 N   818#  819
COMMONDMA        08C9 V  4575#
COMPECBCKS       0811 V  4414#
CPC              000A N   821#  822
CPDDAT           0222 L  1307  1309#
CPGPD            022B L  1311  1313#
CPICOPY          036A L  1389  1390  1392  1393  1396  1397  1454#
CPMNXT           0261 L  1343# 1351
CPNM             0282 L  1340  1341  1344  1352#
CPNOCOPY         037B L  1463  1465  1469#
CPSLST           024F L  1329  1331#
CPUDA            022F L  1317#
CPUDA1           02AD L  1369  1371#
CPUDA2           02B6 L  1371  1373#
CPUDA3           02BF L  1373  1375#
CPUDA4           02C8 L  1375  1377#
CQBUFF           0020 N   835#  836
CQPBBUFFPTR      001E N   834#  835
CQPBFILL         0019 N   831#  832
CQPBFLAGS        0018 N   830#  831
CQPBNMSGS        001C N   833#  834
CQPBQADDR        001A N   832#  833
CQUEUE           0002 N   814#  815
CREADQENTRY      0914 L   930  2543#
CREATPROCENTRY   00E2 L   936  1023#
CRLFSTR          0A45 V  4730#
CRSVD            000D N   824#  825
CS               SREG V   907   910   954   960  1744  3036  3273
CSMABORT         0020 N   858#
CSMBACKGROUND    0002 N   854#
CSMBUFFERED      0001 N   852#
CSMCTRLD         0100 N   861#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
CSMCIRLP        0200 N   862#
CSMCIRLS        0080 N   860#
CSMFILEFULL     0040 N   859#
CSMNOSWITCH     0008 N   856#
CSMPURGING      0004 N   855#
CSMSUSPEND      0010 N   857#
CSTATE          000E N   825#  826
CSTRICOL        0005 N   816#  817
CURBCBA         0871 V  4469#
CURDMA          0920 V  4647#
CURDMASEG       091E V  4646#
CURDSK          08A0 V  4527#
CURPOS          08F7 V  4600#
CURRBAU         0632 V  4271#
CURRMPB         0636 V  4273#
CURRST          0634 V  4272#
CUSLEEP         0024 N   837#  838
CVC             0008 N   822#  823
CVCMXQ          0016 N   829#  830
CVENQ           0012 N   827#  828
CVOUTQ          0014 N   828#  829
CVSLEEP         0026 N   838#  839
CWRITEQENTRY    09A3 L   932  2635#
DATBCBA         08F4 V  4551#
DAYFILE         004F V  3851#
DBLK            0891 V  4514#
DCNT            088F V  4513# 4652
DCONT           04F8 L  1660  1662  1665  1669  1675#
DDREI           0727 L  2143  2147#
DEBLOCKFX       086F V  4467#
DEBUGINT        00E1 N    55#
DELAYACT        0558 L  1750  1774#
DELAYENTRY      00B5 L   933   988#
DELE            0593 L  1809  1811#
DELETEQENTRY    08AA L   928  2480#
DELLP           0570 L  1800  1802# 1806
DELO            0584 L  1803  1805  1807#
DENIEDCNS       0A98 V  4748#
DENIEDDRV       0A3D V  4745#
DENIEDMSG       0A72 V  4740#
DENIEDPRC       0AA2 V  4752#
DEVVER          000C V   887#
DEXT            0707 L  1673  2136#
DFPASSWORD      0895 V  4517#
DIRBCBA         08B2 V  4550#
DIRBLK          08C1 V  4564#
DIRCNT          0817 V  4419#
DIRLOC          090F V  4632#
DIRMAX          08BF V  4563#
DISPACT         060D L  1748  1749  1751  1754  1755  1757  1758  1828  1894#
DISPATCHENTRY   00C7 L   934   998#
DISPATCHER      0038 N   905  3824#
DISPIN          0465 L  1644  1647#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
DLOG              0808 V  4402#
DLR               006A V  1798  2246  2250  3360  3876#
DLRNULL           0762 L  2247  2254  2261#
DMADEST           0885 V  4501#
DMASLG            0887 V  4502#
DMENX             0911 V  4634#
DONTCLOSE         08F3 V  4603#
DPBADDR           08AC V  4547#
DPBLIST           0011 N  4570#
DPIR              0922 V  4651#
DREG              091C V  4644#
DRET              0712 L  2137  2140#
DRL               006C V  1068  1146  1148  1445  1446  1578  1662  2008  2010  2143
                          2252  2253  2276  2277  3343  3423  3425  3525  3877#
DRLTDRLR          0686 L  1951  1972  1991  1996# 2018
DRVLOLA           0A38 V  4545#
DS                SREG V  1017  1348  1348  1349  1361  1362  1368  1369  1370  1371
                          1372  1373  1374  1375  1376  1386  1401  1402  1403  1404
                          1405  1410  1411  1412  1413  1414  1416  1417  1418  1419
                          1420  1421  1422  1423  1424  1428  1428  1434  1489  1510
                          1511  1515  1540  1540  1589  1593  1689  1690  1691  1692
                          1693  1694  1695  1696  1702  1715  1853  1855  1857  2103
                          2123  2124  2125  2126  2127  2128  2129  2135  2704  2705
                          2707  2857  2878  2879  2882  2911  2985  2991  3033  3036
                          3219  3219  3223  3599  3600  3602
DSKERR            0942 V  4674#
DSKMSG            0932 V  4671#
DSKRECL           0080 N    48# 4142  4150
DSPTABLE          052C V  1744  1748#
DSPTCH            045A L   995  1047  1525  1614# 2299  3170
DSPTCHTOS         0622 V  1678  4256#
EABORT            0028 N   592#
EACTIVEPD         0022 N   586#
EBADENTRY         0002 N   554#
EBADFNAME         0018 N   576#
EBADFTYPE         0019 N   577#
EBADLOAD          001C N   580#
EBADOPEN          001E N   582#
EBADREAD          001D N   581#
EFIXUPREC         0029 N   593#
EFLAGOVRRUN       0005 N   557# 2270
EFLAGUNDERRUN     0006 N   558# 2300
EILLCNS           0013 N   571#
EILLDISK          0017 N   575#
EILLFLAG          0004 N   556# 2314
EILLLST           0025 N   589#
EILLMD            0018 N   579#
EILLPASSWD        0026 N   590#
EMPTYBCBA         0875 V  4471#
ENCLIP            0016 N   574#
ENCLIQ            0010 N   568#
ENDFREE           05F5 L  1852  1859#
ENDLIST           08C9 V  4569# 4570
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
ENDSEG              0044 V  2957  2959  3034#
ENDATTACH           0024 N   588#
ENDCHAR             001A N   578#
ENDCHSNATCH         0015 N   573#
ENDCSEG             0021 N   585#
ENDMEMORY           0003 N   555#
ENDMISC             0027 N   591#
ENDPD               000C N   564#  1502
ENDPDNAME           0014 N   572#  3077  3305
ENDQBUF             0008 N   560#  2891  2922
ENDQD               0007 N   559#  2869
ENDQUEUE            0009 N   561#  2446  2530  2837
ENDTIMPLEMENTED     0001 N   553#
ENDTOWNER           0020 N   584#
ENDUMD              0012 N   570#
ENTRY               009C L   880   951#
ENULLCMD            001F N   583#
EPDNOTERM           0023 N   587#  3186
EQEMPTY             000C N   566#  2578
EQFULL              000F N   567#  2672
EQINUSE             000A N   562#  2408  2513
EQPROTECTED         000D N   565#  2464  2504
ERRDRV              092F V  4668#
ERRINTERCEPT        0092 V  3922#
ERRORMODE           0093 V  4515#
ERRPDADDR           0930 V  4669#
ERRTYPE             0B1B V  4423#
ES                  SREG V  1361  1362  1362  1401  1434  1462  1464  1509  1512  1515
                            1516  1545  1548  1590  1689  2110  2131  2391  2392  2406
                            2412  2438  2439  2445  2463  2467  2468  2487  2488  2490
                            2491  2569  2570  2571  2576  2588  2589  2591  2612  2662
                            2663  2664  2670  2683  2684  2686  2704  2705  2707  2709
                            2825  2826  2828  2829  2880  2880  2882  2883  3062  3062
                            3071  3079  3281  3282  3285  3286  3295  3354  3354  3356
ESSAV               0624 V  1545  1554  4261#
EXIMSK              0BDC V  4561#
EXIVAL              0914 V  4638#
FALSE               0000 N    46#    65    68  1582  1588  2142  2262  3278  3292  3374
                            3524  3919  4258
FARPUTSP            044C L   909  1598#
FBDOSTERM           052D N   233#  3148
FCALLMIOS           0032 N   148#
FCBDSK              080B V  4408#  4425
FCBLEN              0020 N    49#  4158
FCCONATTCH          00A2 N   193#
FCIOSTAT            0418 N   226#
FCIOTERM            0417 N   225#  1845
FCLOAD              010E N   200#
FCONASSIGN          0095 N   180#
FCONATTACH          0092 N   176#
FCONDETACH          0093 N   178#
FCONOUT             0002 N   107#
FCONPRINT           040E N   224#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
FCONREAD          000A N    115#
FCONWRITE         0009 N    114#
FCQREAD           008A N    168#
FCQWRITE          008C N    170#
FCREATEPROC       0090 N    174#
FDELIM            006L N    158#
FDISP             03CC L    906  1535#
FDISPATCH         008E N    172#
FFCLOSE           0010 N    121#
FFFLAG            031A V   4472#
FFINDPDNAME       0214 N    207#
FFLAGSET          0085 N    163#
FFLAGWAIT         0084 N    162#
FFLOPD            0E15 L   3500#
FFOPEN            000F N    120#
FFPNXTFLG         0E1F L   3512# 3516
FFPPDFOUND        0E28 L   3514  3517#
FFREADRDM         0021 N    139#
FFREADSEQ         0014 N    126#
FFRELALL          003A N    156# 3166
FFREEMEM          0039 N    155#
FGETDEFDISK       0019 N    131#
FILEID            08FE V   4606#
FINDPD            0E01 L   3040  3339  3561  3593  3475#
FINDPDNAMEENTRY   03CD L    942  3048# 3443
FINDPDNC          0DDC L   3233  3430#
FINDPDTHRD        0DEF L   3226  3454#
FINDXFCB          0814 V   4417#
FINFLAGSET        0203 N    204#
FIXGRP            0000 N    719#  720
FIXLEN            0004 N    722#
FIXOFFS           0003 N    721#  722
FIXPARA           0001 N    720#  721
FLAGALT           0596 L   1756  1813#
FLAGBAD           07DB L   2313#
FLAGDEXIT         07FC L   2270  2300  2314  2335#
FLAGEXIT          07F8 L   2259  2262  2267  2274  2281  2294  2299  2328#
FLAGGOOD          07E1 L   2312  2315#
FLAGMIN           0003 N     62#
FLAGOFF           FFFF N    328# 1826  2272  2280  2293  2295  3386
FLAGON            FFFE N    329# 1822  2269  2273  2292
FLAGS             0056 V   2322  3511  3859#
FLAGSEG           0002 N     61#
FLAGSETENTRY      0729 L    925  2232#
FLAGTICK          0001 N     60# 2245
FLAGVALID         07D3 L   2244  2291  2302#
FLAGWAITENTRY     07A6 L    924  2284#
FLAGZERO          00FF N    330# 2266
FLGIGNORE         0002 N    325#  326  2266  2267  3385
FLGLEN            0003 N    326# 3515
FLGPD             0000 N    324#  325  1822  1823  1826  2273  2280  2293  3386  3513
FLOAD             010A N    199#
FLSTATTACH        009E N    189#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
FLSTDETACH      009F N   190#
FLSTOUT         0005 N   110#
FLUSHED         0902 V  4609#
FMALLOC         0080 N   159#
FMAUALLOC       0309 N   217#  2983
FMAUFREE        030A N   218#  3032
FMEMFREE        0082 N   160#  1856
FMLALLOC        0306 N   219#
FMLFREE         030C N   220#
FNAMSIZ         0008 N    52#
FNDFOUNDONE     0DE8 L  3445  3443#
FNDABORT        0217 N   210#
FNDABORTSPEC    0219 N   212#
FOKABORT        0218 N   211#
FPARSEFILENAME  0098 N   183#
FPDFOUND        0E13 L  3490  3495#
FPDNOTFOUND     0E14 L  3488  3497#
FPDNXTLNK       0E0D L  3491#
FPDNXTPD        0E05 L  3486#  3494
FPERR           0E00 L  3467  3471#
FPNCMPLET       0BE2 L  3069#  3074
FPNCMPNAME      03D3 L  3063#  3072
FPNEXT1         0C01 L  3075  3078#
FPNEXT          0DF4 L  3464#  3469
FPNFOUND        0BF6 L  3070  3075#
FPNNOMATCH      0BFB L  3066  3076#
FQMAKE          0086 N   164#
FQOPEN          0087 N   165#
FQREAD          0089 N   167#
FQWRITE         008B N   169#  3598
FRCONIN         0402 N   227#
FRCONOUT        0403 N   228#
FREEMODE        08F6 V  4597#
FREENXT         05C1 L  1849#  1858
FREEPD          01F0 L  1281#  1892
FREEPDEXIT      0219 L  1296  1299#
FREEPDNPD       01F3 L  1289#  1292
FREEPDTRD       01FE L  1291  1293#
FREEROOT        0052 V  3857#  4613
FSETDMA         001A N   132#
FSETDMAB        0033 N   149#
FSETPRIOR       0091 N   175#
FSETRNDREC      0024 N   142#
FSHARE          030B N   216#  1347
FSLEEP          0212 N   205#
FSNOFF          078D L  2272  2276#
FSNON           0781 L  2269  2272#
FSSET           0776 L  2266  2269#
FSYNC           0215 N   208#
FTERMINATE      008F N   173#  3093
FTYPSIZ         0003 N    53#
FUNC            0889 V  4503#
FUNCTION        0058 V   922#   954
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
FUNSYNC          0216 N   209#
FUSERCODE        0020 N   138#
FWAKEUP          0213 N   206#
FWNOFF           07CD L  2295  2300#
FWNON            07B5 L  2292  2295#
FXINTRN          087A V  4480#
GBUF             0825 L  2903  2909  2919#
GBUFLEN          0A5D L  2890  2893#
GETPDADR         0382 L  1310  1474#
GETQDADDR        0A99 L  2384  2840#
GFLAGON          05AB L  1822  1825#
GGETBUF          0AFD L  2897  2901#
GGUTQD           0AC0 L  2867  2871#
GMX              0AF9 L  2895  2898#
GPDBAD           039C L  1487  1496#
GPDDONE          03C9 L  1494  1520#
GPDHAVE          03FC L  1501  1504#
GQD              0AE1 L  2863  2887#
GQDFREE          0B2D L  2892  2923#
GQUL             0AB3 L  2855  2864#
GRET             0B36 L  2921  2927#
HASH             084B V  4448#
HASPL            084F V  4449#
HASHSEG          08B6 V  4552#
HIGHEXT          089E V  4525#
IBREAKCS         000E V  3674#
IBREAKIP         000C V  3673#
IDEBUGCS         0386 V  3681#
IDEBUGIP         0384 V  3680#
IDIVIDECS        0002 V  3668#
IDIVIDEIP        0000 V  3667#
INTERRUPTS       0014 V  3677#
INCRPDCNT        08F5 V  4596#
INDISP           0622 V  1541  1544  1588  1644  1647  2142  3251  3278  3374  3524
                         4258#
INFO             0881 V  4490#
INFOFCB          0B3C V  4795#
INIT             0042 L   879   902#
INITPD           0477 V  3875  3880  4192#
INITTOS          05B0 V  4222# 4225
INITUDA          04C0 V  4221# 4223
INOMASKCS        000A V  3672#
INOMASKIP        0008 V  3671#
INSERTPROCESS    061A L  1769  1902  1905# 1990  2017
INSEXIT          0641 L  1927  1929#
INSNPD           0621 L  1918# 1926
INSNXT           0635 L  1923  1924  1926#
INSOUT           063A L  1919  1922  1925  1927#
INTDISP          0478 L  1562  1578  1659#
INTDISPEXIT      0411 L  1566# 2139
INTFLAG          0623 V  4259#
IOAUXIN          0005 N   262#
IOAUXOUT         0006 N   263#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
IOCONIN          0001 N   258#  277#
IOCONOUT         0002 N   259#  278#
IOCONST          0000 N   257#  276#
IOFLUSH          000C N   269#  272   3014  302
IOLIST           0004 N   261#  279#
IOLISTST         0003 N   260#
IOPOLLDEV        000D N   270#  295# 1981
IOREAD           000A N   267#  287#
IOSCS            0382 V  3679#
IOSELDSK         0009 N   266#  283#
IOSIP            0380 V  1394  1711  2119  3678#
IOSTATLINE       0008 N   265#
IOSWITCH         0007 N   264#
IOVRELWCS        0012 V  3676#
IOVRELWIP        0010 V  3675#
IOWRITE          000B N   268#  288#
ITRACECS         0006 V  3670#
ITRACEIP         0004 V  3669#
LCB              0086 V  3912#
LCBLEN           000A N   865#
LDCNT            088F V  4652#
LDTAB            0273 V  4143#
LDTABSIZ         00AA N    63# 4143
LINFO            0910 V  4633#
LOCKCNT          08FD V  4605#
LOCKMAX          008A V  3914#
LOCKROOT         0907 V  4615#
LOCKSHELL        08F2 V  4593#
LOCKSP           08F0 V  4591#
LOCKUNLOCK       08F4 V  4595#
LODBOMO          01ED V  4138#
LODBASEP         018C V  4127#
LODCHAIN         0159 V  4134#
LODDISK          01EB V  4136#
LODDMA           01F3 V  4142#
LODFCB           01C2 V  4130#
LODFIFTY         01EC V  4137#
LODFIXREC        01EF V  4140#
LODFIXREC1       01F1 V  4141#
LODINDMA         01E6 V  4131#
LODLBYTE         01FE V  4139#
LODLSTK          018A V  4126#
LODNEXT          018E V  4128#
LODRELS          01E8 V  4132#
LODPD            01C0 V  4129#
LODUDA           0188 V  4125#
LODUSER          01EA V  4135#
LOGEXS           0850 V  4451#
LRET             0800 V  4411#
MAPLEN           0006 N   415#
MAPMAX           0000 N   412#  413
MAPSAT           0002 N   413#  414
MAPSTART         0004 N   414#  415
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
MAKEFLAG          0809 V  4576#
MAKEQENTRY        0301 L   926  2576#
MAKEXFCB          0813 V  4416#
MEL               0076 V  3882#
MAXALL            08BD V  4562#
MDLEN             000A N   347#
MDUL              0058 V  3860#
MEM               0003 N    38#  216   217   218   219   220   333  4020  4021  4022
                               4023  4024  4025  4062  4063  4064
MEMCNT            0640 V  1863  3159  4282#
MEMMOD            0010 N  3803#
MEMMODBIT         0004 N    98#
MEMSPB            0641 V  3157  3925  4285#
MFCODE            0004 N   386#
MFL               005A V  3861#
MFLOAD            0001 N   384#
MFPBLEN           0004 N   400#
MFPBPB            0002 N   399#  400
MFPBSTART         0000 N   398#  399
MFSHARE           0002 N   385#
MFUDADDLY         0040 N   390#
MLENGTH           0004 N   344#  345   355
MLINK             0000 N   342#  343   353
MMP               004C V  3849#
MODENTRY          0000 N   243#  244   965
MODINIT           0004 N   244#  245
MODLEN            0008 N   245#
MODULEMAP         0046 V  3836#
MODULETABLE       0000 N  3794#
MPBFLAGS          0008 N   377#  379
MPBLEN            000A N   379#
MPBMAX            0004 N   375#  376
MPBMIN            0002 N   374#  375
MPBPOADR          0006 N   376#  377
MPBSTART          0000 N   373#  374  2994
MPLIST            0006 N   345#  346   356
MPM               0000 N    65#   66   275   743  1332  1788  1975  3888
MSFLAGS           0006 N   356#
MSGSPB            08A0 V  4310  4819#
MSLENGTH          0004 N   355#
MSLINK            0000 N   353# 1343
MSMAU             0008 N   357#
MSSTART           0002 N   354# 1346  1854
MSTART            0002 N   343#  344   354
MULTCNT           0894 V  4516#
MULTNUM           0818 V  4420#
MULTSEL           0820 V  4426#
MUNUSED           0008 N   346#  347   357
MXDISKQD          0874 V  4109  4798# 4811
MXDISKQPB         0890 V  4809#
MXLOADQD          0194 V  3881  4109# 4118
MXLOADQPB         0180 V  4117#
NASAVED           0184 L  1211  1213  1215#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
NASPEC          0143 L  1209#  1230
NCCB            0049 V  3846#
NCLODEV         0085 V  3910#
NCRS            0047 V  3844#
NCUNDEV         0083 V  1333   3908#
NDPR087         0091 V  3919#
NET             0007 N    92#  4035   4036   4037   4038   4039   4040
NETMOD          0030 N  3820#
NETMODELT       0040 N   102#
NFLAGS          004A V  2312   3509   3847#
NLST            0048 V  3845#
NLSTDEV         0084 V  1329   3909#
NOARGRTENTRY    019E L   945   1192#  2756
NOARGRTSPECENTRY 01BD L  947   1221#  2798
NOCHLD          0C46 L  3129   3135#
NODISP          044A L  1541   1593#
NODTSP2         04A0 L  1666   1670#
NOPROT          039A L  2459   2462   2466#
NOTICK          076A L  2245   2265#
NSLAVES         004E V  3850#
NTLOG           0804 V  4397#
NXIOSFUNCS      000C N   272#   302#
NXTCHLD         0C35 L  3127#  3131   3133
NXTPDNAME       0DDE L  3441#  3450
NXTTICK         075F L  2248   2255   2259#
NXTTPD          073E L  2249#  2256
OARESTORE       01D3 L  1253   1256#
OARET           01ED L  1258   1262   1268#
ODRET           0E38 L  3526   3528#
OFFSETV         08C5 V  4566#
OKARGRTENTRY    01C1 L   946   1234#  2773
OKDISP          0E29 L  3258   3275   3522#
OLAPTYPE        0819 V  4421#
OPENCNT         08FC V  4604#
OPENEND         080A V  4404#
OPENMAX         0088 V  3915#
OPENSENTRY      0857 L   927   2429#
OPENROOT        0905 V  4614#
OPENVEC         0088 V  3913#
OSIF            0098 L   958#  1349   1845   1856   2989   3034   3094   3149   3167   3601
OSINT           00E0 N    54#    55   3677
OSSEG           0040 V  3832#
OSVERNUM        007C V  3890#  3895#
OWNERBDS7       008C V  3916#
PARORT          0021 N   491#   492
PCKEDDCNT       08A1 V  4531#
PARAMETERSEGMENT 092B V 4663#
PARLG           080C V  4409#
PATCH           0E8D L  3614   3621#
PBCBCNT         0879 V  4476#
PCM11           0001 N   537#
PCMCTLC         0008 N   540#
PCMCTLD         0080 N   542#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
PCMCILS          0002 N   538#
PCMOD            0018 N   486#  487
PCMRCOI          0004 N   539#
PCMRSX           0300 N   543#
PCMS             0020 N   490#  491  3449
PCONMODE         0022 N   492#  493  1322
PDADDR           08A4 V  4534#
PDCMI            089D V  4521#
PDENTRY          00D8 L   930  1014#
PDISP            0450 L  1002  1011  1026  1079  1602  1605# 2146  3298  3375  3527
PDLEN            0030 N    58#  1485  1513
PDSK             0012 N   481#  482
PERMSG           0960 V  4676  4682#
PF8087           8000 N   532#
PFACTIVE         0100 N   525#
PFCHILDABORT     0800 N   528#  1327  3145
PFCTLC           0080 N   524#  1261  3140  3142  3254  3553  3563
PFCTLD           0400 N   527#
PFDSKLD          2000 N   530#
PFKEEP           0002 N   517#  3557
PFKERNAL         0004 N   518#  4196
PFLAG            0006 N   478#  479  1212  1216  1260  1261  1295  1327  1518  1770
                          1915  1928  2279  2461  2500  3140  3142  3145  3254  3546
                          3567
PFNOCTLS         1000 N   529#
PFNOKBD          4000 N   531#
PFPURE           0008 N   519#
PFRAW            0040 N   522#
PFRESOURCE       0020 N   521#  1327  1770  1915  1928  2279
PFSYS            0001 N   516#  2461  2500  3561  4196
PFTABLE          0010 N   520#  1295  1518
PFTEMPKEEP       0200 N   526#  1212  1216  1260  3552
PHYMSK           08C8 V  4568#
PHYOFF           0870 V  4468#
PHYSHF           08C7 V  4567#
PLASTBPA         0877 V  4475#
PLDSK            0014 N   483#  484
PLINK            0000 N   474#  475  1066  1068  1071  1073  1076  1077  1141  1143
                          1147  1173  1175  1297  1314  1445  1505  1666  1668  1733
                          1734  1798  1802  1807  1807  1823  1840  1884  1886  1901
                          1918  1930  1930  1943  1949  1982  1987  1987  1988  2010
                          2016  2250  2252  2276  3338  3343  3360  3363  3403  3404
                          3424  3484  3493
PLR              006E V   963  1660  1943  3878#
PLST             0024 N   493#  494  1328  1335
PLUSER           0015 N   484#  485
PMEM             0016 N   485#  486  1340  1341  1342  1851
PNAME            0008 N   479#  480  3068
PNAMSIZ          0008 N    50#   51    52   480   737
POLLDANOTHER     064A L  1947#  1994
POLLDNEXT        0682 L  1984  1993#
POLLENTRY        00A5 L   923   977#
POLLIT           0662 L  1969  1971  1973#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
PPAREN          001E N   489#   490   1321   3130   3132   3143
PPRET           002C N   494#   495
PPRIOR          0005 N   477#   478   1069   1667   1669   1920   1921   1970   1971   3116
                         3279
PREGL           0A5D V  4735#
PREGAT          0A65 V  4737#
PPTY            0A4B V  4731#
PREXT           0A5A V  4734#
PROCCREAT       021A L  1026   1303#  1450
PRVPOS          08F9 V  4601#
PSCTORAIT       0009 N   511#
PSCRATCH        002E N   495#   496   1044   1771
PSDELAY         0002 N   504#   992
PSDO            0006 N   508#  2583
PSFLAGWAIT      0008 N   510#  2697
PSNO            0007 N   509#  2677
PSPOLL          0001 N   503#   984
PSRUN           0000 N   502#  1078   1145   1553   1610   1665   1900   1989   2015   2251
                        2277   4194
PSSLEEP         0005 N   507#  1045
PSSWAP          0003 N   505#
PSSYNC          000A N   512#  1108
PSTAT           0004 N   476#   477    992   1045   1078   1145   1551   1553   1585   1610
                        1665   1742   1771   1900   1989   2015   2251   2277   2297   3169
                        3269
PSTERM          0004 N   506#  3169
PTHREAD         0002 N   475#   476   1288   1290   1294   1294   1440   3065   3125   3128
                        3232   3463   3465
PTKCNT          0019 N   487#   488   1210   1214   1217   1250   1254   1257   1269   1326
PUDA            0010 N   480#   481   1356   1357   1548   2103   3112   3282   3354
PUL             005C V  1297   1298   1499   1505   3862#
PUSER           0013 N   482#   483
PWAIT           001A N   488#   489    982   1804   1808   1810   1976   1979   2248   2255
                        3366   3367   3368
PWMODE          080D V  4580#
PWSCRTCH        002E N   496#
QARET           0A75 L  2796   2804#
QASSIGNSYNC     0A5F L  2616   2712   2777#
QBUF            001A N   640#   641   2593   2603   2692   2702   2899   2902   2907   2915
                        2995   3029   3591
QDERR1          08D2 L  2496   2504#
QDERR2          090A L  2514   2531#
QDFOUND         08FC L  2520   2522#
QDLEN           001C N   641#  2853   2875
QDMOD           0807 L  2519   2529#
QDMOD           08EE L  2517#  2521
QDOK1           08D9 L  2498   2502   2507#
QDS             0012 N   636#   637   2414   2511   2582   2711
QDRET           090D L  2527   2533#
QFDEV           0040 N   652#
QFHIDE          0004 N   647#  2458   2497
QFKEEP          0002 N   646#  2495   4111   4800
QFLAGS          0004 N   632#   633   2458   2495   2497   2595   2689   2885   2894   3007
```

CCP/M-86 2.0 V.1
COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER.# CC 104

```
                    3589
QFMX        0001 N   645#  2595  2689  2894  3589  4111  4800
QFRPL       0020 N   651#
QFRSP       0008 N   649#
QFTABLE     0010 N   650#  2635  3008
QLINK       0000 N   629#   630  2595  2596  2419  2440  2443  2516  2518  2523
                     2524  2830  2832  2872  2925  3009  3584  3586
QLR         0074 V  2393  2418  2420  2440  2516  2830  3584  3881#
QMAU        0060 V  2986  3030  3865#
QMCHK       080C L  2390#
QMERR       0851 L  2386  2409  2423#
QMGD        0835 L  2397  2411#
QMGOTQD     080C L  2385  2387#
QMNXT       0815 L  2394#  2404
QMRET       0854 L  2422  2425#
QMSGCNT     0016 N   638#   639  2416  2572  2614  2665  2697  2710
QMSGLEN     000E N   634#   635  2592  2687  2896  2904  2979
QMSGOUT     0018 N   639#   640  2417  2602  2613  2696
QNAME       0006 N   633#   634  2400  2401  2452
QNAMSIZ     0008 N    51#   634   679  2399  2451
QNET        0002 N   630#   631
QNMSGS      0010 N   635#   636  2608  2666  2698  2700  2889  2905  2980
QNQ         0014 N   637#   638  2415  2512  2615  2676
QOCMP       0871 L  2444  2449#
QOERR       08A3 L  2447  2465  2472#
QONQD       0865 L  2442#  2456
QORET       08A6 L  2470  2474#
QORG        0003 N   631#   632
QPBBUFFPTR  0006 N   677#   678  2591  2636
QPBFLGS     0000 N   673#   674
QPBLEN      0010 N   679#
QPBNAME     0008 N   678#   679  2453
QPBNET      0001 N   674#   675
QPBNMSGS    0004 N   676#   677
QPBQADDR    0002 N   675#   676  2467  2490  2571  2664  2828
QRELEASE    08AF L  3011  3016#
QREND       0985 L  2596  2600  2609  2611#
QRERR       0998 L  2567  2579  2623#
QRLMSG      096F L  2593  2601#
QRREADIT    0951 L  2573  2590#
QRVER       0925 L  2566  2568#
QRWAIT      0935 L  2575#
QRWAIT1     093F L  2577  2580#
QSPACE      0867 L  2920  2970#
QSPACERET   0897 L  2992  2997#
QSPB        0649 V  2757  2771  2601  3151  3154  4285  4294#
QSYNC       0A43 L  2382  2437  2509  2562  2653  2747#
QUEVERIFY   0A77 L  2563  2655  2817#
QUL         005E V  2865  2873  2924  2926  3009  3010  3863#
QUNSYNC     0A51 L  2426  2475  2534  2618  2624  2714  2719  2740  2762#
QVNQF       0A95 L  2833  2836#
QVNXT       0A88 L  2831#  2834
QWAIT       0A35 L  2584  2678  2725#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
QWEND           0A1E L  2690  2693  2708#
QWERR           0A2L L  2659  2673  2713#
QWLMSG          09FB L  2688  2695#
QWMOVE          0A09 L  2699  2701#
QWVER           09E4 L  2658  2661#
QWWAIT          09C6 L  2669#
QWWAITI         09D0 L  2671  2674#
QWWRITETI       09E2 L  2667  2685#
RCOUNT          0913 V  4637#
RDOIREFLAG      0810 V  4413#
READQ           0916 L  2540  2548#
READQEMTKY      0910 L   929  2538#
RELOC           0821 V  4427#
REMQD           0B9B L  2407  2525  3001#
REMQDMPLDISABLE 080F V  4586#
RESEL           080F V  4412#
RESTORE         0B93 L  2109#
RETURNSEG       092D V  4664#
RLOG            0300 V  4395#
RLR             0068 V   980   991  1008  1018  1042  1098  1126  1165  1207  1248
                        1320  1546  1583  1609  1664  1732  1733  1839  1841  1850
                        1883  1835  1901  1968  1988  2016  2025  2296  2460  2499
                        2597  3104  3110  3126  3139  3168  3245  3338  3582  3875#
RLSMX           0E5F L  3153  3571# 3604
RMDONE          0E8C L  3588  3608#
RMF             090D V  4630#
RMNXT           0E66 L  3585# 3590  3592
RODMSG          0969 V  4676  4684#
RODSK           0806 V  4401#
ROFMSG          0978 V  4676  4687#
ROGTOCBA        0873 V  4470#
RQDEXIT         0B0E L  3008  3012#
RSPSEG          0042 V  3833#
RTN             0002 N    87#  204   205   206   207   208   209   210   211   212
                         869   969  1273  1525  2152  2343  2809  3040  3082  3958
                        4065  4066  4067  4068  4069  4070  4071  4072  4073  4074
                        4075  4076  4077  4078  4079  4091  4092
RTNMOD          0008 N  3799#
RTNMODBIT       0002 N    97#
RTNPOISP        003C N   908  3327#
RTNRET          0097 L   955#
RWFXS           0860 V  4457#
SAVEFCB         0864 V  4796#
SAVEMOD         03DC V  4579#
SAVESP          0836 V  4789#
SAVEXFCB        03DB V  4578#
SCEND           0B5D L  2958  2960#
SCFXS           086A V  4462#
SCHEDULE        0646 L  1772  1811  1824  1892  1902  1933# 2031
SCL             0070 V  3879#
SCNO            0864 L  2946  2951  2955  2962  2965#
SDCNT           0909 V  4619#
SDCNTO          0908 V  4620#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
SEARCHA          0035 V  4491#
SEARCHBASE       088D V  4509#
SEARCHDFST       088B V  4508#
SEARCHL          0888 V  4504#
SEARCHUSERD      0812 V  4415#
SECTOR           0870 V  4483#
SECTPT           0898 V  4558# 4570
SELDSK           087F V  4487#
SELMSG           0987 V  4676  4690#
SERIAL           003C V   899#
SETPRIORITY      00CC L   937  1005#
SETRDFLAG        0900 V  4607#
SGOTIT           0144 L  1103  1105  1111#
SHELLRR          0926 V  4658#
SHELLST          0924 V  4657#
SINGLE           0912 V  4635#
SLEEPACT         0542 L  1753  1760#
SLEEPENTRY       00E8 L   940   985  1029#  1109  2742
SLR              0096 V  1870  3925#
SNIPIT           00C8 L  3341  3370  3396#
SPBDPD           0000 N   361#  362
SPBPD            0002 N   362#  363
SPBSTART         0004 N   363#
SPSAVE           0B3A V  4347# 4791#
SRCHDISK         0048 V  3848#
SS               SREG V  1346  1676  1678  1855  2133  2984  3033  3112  3600
SSSAVE           0B38 V  4346# 4790#
SUP              0001 N    86#  199   200  3961  3962  3965  3966  3970  4003  4010
                         4013  4017  4026  4028  4029  4030  4031  4053  4054  4085
                         4086  4087  4089  4090  4101
SUPERVISOR       0008 N   883#  960
SUPMOD           0000 N  3795#
SUPMODBIT        0001 N    96#
SWITCH           06A1 L  2009  2020#
SWITCH1          06AD L  2030  2032#
SYLIAK           0000 N   689#  690  1870  1872
SYNCENTRY        0127 L   943  1084# 2756  3122  3152  3158  3163  3216
SYNCLEN          0008 N   693#
SYNEXT           0006 N   692#  693  1132  1184
SYOWNER          0002 N   690#  691  1101  1106  1128  1150  1166
SYSDAT           0006 V   882# 1540  1678  1715  2135  2392  2953  2991  3036  3763
SYSDATCHK        0B37 L  1486  2854  2908  2931#
SYSENT           00A0 V   969  3958#
SYSRESETENTRY    0C03 L   922  3090#
SYWAIT           0004 N   691#  692  1107  1138  1142
TEMPDISK         0050 V  3852#
TERMACT          0552 L  1752  1830#
TERMERR          0CA5 L  3107  3172# 3259
TERMERRT         0CAD L  3180  3185#
TERMINATE        0C17 L  3106  3109# 3285
TERMINATEENTRY   0C08 L   935  1264  3097# 3248
TERMRT           0C68 L  3141  3144  3146#
THRDRT           0072 V  1286  1439  1441  3125  3232  3463  3880#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
THRDSPL         0659 V  3121  3136  3162  3215  3237  4302  4310#
TICK            0C0C V  1795  2262  4872#
TICKSPERSEC     0051 V  3853#
TLOG            0802 V  4396#
TNSYNC          055D L  1871# 1873
TOD             007E V  3900# 4625#
TODDAY          007C V  3901#
TODHR           0080 V  3902#
TODLEN          0005 N    59#
TODMIN          0081 V  3903#
TODSEC          0082 V  3904#
TRACK           0878 V  4492#
TRUE            FFFF N    45# 1414  1541  1544  1549  1644  1647  1795  2137  3251
                      4336
TSYNCDONE       05FA L  1874  1880#
U8CB7LEN        015C N    57#
UA0             0827 V  4432  4435#
UA1             082D V  4435  4437#
UA2             0833 V  4437  4439#
UA3             0839 V  4439  4441#
UA4             083F V  4441  4443#
UA5             0845 V  4443  4445#
UALROOT         0825 V  4432#
UAX             0020 V  1690  2129  3725#
UBP             002C V  1696  2123  3731#
UBX             0022 V  1691  2123  3726#
UCBNCB          0062 V  1423  3754#
UCX             0024 V  1692  2124  3727#
UDADVLEN        0019 N  3718#
UDASAVE         0929 V  4662#
UDBLK           000E V  3713#
UDCNT           000C V  3712#
UDEBUGS         005E V  3751#
UDEBUGIP        005C V  3750#
UDELAY          0066 V  1410  3756#
UDPASSWORD      0012 V  1363  3716#
UDT             0028 V  1694  2127  3729#
UDTMSYS         0060 N  3698# 4223
UDMADEST        0002 V  3706# 3718
UDMASEC         0004 V  1413  3707#
UDPARAM         0000 V   993  1043  1421  1767  1799  1820  2298  3355  3382  3703#
UDSSAV          0032 V  1417  1550  1589  3734#
UDX             0026 V  1693  2125  3729#
UERRORMODE      0010 V  1411  3147  3714#
UESSAV          004C V  1420  1555  1590  3741#
UFLAGSAV        004E V  1422  1643  2141  3291  3742#
UFUNC           0006 V  3708#
UINIAT          0013 V  1414  1549  1582  2137  3292  3722#
UINITCS         0050 V  1369  1370  1430  3744#
UINITDS         0052 V  1371  1372  1416  3745#
UINITES         0054 V  1375  1376  1419  3746#
UINITSS         0056 V  1373  1374  1402  3747#
UINSYS          0060 V  3752#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
UIVECTORS          0038 V  1388  1703  2111  3738#
UIVECTORS2         0044 V  3740#
ULEN               0100 N    56#   57  3113  3283  4221
ULSTCCS            0064 V  1424  3755#
UMULTCNT           0011 V  1413  3715#
UNKNOWN            0000 N    47#  3885  4197
UNSYNCENTRY        0146 L   944  1116#  1676  2772  3137  3155  3238
UOSCS              005A V  3749#
UOSTP              0058 V  1395  1711  2113  3748#
UPDCNT             001A V  1412  3717#
URETSEG            0030 V  1017  3735#
USEARCHA           0008 V  3710#
USEARCHACASS       000A V  3711#
USEARCHL           0007 V  3709#
USER               0000 N    85#   107   110   114   115   120   121   126   131   132
                             138   139   142   148   149   155   156   158   159   160
                             162   163   164   165   167   168   169   170   172   173
                             174   175   176   178   180   183   189   190   193
USERZEROPASS       0023 V  4653#
USI                002A V  1695  2126  3730#
USOWN              016D L  1136  1144#
USP                001C V  1405  1429  1676  2134  3284  3723#
USRCODE            0080 V  4488#
USRET              017E L  1129  1152#
USS                001E V  1403  1428  1676  2133  3285  3724#
USTACKSP           0034 V  1404  3736#
USTACKSS           0036 V  3737#
USTATSAV           0061 V  1552  1584  3753#
USWAIT             015D L  1134  1137#
USZERO             017A L  1140  1149#
UUNUSED            0040 V  3739#
UWRKSEG            002E V  1355  1488  1511  2439  2488  2570  2589  2663  2684  2826
                           2856  2879  2910  2950  3062  3219  3732#
VERSION            0078 V  3886#
VRECORD            0915 V  4639#
WAKEUPENTRY        00FF L   941  1050#
WFLAG              090E V  4631#
WKEOUT             0125 L  1065  1080#
WRITEQ             09A5 L  2632  2640#
WRITEQENTRY        099F L   931  2630#
WUOPL              0112 L  1070#  1073
WUORLEND           011A L  1072  1075#
XCODCA             0001 N   312#   313
XCODX              0003 N   313#   314
XCSFUNC            0000 N   311#   312
XCSLEN             0005 N   314#
XDCNT              0015 V  4418#
XE10               0A1C V  4678  4719#
XE11               0A29 V  4678  4724#
XE3                0995 V  4678  4694#
XE5                09B3 V  4678  4701#
XE6                09C7 V  4678  4705#
XE7                09DC V  4678  4709#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
XE8                0928 V  4678  4712#
XE9                09FF V  4678  4716#
XEXPLIST           0946 V  4670#
XFCBREADONLY       089F V  4526#
XIOS               0006 N    91#
XIOSIF             002E L   963#  1791  1981
XIOSMOD            0028 N   966   3616#
XIOSRIDRIT         0020 N   101#
ZERULENGTH         0015 N  4425#
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____